Uma Solução Modularizada e Plugável para Indexação de Informações sobre Trabalhos Acadêmicos escritos em Latex

Daniel do Espirito Santo Correia[1], **Luis Paulo da Silva Carvalho**[1]

[1]Instituto Federal de Educação, Ciência e Tecnologia da Bahia (IFBA)
Campus Vitória da Conquista

danncorreia95@hotmail.com, luispscarvalho@gmail.com

Abstract. ***Context:*** *This work presents the development of a modular and pluggable solution for indexing information on academic papers written in LaTeX.* ***Goals:*** *The main objective of this work was to create a web system, called InProLa (Indexing and Processing of LaTeX), using a pluggable architecture capable of processing and indexing scientific papers written in LaTeX, as well as making these papers available so they can be searched by their content.* ***Method:*** *A General/Generic Architecture was defined to support the addition of functional and independent plugins. The architecture was instantiated as a Web Application, InProLa, using the JavaScript programming language and modern frameworks. To test InProLa, recent undergraduate thesis papers, written in article format, from the Bachelor's Degree in Information Systems at IFBA, Vitória da Conquista campus, were used.* ***Results:*** *InProLa was successfully developed, and its source code is publicly available, allowing for its evolution through the addition of plugins and reuse in the creation of other software solutions that can benefit from plugin-oriented architecture.* ***Conclusion:*** *The resulting Web Application, InProLa, provides evidence that it is possible to use a plugin-oriented Software Architecture to process and make available information obtained from LaTeX documents.*

Resumo. ***Contexto:*** *Este trabalho apresenta o desenvolvimento de uma solução modularizada e plugável para indexação de informações sobre trabalhos acadêmicos escritos em LaTeX.* ***Objetivo:*** *O principal objetivo deste trabalho foi criar um sistema web, chamado InProLa (Indexação e Processamento de Latex), utilizando uma arquitetura plugável, capaz de processar e indexar trabalhos científicos escritos em LaTex, bem como disponibilizar tais trabalhos de forma que possam ser pesquisados pelos seus conteúdos.* ***Método:*** *Foi definida uma Arquitetural Geral/Genérica para dar suporte à adição de plugins funcionais e independentes. A partir daí a Arquitetura foi instanciada na forma de uma Aplicação Web, a InProla, utilizando linguagem de programação, Javacript, e frameworks modernos. Para testar o InProLa, foram utilizados trabalhos de conclusão de curso recentes, escritos na forma de artigos, do Bacharelado em Sistemas de Informação do IFBA, campus Vitória da Conquista.* ***Resultados:*** *O InProLa foi desenvolvido com sucesso e o seu código-fonte se encontra disponbilizado para acesso público, possibilitando sua evolução através da adição de plugins e reuso durante a criação de outras soluções de software que possam se beneficiar da sua arquitetura.* ***Conclusões:*** *A Aplicação Web resultante,*

inProLa, fornece indícios de que é possível utilizar uma Arquitetura de Software orientada a plugins para realizar o processamento e disponbilização de informações obtidas a partir de documentos Latex.

1. Introdução

O Tex foi desenvolvido no final dos anos 70 com o intuito de criar textos com melhor qualidade na representação gráfica, porém era considerado muito técnico pelos seus usuários e exigia um maior conhecimento da plataforma para ser utilizada. Por volta da década de 80, o LaTeX foi criado por Leslie Lamport [Alonso and Berti 2019]. A sua criação teve como objetivo adicionar mais comandos ao Tex, permitindo que os seus usuários conseguissem produzir trabalhos com alta qualidade tipográfica e um layout profissional.

O funcionamento do LaTeX foi baseado em um conjunto de macros, nas quais se adicionam informações e definem-se instruções de formatação das mesmas. Ele não é um processador de palavras. Ao invés disso, ele encoraja os autores a não se preocuparem com a aparência ao escrever, mas a se concentrarem em criar o conteúdo correto [Project 2024].

Sendo o LaTeX muito utilizado para a elaboração de documentos técnicos e científicos e por trazer facilidades de formatação e padronização, a *startup* Overleaf, fundada em 2012, desenvolveu uma plataforma parcialmente gratuita onde usuários conseguem escrever um documento e pré-visualizá-lo, além de permitir compartilhá-lo com outras pessoas para que possam realizar suas contribuições. Segundo a desenvolvedora [Overleaf 2012], a plataforma possui, hoje, quinze milhões de usuários ao redor do mundo, entre eles pesquisadores, estudantes e professores. Isto evidencia que um grande volume de informações se encontra atualmente mantido em documentos LaTeX e que ter acesso indexado a tais informações pode ser vantajoso. Dessa forma, este projeto de pesquisa idealiza e desenvolve uma plataforma online para que os docentes e discentes possam ter acesso a esse conteúdo de forma prática.

Segundo [Cosentino 2023], arquitetura plugável ou *plugin architecture* é uma das muitas soluções que visam trabalhar com modularidade, permitindo que extensões sejam integradas a sistemas sem a necessidade de alterações na estrutura e no código. Essa estrutura pode ser dividida em 3 partes: a aplicação provedora ou *host application*, a interface de *plugins* ou *plugin interface* e os *plugins*.

Para este trabalho foi utilizada uma arquitetura plugável, pois ela irá permitir a evolução dos diferentes processamentos sobre os textos LaTeX. Sobre a arquitetura, foi desenvolvido um sistema web, chamado InProLa (Indexação e Processamento de LaTeX), capaz de processar e indexar trabalhos científicos escritos em LaTeX, bem como disponibilizar tais trabalhos de forma que possam ser pesquisados pelos seus conteúdos.

1.1. Justificativa

Com a contínua adesão atual de alguns professores orientadores na qual os trabalhos de conclusão do curso de Bacharelado em Sistemas de Informação do IFBA, campus Vitória da Conquista estarem sendo escritos em LaTeX, e por ainda não haver um local onde se concentraria a indexação das informações destes documentos de forma digital, este projeto de pesquisa idealiza e desenvolve uma plataforma online para que os docentes e discentes possam ter acesso a esse conteúdo de forma prática e rápida.

