Desenvolvimento de um sistema para extração e envio de Notícias do IFPA utilizando web scraping

Wendell dos Santos Morais
Instituto Federal de Educação, Ciência e Tecnologia do Pará
Rod. Ernesto Alcyioli, Km 3 - Estrada do Forte
Bairro Nova Colina - Altamira - PA - Brasil
+55 (93) 992447535
wendellmorais.dev@gmail.com

Vanessa Castro Rezende
Instituto Federal de Educação, Ciência e Tecnologia do Pará
Rod. Ernesto Alcyioli, Km 3 - Estrada do Forte
Bairro Nova Colina - Altamira - PA - Brasil
+55 (94) 8173-3262
vanessa.rezende@ifpa.edu.br

RESUMO

Pesquisas realizadas com a comunidade acadêmica do Instituto Federal do Pará (IFPA), identificaram gargalos na divulgação das informações publicadas no site institucional, destacando-se a baixa usabilidade e a pouca visibilidade do canal. Diante desse cenário, este trabalho propõe o desenvolvimento de uma aplicação para extração e envio automático de notícias oriundas do site oficial do IFPA, visando facilitar o acesso às informações institucionais, melhorando a comunicação interna. Para alcançar esse objetivo, a técnica de web scraping foi utilizada como mecanismo de extração de dados públicos da web. A aplicação foi inicialmente implementada baseada em uma arquitetura monolítica e, posteriormente migrada para uma arquitetura de microsserviços, a fim de realizar uma análise comparativa entre as duas abordagens. Essa comparação baseou-se em 12 cenários de teste distintos, variando a quantidade de notícias enviadas e de destinatários, simulando um aumento gradual do volume de dados processados. As métricas avaliadas incluíram o tempo total de envio, o consumo de CPU e o uso de memória. Os resultados indicaram que a arquitetura de microsserviços apresentou maior eficiência no tempo de envio, especialmente em cenários com alto volume de dados, enquanto a abordagem monolítica se destacou pelo menor consumo de recursos computacionais.

Palavras-chave

Arquitetura de software; Desenvolvimento de software; Extração de dados na web; Web Scraping.

ABSTRACT

Current research conducted with the academic community at the Instituto Federal do Pará (IFPA) – Campus Altamira revealed bottlenecks in disseminating information from the institutional website, highlighting usability issues and lack of channel promotion. In response, this work proposed developing an application for automatic extraction and delivery of news from IFPA's official website, aiming to facilitate access to institutional information and improve internal communication. To achieve this, Web Scraping techniques were used to optimize public data extraction from the web. The application was initially built using a monolithic architecture, later migrated to a microservices-based architecture to support a comparative study between both approaches. The comparison involved 12 test scenarios, varying the number of news items and recipients to simulate increasing data volumes. The analysis considered metrics such as news delivery time, CPU usage, and memory consumption. Results showed that the microservices-based application outperformed the monolithic version in delivery time across most scenarios, particularly under higher loads. However, the monolithic version demonstrated better efficiency in resource consumption. These findings suggest that while microservices may offer performance benefits under stress, monolithic architecture can be more resource-efficient, indicating trade-offs depending on application context and expected workload.

CCS Concepts

•Information systems → Email; Database management system engines; •Computing methodologies → Massively parallel and high-performance simulations; •Networks → *Network performance evaluation*; **•Computer systems organization →** *Dependable and fault-tolerant systems and networks*; **•Software and its engineering →** *Software system models; Inheritance*; **•General and reference →** Performance;

Keywords

Software Architecture; Software Development; Web Data Extraction; Web Scraping.

1. INTRODUÇÃO

Com o passar do tempo, a sociedade contemporânea sentiu a necessidade de alavancar a produção e o consumo de informação, a fim de promover seu desenvolvimento, evoluindo assim para a sociedade da informação [4]. Essa evolução foi impulsionada pelo advento da Web, associado ao crescimento da quantidade de meios de comunicação, que possibilitaram a produção e a propagação de um abundante volume de dados, resultando em uma sobrecarga de informações que provocou diversas mudanças de comportamento nos indivíduos da sociedade da informação. Apesar de ser o cerne desta nova sociedade, esse vasto volume de informações traz consigo uma série de desafios, especialmente quando se trata da sua disseminação [7].

Diante desse cenário, as Tecnologias da Informação (TICs) permitem ampliar o universo de disseminação das informações. No entanto, é relevante verificar em que medida há efetivamente a transmissão de informação e se ela atinge efetivamente o público alvo para o qual ela está direcionada. [15] afirmam que, existe um número maior de canais de informação à disposição das instituições e do público, mas não há garantia com relação ao alcance dessas informações.

Um fator relevante ao contexto é a aceitabilidade do público com relação ao canal utilizado na disseminação das informações, uma vez que a centralização de dados em plataformas com baixa usabilidade ou a ausência de estratégias eficientes para a divulgação de conteúdos também pode resultar no baixo alcance dessas informações ao público alvo, criando o potencial de acarretar uma desinformação em massa [25, 22].

No que tange a divulgação de informações do poder público, a qualidade do meio selecionado para a disponibilização das informações, assim como sua divulgação, tornam-se um ponto crucial para a transparência do conhecimento público [22]. [25] materializam essa afirmação ao expor os problemas desencadeados pela falta de transparência associada à complexidade de acesso às informações e levanta a importância dos meios de divulgação públicos atenderem os requisitos estabelecidos pela Lei nº 12.527, de 18 de Novembro de 2011, que regula o acesso à informação no Brasil.

Pesquisas conduzidas com a comunidade acadêmica do Instituto Federal do Pará (IFPA) no campus da cidade de Altamira revelaram um baixo acesso às informações concentradas no canal institucional, devido a diversos fatores, dentre os quais se destacam a complexidade na usabilidade e a ausência de divulgação do canal. Como forma de mitigar o impacto da desinformação entre os membros da comunidade acadêmica do campus de Altamira, os servidores da instituição optaram por adotar meios alternativos para disseminar as informações institucionais, como grupos de WhatsApp, embora isso possa implicar em um esforço manual considerável.