1.2. Problema

Após uma breve pesquisa, não foi possível encontrar soluções escaláveis que consigam realizar a indexação e processamento específico de trabalhos escritos em LaTex, tal qual este projeto visa fazer. Sobre isso, também não foram encontradas referências específicas a esforços similares que buscaram organizar a arquitetura deste tipo de software de forma plugável. Julgamos que a arquitetura plugável é necessária para que, ao longo da vida útil do software, novos módulos de processamento de textos LaTex sejam adicionados para disponibilizar novas informações ou visões sobre trabalhos acadêmicos de forma automatizada.

1.3. Objetivos

1.3.1. Geral

O principal objetivo deste trabalho é **criar um sistema web, chamado InProLa (Indexação e Processamento de Latex), utilizando uma arquitetura plugável, capaz de processar e indexar trabalhos científicos escritos em LaTex, bem como disponibilizar tais trabalhos de forma que possam ser pesquisados pelos seus conteúdos.**

1.3.2. Específicos

Associados ao objetivo geral, são identificados os específicos abaixo:

- Planejar e desenvolver uma arquitetura orientada a plugins que possibilite o gerenciamento destes com o mínimo de interrupções na execução do sistema;
- Criar uma aplicação web, chamada InProLa, utilizando a arquitetura plugável;
- Documentar o método através do qual os plugins devem ser desenvolvidos e integrados à aplicação web;
- Criar plugins para demonstrar o uso do método e da aplicação web.

2. Metodologia

2.1. Classificação da pesquisa

Esta é uma pesquisa aplicada predominantemente quantitativa, e foi utilizado o conhecimento da pesquisa aplicada para resolver essa demanda. Observa-se que ela é experimental. Também se fez necessário utilizar pesquisa de laboratório, pois todos os dados utilizados serão controlados pelo pesquisador e pelo professor orientador. Com isso, a fonte de informação foi provida por uma pesquisa documental.

2.2. Procedimentos Metodológicos

Foram seguidos os passos descritos na Figura 1. Tais atividades são explicadas de forma mais detalhada nas subseções seguintes.

2.2.1. Definir a estrutura da aplicação em geral

Este passo, destacado na Figura 1 pela atividade 1, teve como objetivo definir os padrões de projeto e de arquitetura a serem utilizados no desenvolvimento do sistema, visando garantir um melhor ambiente para manutenção e escalabilidade.

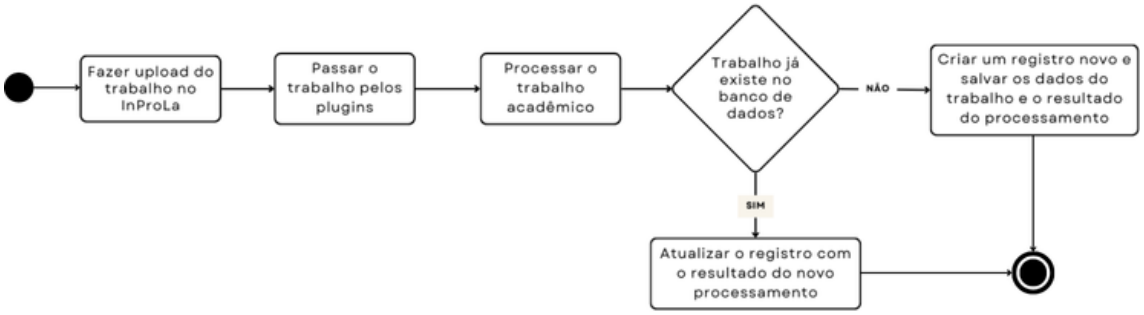

Figura 1. Fluxograma dos passos seguidos.

2.2.2. Desenvolver o back-end

Com a atividade 2, o objetivo se concentrou em ter um módulo responsável por sustentar o funcionamento dos plugins e seus processamentos, manter as regras de negócio, controlar usuários e acessar o banco de dados.

2.2.3. Desenvolver o primeiro plugin de processamento

Após desenvolver o back-end demos início à atividade 3, pois, para testar a funcionalidade de gerenciamento de plugins, era necessário que existisse ao menos um deles em mãos. Ele também foi utilizado como modelo para os próximos.

2.2.4. Desenvolver o front-end

É necessário ter um local que os usuários consigam acessar as funcionalidades do sistema. Para isso, a atividade 4 foi utilizada para desenvolver uma aplicação front-end responsável por toda a parte visual e interativa.

2.2.5. Avaliar a solução final

A atividade 5 resumiu-se em avaliar a implementação do InProLa utilizando os arquivos de TCCs que foram escritos em LaTeX e apresentados no curso, Bacharelado em Sistemas de Informação do IFBA, campus Vitória da Conquista, até o momento. Após isso, foram realizadas pesquisas sobre a base indexada, esperando que o sistema retornasse os dados selecionados.

2.2.6. Finalização do artigo

Por fim, na última etapa, foi realizada a atividade 5.1 de forma que os conhecimentos e contribuições deste trabalho fossem documentados durante a escrita deste artigo.