É relevante pontuar que a tarefa de compilar e enviar regularmente essas informações, além de onerosa, está sujeita a erros comuns relacionados à qualquer tipo de atividade essencialmente manual e repetitiva, demandando tempo e recursos consideráveis. Nesse cenário, as soluções de software surgem como uma alternativa eficiente para a automatização desse processo [21].

Um sistema construído para facilitar o processo de disseminação automática de informações pode ser uma ferramenta valiosa para aprimorar a eficiência, a precisão e a qualidade da divulgação de informações em diversos setores da sociedade. Esses sistemas têm a capacidade de processar grandes volumes de dados e disseminar informações de forma rápida e eficiente, sem exigir intervenção humana em tarefas repetitivas e demoradas [6]. Diante desse cenário, a comunidade científica tem realizado diversas pesquisas que demonstram o sucesso da aplicação de sistemas que realizam a automatização do processo de disseminação de informações aplicados aos mais variados tipos de contextos, especialmente quando se trata de informações disponibilizadas na Web [14].

2. FUNDAMENTAÇÃO TEÓRICA

2.1 Web Scraping

A comunicação dos usuários através da Internet ocorre majoritariamente através do acesso de páginas Web em um navegador, onde as informações são exibidas de forma simples a fim de facilitar o entendimento do usuário. No entanto, essa é só uma parte do potencial que a web proporciona [14]. Um exemplo é a possibilidade de compartilhamento ou exposição de dados entre aplicações web através do uso de APIs (*Application Programming Interface*).

Contudo, para consumir dados de uma API, é necessário que a sua criação envolva a disponibilização de acesso por parte do proprietário da aplicação, gerando uma dependência que pode representar um desafio quando se trata da coleta de dados desses sistemas. Nesses casos, a técnica de Web scraping tem se mostrado como uma alternativa eficiente de coleta de dados cuja principal vantagem é eliminar a necessidade da criação de APIs, uma vez que possibilita a extração de dados de páginas web de forma automatizada [19].

No Web Scraping, a extração automática de dados de páginas web é viabilizada através de requisições para os servidores web dessas páginas com o uso do protocolo de transferência de hipertexto HTTP (*Hypertext Transfer Protocol*) que, por sua vez, retorna os dados de forma desestruturada em um arquivo HTML (*Hypertext Markup Language*), como pode ser observado através do diagrama da Figura 1. Após a efetivação da extração, os dados obtidos podem ser estruturados, armazenados e utilizados para tomada de decisões ou expostos por outros meios de comunicação [19].

Figure 1: Fluxo de um sistema de Web Scraping.

[19] afirma que o web scraping abrange técnicas de programação, análise de dados e segurança da informação. Nesse contexto, diversas bibliotecas de software têm sido desenvolvidas com o objetivo de facilitar o uso do Web Scraping no desenvolvimento de software. Uma delas é o HAP (*Html Agility Pack*), ferramenta de código aberto criada pela empresa *ZZZ Projects* em 2007, utilizando a linguagem de programação C# [11]. Além do HAP, outras soluções foram desenvolvidas no ambiente da linguagem Python, como a biblioteca BeautifulSoup que foi criada em 2004 por Leonard Richardson que afirma que sua utilização pode economizar horas ou até dias de trabalho de um desenvolvedor [24].

Logo, o uso dessa técnica tem desempenhado um papel significativo em áreas como o comércio e a indústria, uma vez que uma aplicação que utiliza web scraping pode, por exemplo, extrair e comparar preços dos produtos contidos no website da concorrência, fornecendo subsídios relevantes para a tomada de decisões a partir de grandes volumes de dados de uma infinidade de websites, reduzindo drasticamente o tempo, a complexidade e a taxa de erro da extração de dados quando comparado com trabalhos manuais [10].

2.2 Arquiteturas de Software

O desenvolvimento de software envolve a construção de produtos cujo principal propósito é alcançar os objetivos de negócio de uma organização com foco no usuário. Nesse contexto, a arquitetura de software surge como uma "ponte" entre a abstração, que representa as finalidades do negócio, e a realidade, que é personificada pelo produto de software resultante desse processo de abstração. Para Bass et al. [2], a arquitetura de um software pode ser entendida como um conjunto de estruturas que permitem uma visão mais profunda sobre a solução a ser desenvolvida, buscando compreender seus elementos, suas características e, principalmente, a comunicação entre esses elementos.

Popularizado por Shaw e Garlan [26], o conceito de arquitetura de software foi criado com o objetivo de estruturar os grandes sistemas desenvolvidos na época, que tinham como característica uma alta complexidade de manutenção e evolução. Com o passar do tempo, a criação do conceito de arquiteturas de software proporcionou a

criação de diferentes estilos arquiteturais cujas características e funcionamento estão diretamente relacionados com as necessidade tanto de negócio como de desenvolvimento da aplicação em si, uma vez que têm o poder de proporcionar desacoplamento, flexibilidade e escalabilidade aos sistemas desenvolvidos [23].

Shaw e Garlan [26] definem um estilo arquitetural como "algo que define um vocabulário de componentes e tipos de conectores, juntamente com um conjunto de restrições sobre como eles podem ser combinados".

2.3 Sistemas Monolíticos

Um dos modelos arquiteturais mais utilizados na indústria de software são os sistemas monolíticos, visto que podem proporcionar maior simplicidade em atividades básicas do desenvolvimento de software, como o monitoramento, a realização de testes e a resolução de problemas [20].