3. Fundamentos Teóricos e Técnicos

3.1. Arquitetura Orientada a Plugins

Observando a Figura 2 é possível compreender como explica os componentes de uma arquitetura baseada em plugins funciona [Gorbonosov et al. 2013][Cosentino 2023]:

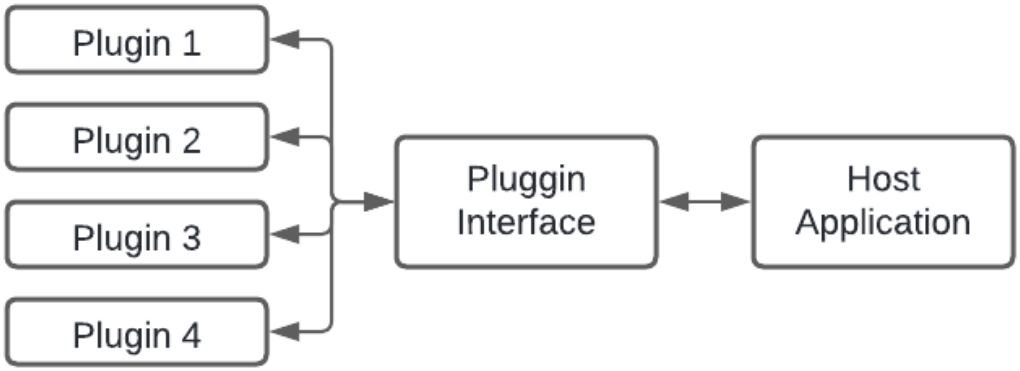

Figura 2. Uma arquitetura plugável

- A aplicação provedora ou *host application* é a responsável por fornecer os ambientes de utilização do usuário final;
- A interface de plugins ou *plugin interface* é a responsável por fornecer um padrão de conexão e chamadas a plugins. Esse padrão inclui métodos, propriedades, parâmetros e eventos que os plugins devem possuir ao serem desenvolvidos;
- O plugin é um módulo individual que a *host application* utiliza para executar passos de processamento. Cada plugin pode ser desenvolvido para uma funcionalidade específica, evitando acúmulos de responsabilidades .

3.2. LATEX

Segundo [Alves 2020] LaTeX é uma linguagem de marcação, com uma sintaxe parecida com HTML, voltada para a escrita de trabalhos acadêmicos e documentos. Essa semelhança se dá devido à utilização de tags junto com a escrita do texto. É possível observar algumas dessas tags na Figura 3: /title, que é utilizada para definir o título, e /author, onde se define o autor.

Ao invés de escrever o trabalho observando sua aparência final, com esta tecnologia, primeiro foca-se na estrutura e na qualidade do texto. Após isso, atribui-se a estilização. O resultado desta combinação de tags e texto pode ser observado na Figura 4.

3.3. Javascript

Javascript é uma linguagem de programação web criada pela Netscape em parceria com a Sun Microsystems, com o propósito de adicionar interatividade dentro de uma página web [Junior 2022]. Ela é constantemente utilizada como base no desenvolvimento de diversos *frameworks*, alguns voltados para a criação de sistemas web, tais como, por exemplo,

```
\title{Your Paper}
\author{You}

\begin{document}
\maketitle

\begin{abstract}
Your abstract.
\end{abstract}

\section{Introduction}

Your introduction goes here! Simply start writing your document and use the Recompile button to
view the updated PDF preview. Examples of commonly used commands and features are listed below,
to help you get started.

Once you're familiar with the editor, you can find various project setting in the Overleaf menu,
accessed via the button in the very top left of the editor. To view tutorials, user guides, and
further documentation, please visit our \href{https://www.overleaf.com/learn}{help library}, or
head to our plans page to \href{https://www.overleaf.com/user/subscription/plans}{choose your
plan}.

\section{Some examples to get started}

\subsection{How to create Sections and Subsections}
```

Figura 3. Exemplo de um artigo escrito em Latex antes de ser compilado

React.js, Angular, Vue.js, para aplicações mobile, através do React Native e Ionic e para a criação de APIs, como o Node.js.

Como consequência da sua ampla utilização, é possível observar na Figura 5 que JavaScript é a tecnologia mais popular entre os programadores, resultado de sua diversidade de *frameworks* que pode ser aplicada em diversas etapas do desenvolvimento de softwares.

4. Desenvolvendo o InProLa

O InProLa foi pensado para ser uma plataforma que utiliza plugins para realizar os processamentos dos projetos escritos em Latex e para que fosse possível ter um gerenciamento destes plugins, podendo adicionar novos, atualizar os já existentes e remover os desnecessários. Para tanto, na Figura 6 é possível observar as tecnologias que utilizamos no desenvolvimento.

Na primeira coluna da Figura 6 se encontram as tecnologias utilizadas no *backend*, como javascript, que também foi escolhido como linguagem para o *frontend*.

Para o desenvolvimento do *backend* foi escolhido o node.js, que é um ambiente de execução javascript [Node.js 2024], por ter uma comunidade de suporte relativamente grande e robusta, facilitando a busca e uso de bibliotecas para complementar a arquitetura do projeto e também por ser bastante utilizado atualmente [Kinsta 2023]. Também foi utilizado o Nest.js [NestJS 2024], que atuou juntamente com o Node.js, facilitando o gerenciamento dos *plugins*.

Para desenvolver o *frontend*, o *framework* utilizado foi o Angular [Angular 2024].

Your Paper

You

July 10, 2024

Abstract

Your abstract.

1 Introduction

Your introduction goes here! Simply start writing your document and use the Recompile button to view the updated PDF preview. Examples of commonly used commands and features are listed below, to help you get started.

Once you're familiar with the editor, you can find various project setting in the Overleaf menu, accessed via the button in the very top left of the editor. To view tutorials, user guides, and further documentation, please visit our help library, or head to our plans page to choose your plan.