Uma característica marcante dos softwares desenvolvidos com base em arquiteturas monolíticas é o fato de possuírem uma única unidade de implantação, composta basicamente pela UI (*User Interface* - Interface de Usuário), pela camada de acesso a dados e pela camada de regras de negócio que são implantadas em conjunto [20]. A utilização desse tipo de sistema ganha protagonismo na criação de sistemas pequenos, médios, ou protótipos que precisam ser implantados rapidamente, pois sua implantação é caracterizada por possuir baixa complexidade.

A Figura 2 representa o esquema básico de estruturação de um sistema monolítico. Note que todos os processos são centralizados em uma única aplicação que acopla todas as responsabilidades, geralmente limitando-se à uma única tecnologia de desenvolvimento [3]. A Engenharia de Software considera os sistemas monolíticos como uma aplicação sem dependência externa, sendo construída para funcionar como um sistema único e independente, sem a preocupação de se tornar um módulo para outra aplicação no futuro [16].

Figure 2: Arquitetura Monolítica.

Newman [20] categoriza os principais tipos de sistemas monolíticos através de três características:

1. **Monolítico com um único processo** é o tipo mais comum de sistema monolítico, baseado na implantação do código contida em um único processo, que pode ser estendido para várias instâncias a fim de aprimoramento de performance.
2. **Monolítico modular** é a variação em que o processo único é composto por módulos independentes que compõem a aplicação. Sua essência pode ser semelhante aos microsserviços, porém tendo uma topologia mais simples. No entanto, quando o banco de dados não é modularizado como o código, torna-se um desafio migrar seus módulos para uma arquitetura com sistemas autônomos.
3. **Monolítico distribuído** possui um sistema distribuído entre serviços, mas que são implantados em conjunto. Newman [20] afirma que um sistema monolítico distribuído tem todas as desvantagens de um sistema distribuído e as desvantagens de um sistema monolítico, devido esse fato, sua utilização deve observar diversos fatores que nem sempre farão a sua utilização se justificar em detrimento à outras abordagens.

A abordagem monolítica pode ser uma opção benéfica para o início do projeto, pois sua implementação é menos complexa quando comparada aos sistemas baseados exclusivamente em microsserviços [20]. Outro ponto relevante é que as arquiteturas monolíticas não são tão facilmente escaláveis, isso pode ser tornar um problema futuro, uma vez que espera-se que o software acompanhe o crescimento da empresa, e por serem sistemas rígidos, a arquitetura monolítica pode chegar a um ponto em sua manutenção seja altamente complexa e de difícil adaptação a novas necessidades [28].

2.4 Microsserviços

Com o avanço da tecnologia nos últimos anos, o setor de desenvolvimento de software presenciou uma rápida mudança na arquitetura de sistemas. Empresas como Netflix, Twitter e Spotify migraram a base de suas aplicações baseadas em uma arquitetura monolítica para a arquitetura de microsserviços com o objetivo de resolver problemas de escalabilidade, alcançar a modularidade, adotar de novas tecnologias e facilitar o desenvolvimento de novas funcionalidades [8].

Segundo Newman [20], os microsserviços possuem foco em fornecer diversas possibilidades para resolução de problemas encontrados no desenvolvimento de software. É um estilo arquitetural que propõe a modularização e o desacoplamento da solução, separando-a em pequenas aplicações autônomas e de responsabilidades únicas. Sendo fortemente inspirado nas concepções do *Domain-Driven Design* (DDD), que define o domínio do software como o conjunto de ideias e processos de negócio inseridos no contexto do software construído [23]. Esses domínios são tidos como referência para a divisão das aplicações que irão compor os microsserviços [20].

Sua divisão é baseada em como o modelo de negócio é estruturado no mundo real, em que cada microsserviço mantém-se focado em resolver a lógica do domínio delimitado, com o objetivo de garantir o desacoplamento de suas regras de negócio, tecnologias e fontes de dados dos demais serviços. Com isso, diferentemente do modelo monolítico, as aplicações são executadas em processos distintos e realizam a troca de informações pela rede, comumente utilizando padrões como REST, gRPC ou comunicação assíncrona por meio de Mensageria [20]. A Figura 3 apresenta um exemplo de aplicação genérica desenvolvida com a arquitetura de microsserviços.

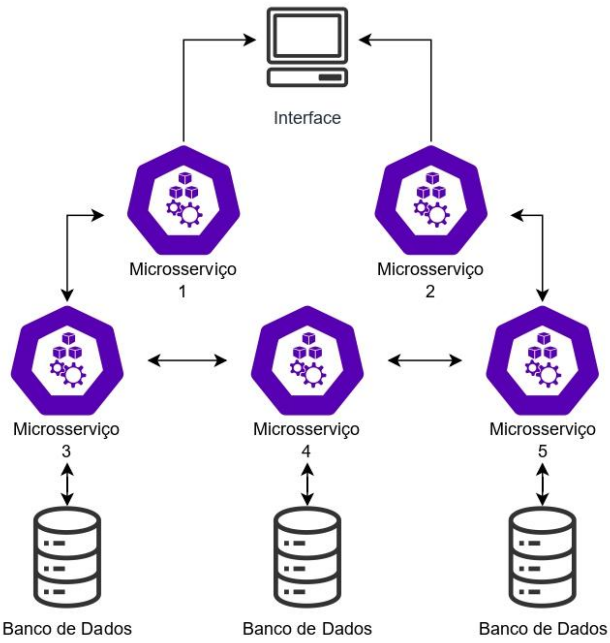

Figure 3: Arquitetura de Microsserviços.

Essa divisão proporciona um dos maiores benefícios da arquitetura em relação aos modelos tradicionais, pois quando os limites de cada microsserviço são bem delimitados, diminui-se o acoplamento, aumenta-se a coesão e a flexibilidade do software [20]. Além disso, proporciona uma maior manutenibilidade do sistema, pois os domínios do negócio estarão separados em bases de código menores e mais especialistas. Essa vantagem pode se tornar um desafio em sistemas monolíticos, por terem todo o funcionamento da aplicação unificado em um único local.

No entanto, Newman [20] ressalta que, apesar dos inúmeros benefícios oferecidos, a definição dos limites entre os serviços deve ser feita com cautela, pois ela pode ter um impacto significativo no sistema. Ademais, sendo uma arquitetura distribuída, a arquitetura de microsserviços carrega consigo todas as complexidades desse tipo de software.

3. MATERIAIS E MÉTODOS

3.1 Processo de construção da solução

Visando alcançar o objetivo de desenvolver uma solução de extração e envio automático de notícias no IFPA, foram desenvolvidas subatividades categorizadas em quatro fases discriminadas através do fluxo geral descrito na Figura 4.

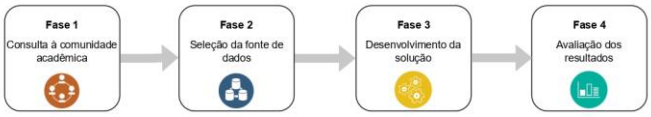

Figure 4: Fluxo de construção da solução de software.

A primeira fase envolveu a consulta à comunidade acadêmica sobre a disseminação de notícias do IFPA, realizada via formulário, a fim de avaliar a viabilidade da solução que seria proposta nessa pesquisa. Após essa avaliação, foi realizado um levantamento das principais fontes de dados que seriam utilizadas como insumo para a aplicação proposta, que foram os portais de notícias do IFPA Geral e IFPA Altamira. Em seguida, iniciou-se o desenvolvimento da aplicação que realiza a extração das notícias e as envia por e-mail. Por fim, a fase de avaliação dos resultados objetivou avaliar o desempenho da solução de software proposta, além de realizar uma análise comparativa de uma versão dessa aplicação baseada em uma arquitetura monolítica em detrimento a uma versão inspirada em uma arquitetura de microsserviços.

3.1.1 Consulta a Comunidade Acadêmica

Visando avaliar a viabilidade da solução proposta pelo presente trabalho, foi desenvolvida uma pesquisa com o objetivo de entender as necessidades da comunidade acadêmica em relação à disseminação de informações dentro do IFPA. A consulta consistiu em um formulário construído através da ferramenta *Google Forms*, que foi enviado aos alunos, técnicos e professores do instituto. O formulário obteve um total de 41 participantes, dos quais 75,6% foram alunos, 22% professores e 2,4% foram técnicos administrativos. É relevante pontuar que cada participante estava limitado a apenas uma resposta, pois o objetivo foi obter informações com maior confiabilidade e integridade, reduzindo a incidência de possíveis ruídos.

Inicialmente, foi questionada a frequência com que o entrevistado acessa os portais de notícias do IFPA Campus Altamira (https://altamira.ifpa.edu.br/) e o site de notícias gerais do instituto (https://ifpa.edu.br). A pesquisa revelou que 43,9% raramente acessa o portal de Altamira, 9,8% acessa mensalmente, 26,8% semanalmente, 9,8% diariamente e 9,8% afirma nunca ter entrado no site. Tendência similar foi constatada com relação ao portal geral de notícias, onde 39% dos entrevistados alega acessar o site raramente, 17,1% mensalmente, 24,4% semanalmente, 9,8% diariamente, enquanto 9,8% afirma nunca ter acessado o portal. Essa pesquisa revelou que, dentre os entrevistados, sendo a maioria alunos, existe um baixo percentual de acesso periódico.

Após os questionamentos sobre a frequência de acesso, foram exibidas perguntas sobre a usabilidade dos sites. Visando maior veracidade das respostas, os questionamentos acerca de facilidade de acesso e utilização não foram exibidos aos entrevistados que afirmaram nunca ter acessado o respectivo portal, visto que não teriam experiência de usabilidade para poder avaliar os portais de notícias. Após essa etapa da entrevista, 86,1% dos entrevistados afirmaram ter uma experiência neutra ou negativa acessando o portal do campus Altamira, enquanto no site geral de notícias o percentual para o mesmo questionamento correspondeu 77,2% dos entrevistados.

Nesse contexto, a maior parte dos entrevistados, 61%, acredita já ter perdido oportunidades de auxílio ou participação em eventos na instituição devido à não utilização frequente do site do IFPA. Além disso, 75,6% acredita que uma solução de software que envia notícias do instituto por e-mail seria útil para a comunidade acadêmica.

Figure 5: Gráfico gerado pelas respostas ao formulário.

Por fim, os resultados da pesquisa indicaram a necessidade e a importância de se adotar estratégias alternativas de comunicação,

uma vez que estas poderiam ser capazes de atender às necessidades da comunidade acadêmica. Tal necessidade serviu como base para demonstrar a relevância da solução de software para a disseminação de notícias proposta no presente trabalho.

3.1.2 Seleção das fontes de informação

Segundo Baggio et al. [1], identificar as fontes de informação é o primeiro passo para prestar um serviço de informação aos usuários, sendo elas parte integral desse serviço.

Nesse sentido, as fontes de informação eleitas para alimentar a solução de software proposta foram os sites institucionais do IFPA Altamira e IFPA Geral, ambos portais públicos fornecidos pelo IFPA para divulgação de notícias oficiais. O site IFPA Altamira, foi desenvolvido utilizando o Sistema de Gerenciamento de Conteúdo (*CMS - Content Management System*) de código aberto *Joomla* (https://www.joomla.org/) e o objetivo de sua criação foi atuar como um portal abrangente para alunos, professores, técnicos administrativos e a comunidade em geral, proporcionando acesso a uma variedade de conteúdos, incluindo avisos importantes, notícias atualizadas sobre eventos acadêmicos e administrativos, além de oportunidades de auxílio. O foco principal do conteúdo é em torno dos acontecimentos no IFPA Campus Altamira, tornando-o uma fonte confiável de informações para todos os interessados.