Figura 4. Trecho do Artigo da figura 3 após ser compilado

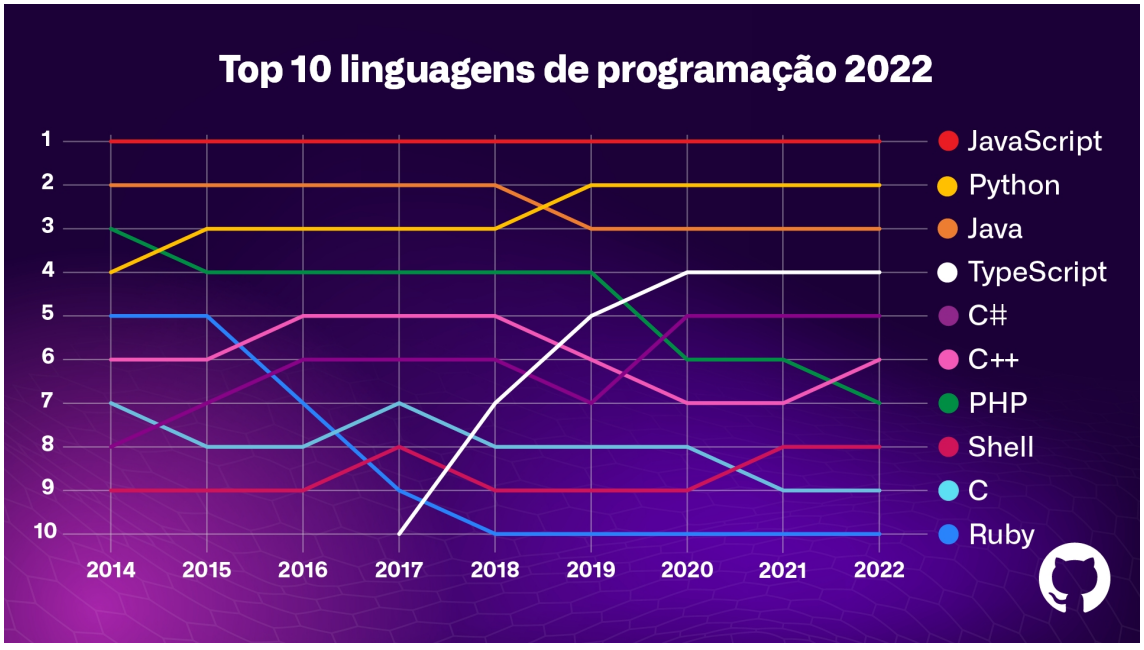

Figura 5. Linguagens de programação mais populares entre 2014 e 2022 segundo o GitHub [Infor Channel 2023]

Segundo [PET Sistemas de Informação, UFSM 2024] ele foi desenvolvido pelo Google e vem se tornando bastante popular por oferecer uma estruturação baseada em componentes, contribuindo para uma facilidade de manutenção de código.

Para armazenar os dados dos textos LaTeX após serem processados, utilizamos o MongoDB [MongoDB 2024], um banco de dados NoSQL orientado a documentos.

Na Figura 7, se encontra a estrutura de arquivos do *frontend* e do *backend* do InProLa. No backend a pasta src, chamada também de *source*, tem o objetivo de armazenar arquivos de código do projeto. A pasta libs é o local destinado para armazenar os plugins

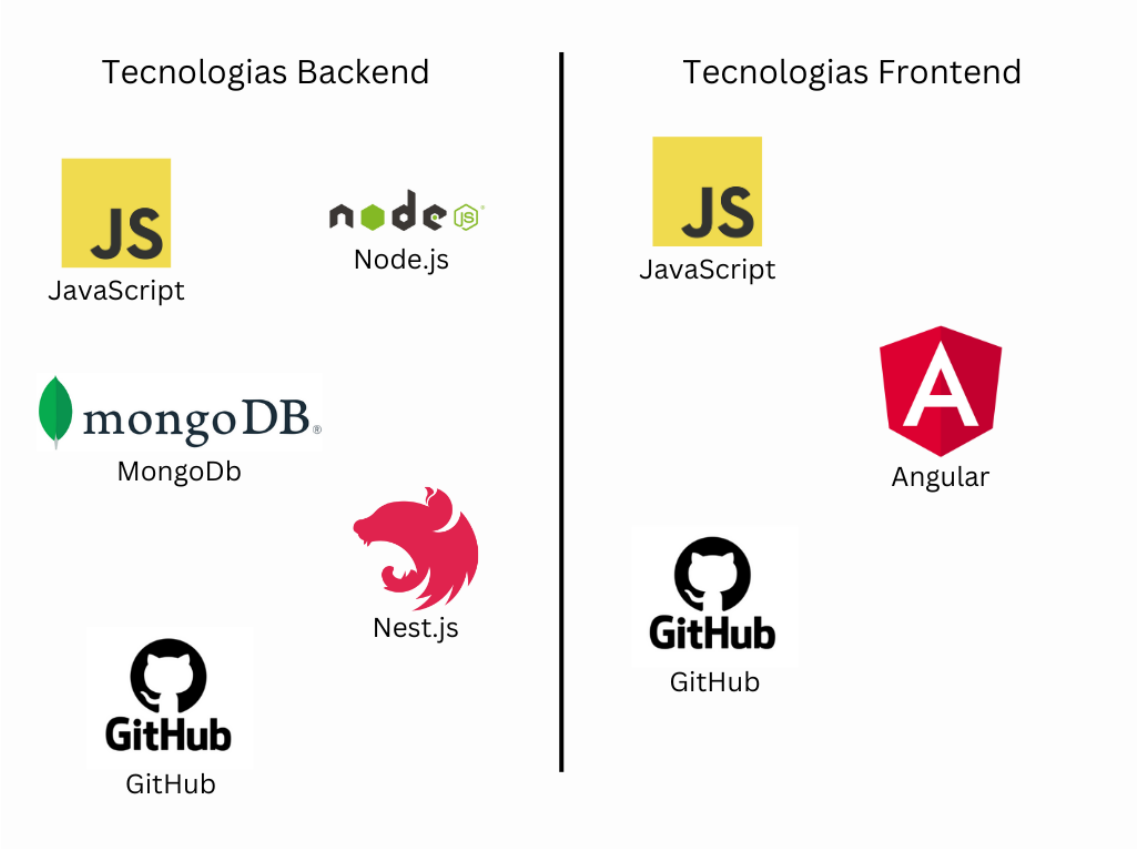

Figura 6. Tecnologias utilizadas no InProLa

desenvolvidos. A pasta *common* é para onde são destinados os arquivos compartilhados entre o projeto e os *plugins*, tais como classes e funções. Já no *frontend* temos a pasta *common*, que é utilizada para armazenar arquivos utilizados de forma global, como os *interceptors* e *pipes*. As demais pastas (no mesmo nível da *common*) são reservadas para os componentes funcionais do *frontend*. Cada pasta possui um nome relacionado à funcionalidade do seu componente respectivo. Na figura, por exemplo, a pasta *search.component* responde por funcionalidades relacionadas à pesquisa sobre os dados minerados dos textos Latex que foram indexados pelo InProLa.