Por sua vez, o site IFPA Geral abrange uma esfera mais ampla, tratando das necessidades informativas de toda a rede do Instituto Federal do Pará. Também desenvolvido utilizando o CMS Joomla, esse portal é uma fonte de informações abrangente, fornecendo notícias e informações que afetam a comunidade acadêmica em geral, incluindo políticas educacionais, inovações pedagógicas e eventos em toda a rede do IFPA.

Um aspecto identificado durante a fase de planejamento do presente trabalho, é a ausência de disponibilização dessas informações do IFPA por meio de bases de dados acessíveis ou APIs (*Application Programming Interface* - Interface de Programação de Aplicações). Tal lacuna impõe desafios com relação à extração de dados, uma vez que as informações estão disponíveis apenas por meio de visualização direta nos sites. Por esse motivo, adotou-se a técnica de Web Scraping como método de coleta de dados, permitindo a extração das informações necessárias diretamente das páginas web dos sites do IFPA Altamira e IFPA Geral.

É relevante pontuar que, ambas as fontes de informação, sites IFPA Altamira e IFPA Geral, possuem uma infinidade de conteúdos relevantes a instituição, no entanto, o presente trabalho envolverá a extração e disseminação de dados apenas da guia ”Últimas Notícias” de cada um desses portais de divulgação, a fim de fornecer dados atualizados sobre os acontecimentos da comunidade acadêmica assim que disponibilizados na plataforma oficial.

3.1.3 Desenvolvimento da Solução de Software: Notícias IFPA

O desenvolvimento da solução de software intitulada como “Notícias IFPA” foi projetada para atender às demandas específicas de coleta e envio de notícias do IFPA. Através da Figura 6 é possível observar as etapas que nortearam o processo de implementação dessa solução que partiu da criação de uma página para a captação dos endereços de e-mail dos usuários interessados em utilizar o serviço, seguindo para a implementação da solução de extração de dados utilizando a técnica de Web Scraping, assim como o envio automático de e-mails.

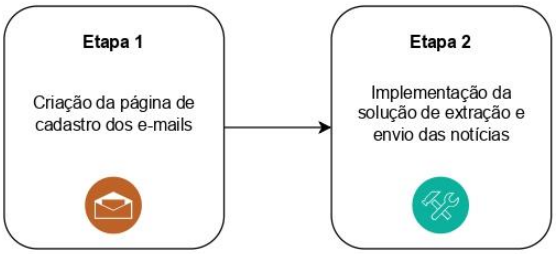

Figure 6: Etapas de desenvolvimento da solução de software.

Vale destacar que o processo de implementação da solução foi concebido seguindo os princípios da Arquitetura Limpa, cujo cerne é determinar um código que possua uma clara distinção entre a lógica de negócio, o acesso a dados e a conexão com interfaces externas. As camadas do software são projetadas para serem independentes de *frameworks*, UI (*User Interface - Interface de Usuário*), banco de dados ou qualquer agente externo, permitindo uma maior flexibilidade e adaptabilidade à mudanças ou novas tecnologias [18].

O desenvolvimento da aplicação “Notícias IFPA” iniciou-se com uma abordagem monolítica, na qual todas as funcionalidades do sistema - desde a UI (*User Interface* - Interface de Usuário) até a lógica de acesso a dados, foram integradas em um único bloco de software. Esta estratégia foi inicialmente adotada com o intuito de simplificar o processo de desenvolvimento e implantação do sistema.

No entanto, após o término do desenvolvimento da aplicação monolítica, houve uma transição para uma aplicação baseada em uma arquitetura mais descentralizada, surgindo assim uma nova versão do “Notícias IFPA” utilizando uma arquitetura baseada em microsserviços. Tal mudança permitiu que diferentes partes do sistema fossem desenvolvidas e implantadas independentemente. É pertinente pontuar que, assim como a versão monolítica, o software construído na arquitetura de microsserviços também seguiu os princípios da Arquitetura Limpa.

As seguintes seções abordarão os detalhes que nortearam o desenvolvimento das duas aplicações, denominadas nesse trabalho como monolítica e microsserviços, assim como o desenvolvimento da página para a captação dos endereços de e-mail que foi reutilizada nas duas aplicações e a utilização da ferramenta de versionamento para o controle e a organização do processo de implementação das aplicações.

3.1.4 Página de Captação dos Endereços de E-mail

A construção da Página de Captação dos Endereços de e-mail surgiu pela necessidade de possuir uma base de e-mails para recebimento das notícias extraídas pelo Notícias IFPA. Em geral, essa página foi desenvolvida sob a premissa de oferecer uma experiência simples e intuitiva para os usuários e, ainda assim, cumprir o seu propósito.

A fim de cumprir o objetivo proposto, foram desenvolvidas duas telas. A primeira é a ”Página inicial” e a segunda é a ”Página de Cadastro”. A Figura 7 apresenta a página inicial do “Notícias IFPA”, onde é apresentado o objetivo do projeto, bem como suas funcionalidades e as vantagens de se inscrever. Ao clicar no botão ”QUERO ME INSCREVER”, o usuário é redirecionado para o formulário de cadastro do endereço de e-mail (Figura 8).

Figure 7: Página inicial do site Notícias IFPA.

A Figura Figura 8 apresenta o formulário de cadastro endereços de e-mail, onde o usuário pode selecionar as fontes de notícias que irá receber e cadastrar o seu endereço eletrônico. Tais ações podem ser realizadas através dos campos: “IFPA Altamira” e “IFPA Geral”, indicam de quais fontes o usuário deseja receber as notícias. Note que, pelo menos um dos campos precisa ser selecionado; ”Seu principal e-mail”, é onde usuário deverá preencher com um endereço de correio eletrônico válido, uma vez que o sistema realiza essa validação.

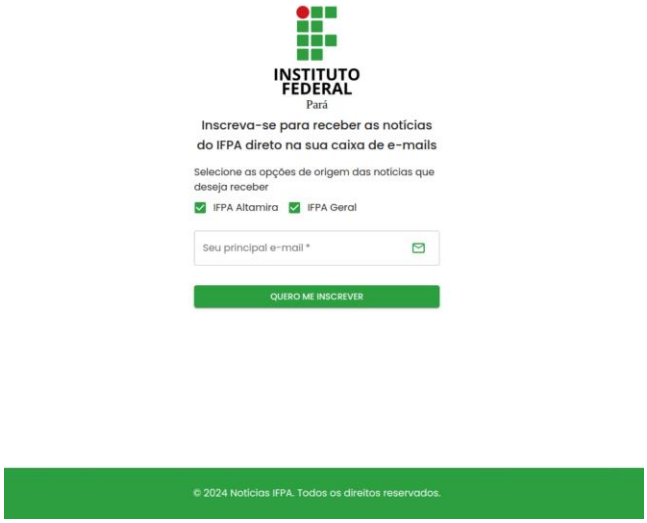

Figure 8: Página de cadastro dos e-mails.

A construção da página se deu através do uso da linguagem de programação Typescript aliada à biblioteca front-end React, tendo utilizado o Git para versionamento do código fonte.

3.1.5 Implementação da aplicação de extração de dados em uma arquitetura Monolítica

Em geral, o processo de extração de notícias, baseado em uma arquitetura monolítica, fundamentado na técnica de Web Scraping, demandou uma análise técnica dos elementos do site alvo, visando a implementação de um mecanismo para identificação do conteúdo inserido no portal. A análise envolveu a inspeção do código HTML da página para identificar os seus elementos e a hierarquia de sua estrutura, bem como as classes identificadoras de cada elemento na página. Com isso, foi possível localizar e extrair somente o conteúdo da notícia a ser compartilhada.

A Figura 9 apresenta o esquema básico de estruturação do que sistema se divide em três principais funcionalidades internas:

- **Extração de notícias:** Utiliza a técnica de Web Scraping juntamente com a biblioteca Html Agility Pack para monitorar o portal do IFPA, identificando novas publicações a serem enviadas;
- **Gerenciamento de e-mails:** Atua como um repositório para os endereços de e-mail cadastrados por meio da interface da página “Notícias IFPA”. Posteriormente, esses endereços são armazenados e disponibilizados para o processo de envio de notícias;
- **Envio de e-mails:** Estabelece uma conexão com um servidor SMTP (*Simple Mail Transfer Protocol*) e realiza o envio automático das notícias para os endereços de e-mail dos destinatários previamente cadastrados.

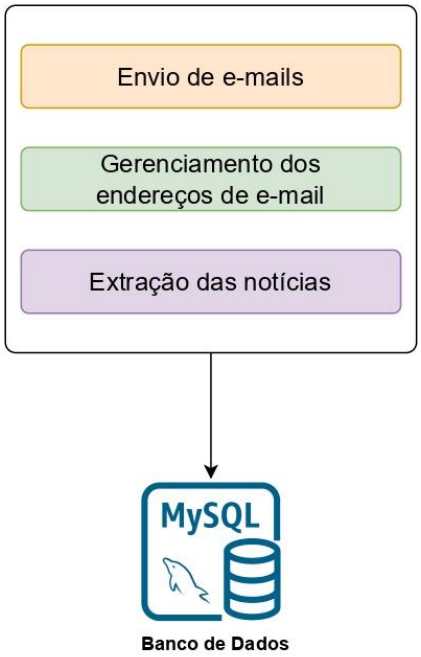

Figure 9: Arquitetura geral da aplicação monolítica.

Por sua vez, diversas ferramentas foram utilizadas no desenvolvimento da aplicação, como a linguagem de programação C# durante o processo de implementação da solução, a plataforma .NET e um banco de dados em MySQL para gerenciamento das notícias extraídas através dos endereços de e-mail.

3.1.6 Implementação da aplicação de extração de dados utilizando Microsserviços

A migração de um sistema monolítico para um sistema baseado na arquitetura de microsserviços tem o potencial de ampliar a escalabilidade e modularidade do software em casos com maiores volumes de dados [27, 17]. Partindo desse pressuposto, surgiu a motivação de analisar os efeitos gerados pela mudança arquitetural da aplicação de extração e disseminação de notícias, através da comparação entre as duas aplicações desenvolvidas, monolítica e microsserviços, possibilitando a documentação desse experimento, a fim de gerar material de estudo para estudantes, desenvolvedores e arquitetos de software.

Devido às características intrínsecas de um sistema baseado em microsserviços, foi possível realizar a segregação das principais funcionalidades do sistema, que outrora estavam agrupadas em um único módulo, em serviços distintos, independentes e focados em uma única tarefa. A Figura 10 ilustra essa decomposição da aplicação em três microsserviços, sendo eles o Serviço de Gerenciamento de E-mails (SGE), o Serviço de Extração de Notícias (SEN) e o Serviço de Envio de E-mails (SEE).

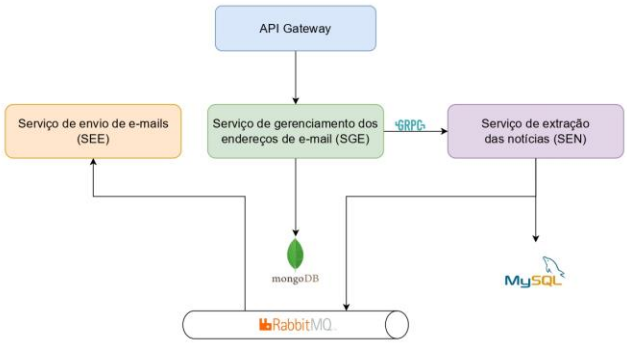

Figure 10: Arquitetura geral da aplicação de microsserviços.

Visando total independência entre os serviços, a base de dados também foi fragmentada em duas partes: um banco de dados MongoDB dedicado ao armazenamento dos endereços de e-mail, que se conecta ao SGE; e um banco de dados MySQL com o intuito de armazenar as notícias que, por sua vez, estará conectado ao SEN.