É possível observar na Figura 8 o fluxo percorrido por um trabalho acadêmico escrito em LaTex ao ser processado pelo InProLa. O trabalho deve ser armazenado no Google Drive do InProLa para que este possa ser processado. É necessário executar um *endpoint* do *backend* para iniciar o processamento dos arquivos LaTex. O primeiro passo do sistema é realizar o download todos os trabalhos armazenados no Google Drive, para que se tenha uma cópia local a ser processada. Com todos os downloads concluídos, cada *plugin* deverá processar os trabalhos, extraindo informações específicas, que são armazenadas no banco de dados.

Observando a Figura 9 é possível compreender melhor o funcionamento de um plugin. Ele recebe como parâmetro um objeto chamado PluginProcessPayload, que é composto pelo trabalho acadêmico que irá ser processado e pelo *client* que irá fornecer acesso ao banco de dados. Para um funcionamento correto, obrigatoriamente, são execu-

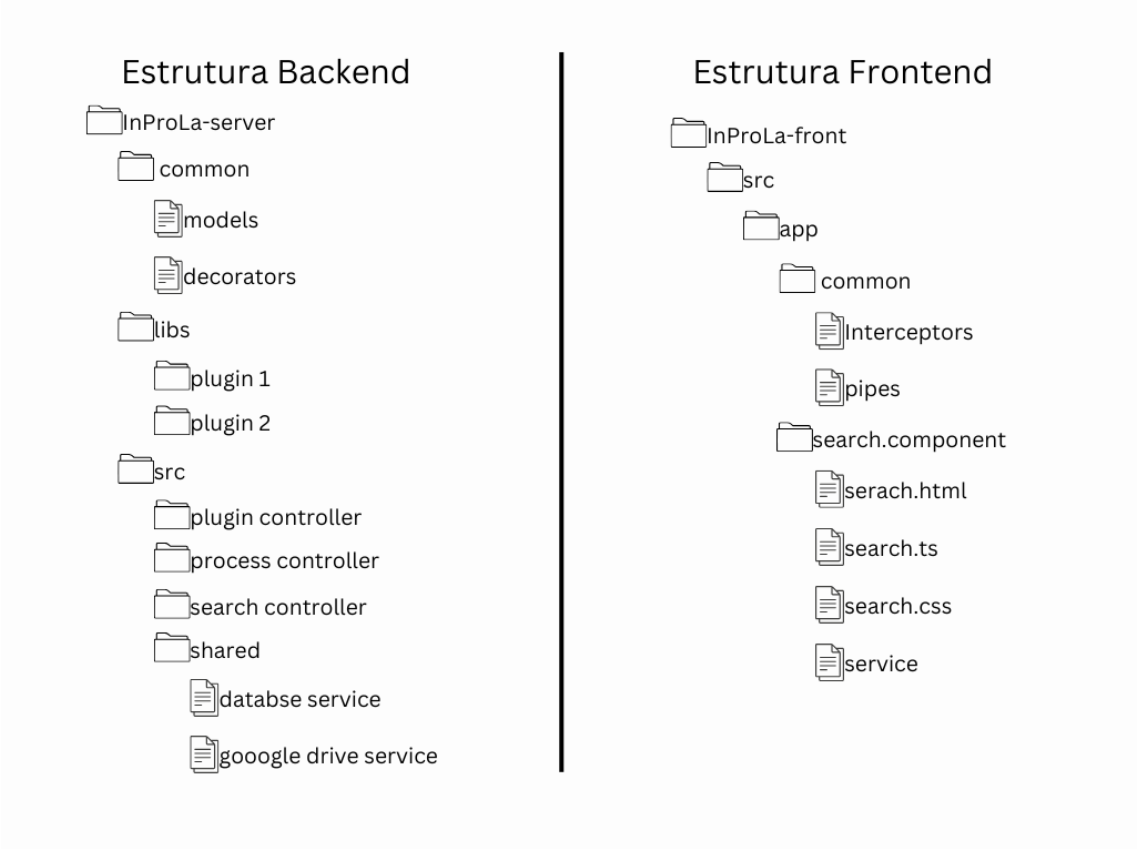

Figura 7. Estrutura dos projetos *backend* e *frontend*

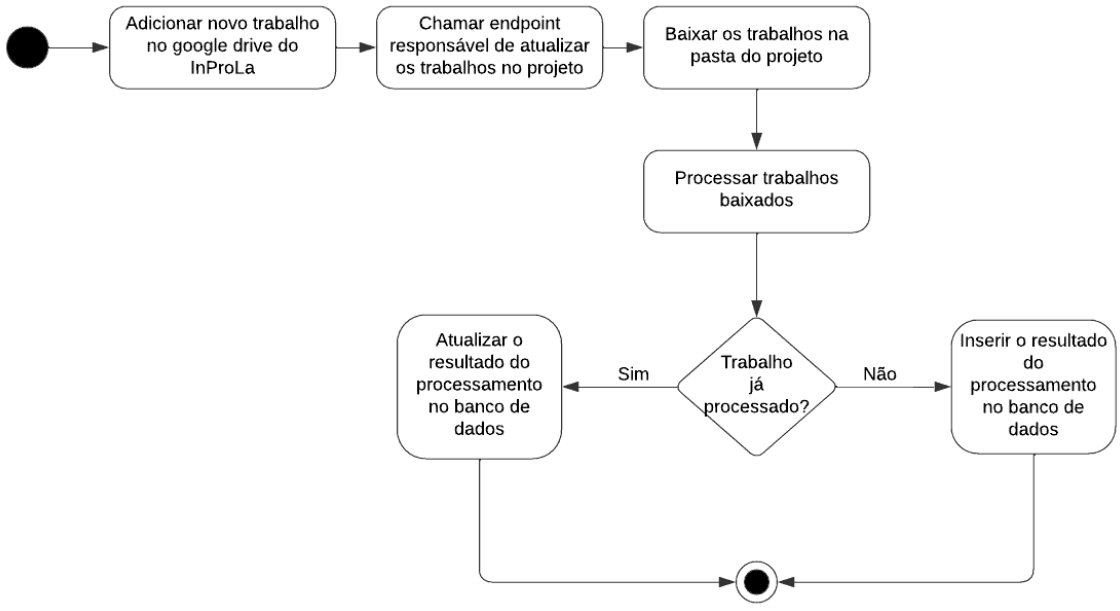

Figura 8. Fluxo de processamento dos trabalhos escritos em Latex

tadas 3 funções ou passos:

- *Process*, entre as linhas 10 e 29, tem como objetivo realizar o processamento do

texto do trabalho e armazenar o resultado no banco de dados. No exemplo, o objetivo é extrair o titulo do trabalho, contido na *tag*, *title*. Deve ocorrer neste passo a inserção do título no banco de dados, verificando se já existe um registro dessa informação. Caso exista, o título será atualizado. Caso não exista cria-se um novo registro com a informação;

- *Search*, entre as linhas 31 e 39, é responsável por verificar quais trabalhos armazenados no banco de dados possuem relação com os critérios de busca escolhidos pelo usuário no *frontend*. No exemplo da figura, o plugin apenas verifica se existe algum trabalho que foi processado possuía um texto específico na sua *tag*, *title*;
- *filterName*, entre as linhas 41 e 43, é responsável por dar nome ao filtro que é exibido no *frontend*. O frontend consulta esta funcionalidade para exibir o nome de cada plugin para os usuários.

Ao seguir um arquitetura orientada a plugins, todos os plugins adicionados ao inProLa devem seguir este mesmo perfil de codificação. Isto permite, por exemplo, que o inProLa consiga executar qualquer novo plugin da mesma forma que os anteriores e, portanto, evitar que seja necessário modificar seu código base [Cervantes and Charleston-Villalobos 2006].

Durante o desenvolvimento todo o código-fonte do InProLa foi mantido e gerenciado através do Github. Ele se encontra acessível pelo QR Code contido na Tabela 1.

Tabela 1. Links para acessar os repositórios do InProLa

Link	QR Code
Repositório do *backend* https://github.com/inProLa/inProLa-server	
Repositório do *frontend* https://github.com/inProLa/inProLa-front	

Compartilhar o código-fonte é importante para que outras soluções de software possam se basear no inProLa para processar formatos de texto, tais como o Latex, ou reusar sua arquitetura para solucionar outros problemas que possem fazer uso de plugins.

5. Trabalhos correlatos