Adicionalmente, a adoção da nova arquitetura introduziu uma API Gateway, que funciona como um ponto único de entrada para todas as requisições externas. Essa centralização facilita a comunicação do software com sistemas externos, auxiliando no processo de autenticação, autorização e distribuição de carga das requisições [20].

Nesse contexto, devido a separação entre as unidades de funcionamento do software, torna-se necessária a implantação de uma camada de comunicação entre os microsserviços. Para a comunicação entre SEN e SGE, foi utilizado o gRPC, que proporciona alta performance e baixa latência na comunicação no contexto de microsserviços [13]. A comunicação entre o SEN e o SEE, foi implementada utilizando um sistema de comunicação assíncrona com o RabbitMQ, que assegura uma maior tolerância à falhas durante a comunicação [12]. Dessa forma, o SEN extrai uma nova notícia, requisita lista de endereços de e-mail para o SGE via gRPC. Com o conteúdo da notícia e os endereços de e-mail em mãos, ele publica essas informações em uma fila do RabbitMQ. O SEE, por sua vez, consome estas mensagens e realiza o envio dos e-mails.

3.2 Critérios de Avaliação

A presente seção oferece uma descrição detalhada acerca dos critérios adotados neste estudo para a comparação entre as duas aplicações de extração desenvolvidas.

3.2.1 Tempo de envio das notícias

O tempo de envio das notícias diz respeito ao intervalo de tempo, em segundos, entre a execução da aplicação até o envio da última notícia do teste. Essa métrica indica a eficiência dos dois sistemas em diferentes cenários, podendo evidenciar também a escalabilidade do sistema, visto que é possível observar o seu comportamento com maior carga de dados a serem processados.

$$T_{envio} = T_{final} - T_{inicial}$$

Onde:

- T_{envio} é o tempo total de envio das notícias.
- T_{final} é o tempo em que a última notícia é enviada.
- $T_{inicial}$ é o tempo em que a aplicação começa a ser executada

3.2.2 Consumo de CPU

O consumo médio de CPU (*Central Processing Unit*) é uma métrica utilizada para determinar o uso de recursos que uma aplicação necessita para cada cenário, influenciando diretamente nos custos de implantação do sistema. Isso é particularmente relevante em ambientes de nuvem, onde o consumo de CPU pode representar a maior parte dos custos operacionais para manter o sistema online. Sendo assim, uma aplicação com menor consumo de CPU resulta em uma economia nos custos de implantação do sistema [3].

O percentual total de consumo de CPU é obtido através da seguinte equação:

$$x = \frac{\sum_{t=1}^{T} CPU\ (t)}{T} (1)$$

Onde:

- x é o valor final de consumo.
- Σ é o somatório dos valores do consumo de CPU medidos em cada segundo do intervalo de tempo.
- t é a variável de Índice que representa cada segundo dentro do intervalo de tempo.
- T é o número total de segundos no intervalo de tempo considerado.
- CPU(t) é o percentual de consumo de CPU no segundo t.

3.2.3 Consumo de memória RAM

O consumo de memória RAM (*Random Access Memory*) pode ser utilizado como uma métrica para avaliar o gerenciamento de recursos e a estabilidade de uma aplicação. Através de sua análise, é possível medir a estabilidade do software em um ambiente com recursos de hardware limitados. Além disso, o consumo de memória pode evidenciar possíveis vazamentos de memória do sistema em situações de alta carga de trabalho, o que pode levar à interrupção da execução da aplicação [5].

De forma similar a medida do consumo de CPU, o cálculo para obtenção dos valores de percentual do consumo de memória RAM é uma média de sua utilização em todo o tempo de execução da aplicação. Portanto, a seguinte equação é utilizada para obter o valor final de consumo:

$$x = \frac{\sum_{t=1}^{T} RAM\ (t)}{T} (2)$$

Onde:

- x é o valor final de consumo de memória RAM.
- Σ é o somatório dos valores do consumo de memória RAM em cada segundo do intervalo de tempo.
- t é a variável de Índice que representa cada segundo dentro do intervalo de tempo.
- T é o número total de segundos no intervalo de tempo considerado.
- RAM(t) é o percentual de consumo de memória RAM no segundo t.

4. RESULTADOS

A presente seção apresentará os resultados obtidos a partir da aplicação da metodologia para o desenvolvimento das aplicações

propostas neste trabalho, dando ênfase na comparação entre as duas arquiteturas em termos de tempo de envio, processamento e memória.

Ressalta-se que todos os testes foram realizados em uma máquina com processador Intel Core Xeon E5-2640 de 2,5 GHz (3 GHz) com limitação de 1 core de processamento e 1GB de memória RAM DDR3. A estratégia utilizada na elaboração dos casos de teste levou em consideração as especificidades dos sites utilizados e da quantidade de e-mails obtidos. Sendo assim, foram elaborados 12 cenários de teste que são detalhados através da Tabela 1.

Table 1: Cenários de Teste para Envio de Notícias por E-mail

Caso de Teste	Descrição
Cenário 1	1 notícia para 1 endereço de email
Cenário 2	1 notícia para 5 endereços de emails
Cenário 3	1 notícia para 10 endereços de emails
Cenário 4	1 notícia para 50 endereços de emails
Cenário 5	5 notícias para 1 endereços de emails
Cenário 6	5 notícias para 5 endereços de emails
Cenário 7	5 notícias para 10 endereços de emails
Cenário 8	5 notícias para 50 endereços de emails
Cenário 9	10 notícias para 1 endereços de emails
Cenário 10	5 notícias para 5 endereços de emails
Cenário 11	5 notícias para 10 endereços de emails
Cenário 12	5 notícias para 50 endereços de emails

Note que os cenários de teste foram subdivididos em três categorias: envio de 1 notícia, envio de 5 notícias e envio de 10 notícias. Por sua vez, cada categoria foi submetida a quatro diferentes quantidades de destinatários: 1, 5, 10 e 50. Todos os 50 endereços de e-mail foram coletados através de solicitação a apoiadores do presente trabalho.