Nesta seção examinamos alguns trabalhos baseados em esforços de mineração de informações a partir de documentos LaTeX.

```typescript
import { Injectable } from '@nestjs/common';
import { PluginServiceInterface } from '../../../common/models/plugin-service-interface';
import { PluginProcessPayload } from '../../../common/models/plugin-process-payload';
import { PluginSearchPayload } from '../../../common/models/plugin-search-payload';
import { PluginService } from '../../../common/pluginService.decorator';

@Injectable()
@PluginService()
export class TitleService implements PluginServiceInterface {
  async process(payload: PluginProcessPayload): Promise<void> {
    const texContent = payload.texFile.texText;
    const firstPart = texContent.split(String.raw`\title{`)[1];
    const titleText = firstPart.split('}')[0];

    await payload.dataBaseClient
      .db('plugins')
      .collection('academic_works')
      .updateOne(
        { fileId: payload.texFile.fileId },
        {
          $set: {
            fileId: payload.texFile.fileId,
            title: titleText,
          },
        },
        { upsert: true },
      )
      .catch((err) => console.error(err));
  }

  async search(payload: PluginSearchPayload): Promise<Array<any>> {
    const regex = new RegExp(payload.searchText, 'i');

    return await payload.dataBaseClient
      .db('plugins')
      .collection('academic_works')
      .find({ title: { $regex: regex } })
      .toArray();
  }

  get filterName(): string {
    return 'Título';
  }
}
```

Figura 9. Código de um exemplo de *plugin*

Um trabalho que tem como objetivo a extração de informações em trabalhos escritos em LaTeX e que utiliza aprendizado de máquina é o de [Duan et al. 2023], que apresenta o *framework* LaTeX Rainbow. Este *framework* destaca-se por sua capacidade de utilizar o código fonte LaTeX para gerar anotações semânticas detalhadas, preservando a estrutura hierárquica e a ordem de leitura dos documentos. A abordagem melhora a precisão das anotações em comparação com outros datasets tradicionais, ao evitar a perda de informações estruturais importantes durante o processo de extração. Contudo, o LaTeX Rainbow não indexa os projetos e nem os disponibiliza em uma plataforma de busca, se

limitando a apenas processar os trabalhos.

O trabalho de [Toksoz et al. 2024] introduz o PseudoSeer, um motor de busca especializado em recuperar artigos acadêmicos contendo pseudocódigo. Este sistema utiliza o Elasticsearch[1] para permitir buscas detalhadas através de diversos aspectos de um artigo, como título, resumo e seções de código LaTeX, oferecendo uma funcionalidade avançada de pesquisa por facetas e consultas de correspondência exata. Este projeto se alinha com as inovações introduzidas pelo PseudoSeer ao buscar aprimorar ainda mais a recuperação de informações em documentos acadêmicos, entretanto a arquitetura utilizada para desenvolver o PseudoSeer é estática, se limitando às funcionalidades realizadas no desenvolvimento do projeto. Devido ao InProLa ter sido desenvolvido utilizando arquitetura plugável, é possível desenvolver novas funcionalidades de processamento sem a necessidade de modificações no projeto base.

O EgoMath2 foi um projeto desenvolvido por [Mišutka and Galamboš 2011], que criaram um motor de busca especializado na recuperação de fórmulas matemáticas em grandes repositórios digitais, como a Wikipedia. O EgoMath2 utiliza fragmentos de LaTeX para identificar e processar notações matemáticas, convertendo-as em representações textuais que podem ser eficientemente indexadas e pesquisadas. Entretanto, ele se limita a apenas buscar fórmulas matemáticas e não cita sobre armazenar e disponibilizar estes trabalhos buscados. O InProLa, por ser orientado a plugins, possibilita desenvolver um recurso parecido com este e também outros para que diferentes processos possam ser executados e disponibilizando os artigos utilizados no processamento.

Após pesquisas, não foi possível encontrar um trabalho com todos os mesmos objetivos deste, de indexar e processar trabalhos em LaTeX. Também, de forma geral, foi identificado que não existem muitos trabalhos relacionados ao processamento de artigos escritos em LaTeX.

6. Demonstrando o uso do InProLa

Ao longo do desenvolvimento, para fins de testes, foram utilizados alguns Trabalhos de Conclusão de Curso (TCC) apresentados por discentes do curso de Bacharelado em Sistemas de Informação do IFBA, campus de Vitória da Conquista, durante o ano de 2023.

Através dos links na Tabela 2 é possível visualizar dois vídeos demonstrando usos possíveis do InProLa: processamento de textos Latex e visualização das informações processadas e um endereço que direciona para a aplicação do InProLa que está disponibilizada em nuvem.

7. Conclusões e trabalhos futuros

Este trabalho teve como objetivo desenvolver um sistema web escalável utilizando uma arquitetura plugável e criado utilizando JavaScript para que discentes e docentes possam ter acesso a artigos/TCCs que foram escritos em LaTeX. Foi possível desenvolver *plugins* para extrair informações do conteúdo e possibilitar a pesquisa pelos trabalhos.

Durante o desenvolvimento do sistema, o maior desafio foi encontrar uma maneira eficiente de gerenciar os *plugins* sem a necessidade de fazer alterações no código base da

[1]https://www.elastic.co/

Tabela 2. Links para acessar os vídeos de demonstração.

Link	QR Code
Vídeo demonstrando o upload de um novo trabalho https://encurtador.com.br/GP8ZG	
Vídeo demonstrando utilização da plataforma https://encurtador.com.br/Zus3x	
Endereço da aplicação https://inprola-2ba4218bd50b.herokuapp.com	

aplicação e sem precisar reiniciá-la para que novos plugins fossem adicionados. Adotamos o *framework*, Nest.js, e com ele foi possível implementar o recurso para gerenciar os *plugins*.

Cada plugin pode ser desenvolvido como uma biblioteca, seguindo um padrão de desenvolvimento, implementando funções pré-definidas em uma interface (conforme ilustrada pela Figura 9). Isto permite que novos plugins sejam adicionados à arquitetura já existente, possibilitando que outros desenvolvedores adicionem novas estratégias para extração e indexação das informações contidas em textos LaTeX.

No desenvolvimento deste trabalho, identificamos pontos que podem ser abordados em trabalhos futuros, tais como:

- Desenvolver plugins que tenham suporte a outros formatos de artigos além do formato da SBC, pois este, atualmente, é o único formato de texto Latex que o InProLa consegue processar. Neste momento, o trabalho se concentrou neste formato, porque ser o escolhido para a escrita dos trabalhos de conclusão do curso, Bacharelado em Sistemas de Informação do IFBA, campus Vitória da Conquista;
- Desenvolver plugins e/ou adapar os existentes para terem suporte de inteligência artificial. Exemplo: elaborar um plugin que possa realizar a descrição dos arquivos

de imagem contidos em textos Latex;
- Desenvolver um app mobile que tenha os mesmos recursos da página web, o que se encontra facilitado pelo fato da arquitetura do InProLa ter sido dividida entre frontend e backend com o processamento dos texto concentrados no backend. Um aplicativo móvel poderia fazer uso do mesmo backend;
- Avaliar a solução em um cenário real, tal como, por exemplo, no IFBA, campus Vitória da Conquista, cujo curso de Bacharelado em Sistemas de Informação já vem utilizando artigos como formato de registro de trabalhos de conclusão, com intuito de coletar *feedbacks* sobre a utilização do sistema. Com isso realizar melhorias na interface gráfica e na experiência de usuário.

Referências

Alonso, A. C. R. and Berti, L. F. (2019). Introdução à edição de textos em latex utilizando a plataforma overleaf. In *Semana da Matemática do INMA, 3ª, minicurso*, Mato Grosso do Sul. [Online; acessado em 16 mar. 2024].

Alves, N. (2020). Introdução ao latex. `https://alves-nickolas.github.io/pdf/LaTeX_XV_STO.pdf`. [Online; acessado em 09 jul. 2024].

Angular (2024). Angular overview. `https://angular.dev/overview`. Accessed: 2024-12-18.

Cervantes, H. and Charleston-Villalobos, S. (2006). Using a lightweight workflow engine in a plugin-based product line architecture. In *International Symposium on Component-Based Software Engineering*, pages 198–205. Springer.

Cosentino, N. (2023). Plugin architecture design pattern – a beginner's guide to modularity. `https://shorturl.at/qCu5Z`. [Online; acessado em 16 mar. 2024].

Duan, C., Tan, Z., and Bartsch, S. (2023). Latex rainbow: Universal latex to pdf document semantic & layout annotation framework. In *Proceedings of the Second Workshop on Information Extraction from Scientific Publications*, pages 56–67.

Gorbonosov, R., Kruk, G., Baggiolini, V., and Zerlauth, M. (2013). Plug-in based analysis framework for lhc post-mortem analysis. Technical report.

Infor Channel (2023). Github lista as linguagens de programação que mais crescem. Accessed: 2025-02-01.

Junior, A. (2022). *Comparação entre os principais frameworks Javascript de Front-end para o desenvolvimento de aplicações web*. Tcc (graduação) - engenharia de software, Universidade Federal do Ceará, Quixadá. [Online; acessado em 09 jul. 2024].

Kinsta (2023). O que é node.js e por que usá-lo? `https://kinsta.com/pt/base-de-conhecimento/o-que-e-node-js/`. [Online; acessado em 18 jul. 2024].

Mišutka, J. and Galamboš, L. (2011). System description: Egomath2 as a tool for mathematical searching on wikipedia.org. In Davenport, J. H., Farmer, W. M., Urban, J., and Rabe, F., editors, *Intelligent Computer Mathematics*, pages 307–309, Berlin, Heidelberg. Springer Berlin Heidelberg.

MongoDB (2024). Mongodb documentation. `https://www.mongodb.com/docs/`. Accessed: 2024-12-18.

NestJS (2024). Nestjs documentation. https://docs.nestjs.com/. Accessed: 2024-12-18.

Node.js (2024). About node.js. https://nodejs.org/en/about. Accessed: 2024-12-18.

Overleaf (2012). Sobre overleaf. https://pt.overleaf.com/about. [Online; acessado em 18 mar. 2024].

PET Sistemas de Informação, UFSM (2024). Angular: Transformando o desenvolvimento web. https://www.ufsm.br/pet/sistemas-de-informacao/2024/02/22/angular-transformando-o-desenvolvimento-web. Accessed: 2024-12-18.

Project, T. L. (2024). Latex – a document preparation system. https://www.latex-project.org/. [Online; acessador em 10 jul. 2024].

Toksoz, L., Srinath, M., Tan, G., and Giles, C. L. (2024). Pseudoseer: a search engine for pseudocode. *arXiv preprint arXiv:2411.12649*.

APÊNDICE

Os conceitos adquiridos durante o curso de Bacharelado em Sistemas de Informação (BSI) pelo Instituto Federal de Educação, Ciência e Tecnologia da Bahia contribuíram para a construção do projeto, desde o levantamento teórico até a aplicação prática. A Figura 10 demonstra todas as disciplinas que contemplam a ementa do curso de BSI e que se relacionam de forma direta e indireta.

As disciplinas de Linguagem de Programação I e II e Programação Web foram as que mais influenciaram no desenvolvimento deste trabalho. Linguagem de Programação contribuiu no desenvolvimento do pensamento lógico. Programação Web traz a utilização da programação para web, como a utilização de HTML, CSS e Javascript, que foi o foco deste trabalho.

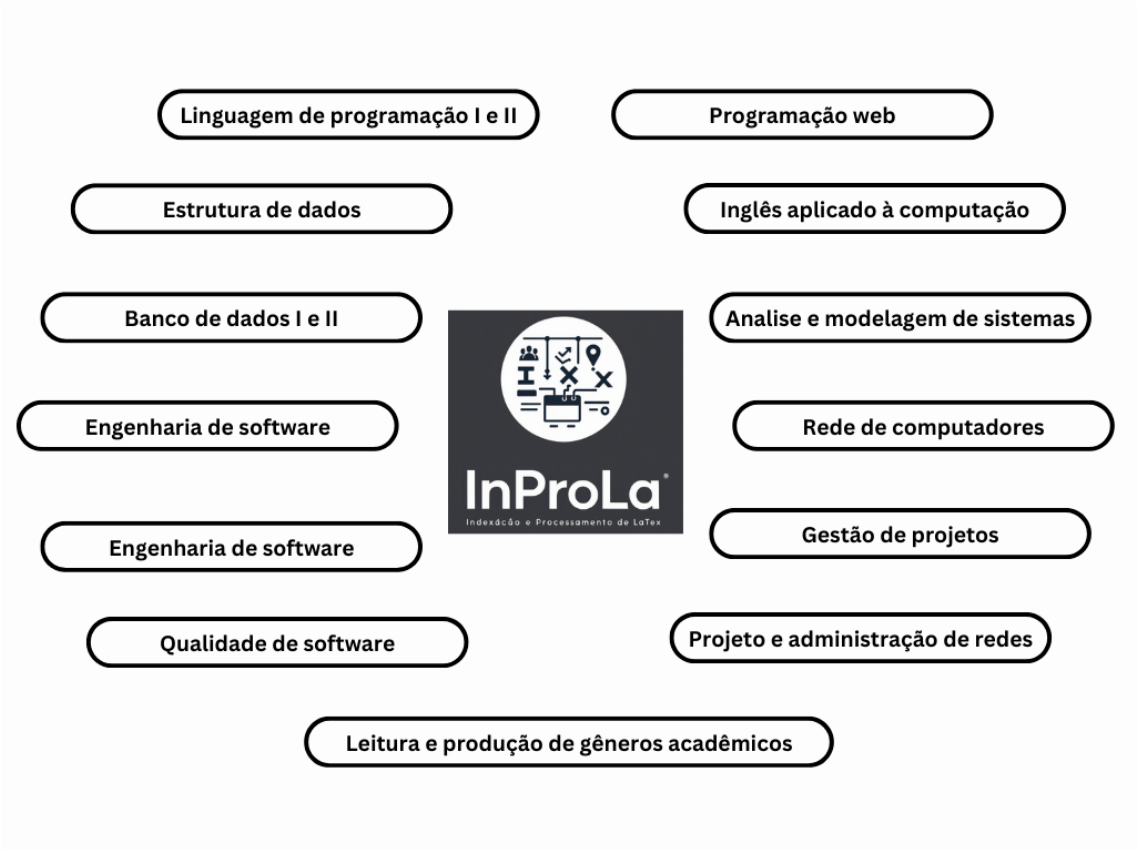

Figura 10. Disciplinas do Curso de BSI relacionadas à Realização do InProLa