É relevante pontuar que a elaboração desses cenários de teste esteve limitada ao envio de no máximo 10 notícias por vez, devido ao formato de exibição dos sites do IFPA. Esses sites utilizam um mecanismo de paginação que limita a exibição a 10 notícias por página. Portanto, a aplicação fica restrita a extrair dados apenas da primeira página exibida pelo site. Partindo dessa limitação, foram utilizadas variações com números menores de notícias a fim de simular e observar os efeitos do aumento da carga nas duas aplicações. Seguindo o mesmo princípio de aumento gradual de carga, o número de endereços de e-mail segue uma sequência crescente de 1 a 50, que foi a quantidade máxima de endereços coletados.

No que se refere a execução dos cenários de teste, vale ressaltar que ambas aplicações, monolítica e microsserviços, foram executadas em seus respectivos containers Docker e as métricas utilizadas para avaliar o desempenho das aplicações, consumo de CPU e memória, foram obtidas através da ferramenta Grafana Cloud, que é uma plataforma de observabilidade e monitoramento de aplicações que, através da sua integração com o Docker Desktop, permite a extração das informações de consumo em tempo real dos containers executados e os exibe de forma visual através de dashboards [9].

O uso do Grafana se mostrou eficiente, uma vez que a ferramenta disponibilizou as informações de consumo de CPU e memória de cada execução registrados de forma visual. Para isso, foi necessário registrar, em cada execução, o horário de início, que marca o momento em que a aplicação foi iniciada, e o horário de término, que corresponde ao momento em que a aplicação enviou o último e-mail. Após a conclusão da execução, configurou-se a ferramenta para que exibisse o consumo das aplicações no período determinado, ou seja, entre o horário de início e o horário de término.

A seção seguinte apresentará os resultados obtidos através da execução dos cenários de teste supracitados, considerando para tal as métricas de tempo de envio, consumo de CPU e consumo de memória, a fim de avaliar o desempenho da aplicação baseada em uma arquitetura de microsserviços em detrimento a aplicação inspirada em uma arquitetura monolítica.

4.1 Avaliação de desempenho das aplicações

A partir da execução dos 12 cenários de teste para cada aplicação (monolítica e microsserviços), as médias de desempenho para as métricas de consumo de CPU e consumo de memória, assim como o tempo total de envio de uma notícia ou um bloco de notícias são apresentados através das Tabelas 2, 3 e 4, onde os campos **Cons. CPU** e **Cons. memória** correspondem ao percentual total de consumo de CPU e ao percentual total de consumo de memória RAM das aplicações durante a execução de cada teste, respectivamente.

Em uma análise global, é possível verificar que, na maioria dos casos, a aplicação baseada na arquitetura monolítica obteve resultados consideravelmente melhores na maioria dos cenários com relação ao consumo de memória, obtendo uma média de 83,59%, enquanto a arquitetura de microsserviços atingiu um consumo de memória em média de 90,96% nos 12 cenários de teste. A mesma tendência pode ser observada no caso do consumo de CPU, no qual a aplicação monolítica consome uma média de 14,30% enquanto a aplicação de microsserviços consome uma média de 19,86%. Em contrapartida aos resultados obtidos com relação ao consumo, no que tange o tempo total de envio das notícias, a aplicação de microsserviços obteve uma média de 36 segundos, enquanto a aplicação monolítica obteve 51 segundos. De uma maneira geral, os cenários de teste apontaram que, à medida que a quantidade de e-mails aumenta, o consumo de memória e

CPU é maior na aplicação de microsserviços, embora ela possua um tempo de envio menor que a aplicação monolítica.

Em todos os cenários onde o envio envolveu 50 e-mails (cenários 4, 8 e 12) pode-se observar que a aplicação de microsserviços obteve resultados significativamente superiores que a aplicação monolítica no que se refere ao tempo de envio, independentemente da quantidade de notícias enviadas. Os gráficos da Figura 11, da Figura 12 e da Figura 13 apresentam esse fato de forma visual, uma vez que evidenciam a disparidade de resultados entre as aplicações nos cenários que envolvem 50 e-mails, demonstrando que possuem em média o dobro de economia de tempo na aplicação de microsserviços. Através desse resultado é possível inferir que a aplicação de microsserviços poderá se comportar melhor que a monolítica com relação a um possível aumento de carga na aplicação.

Por outro lado, os cenários que envolveram o envio para apenas um endereço de e-mail (cenários 1, 5 e 9) não demonstraram uma diferença significativa entre as duas aplicações, em termos de tempo de envio, embora a aplicação monolítica tenha obtido melhores resultados no que se refere ao consumo de CPU e memória. Esse resultado já era esperado devido às características intrínsecas à arquitetura monolítica.

Ao analisar os resultados obtidos de forma específica, é possível observar que a Tabela 2 apresenta números semelhantes nos cenários iniciais (cenário 1 e cenário 2), com baixa diferença de tempo de envio entre as duas aplicações, porém com a aplicação monolítica tendo um consumo menor de recursos. Esse resultado pode ser justificado devido a menor quantidade de containers que estavam sendo executados simultaneamente durante a execução dos testes na aplicação monolítica. Já no cenário 4, a aplicação monolítica apresenta um tempo de envio significativamente maior, o que demonstra a eficiência da aplicação de microsserviços no envio de múltiplas mensagens.

Table 2: Resultados obtidos através do envio de 1 notícia

2*	Tempo total de envio		Consumo CPU		Consumo memória	
	MO	**MS**	**MO**	**MS**	**MO**	**MS**
Cenário 1	25s	**21s**	**13%**	19,9%	**79,7 %**	93,3%
Cenário 2	**17s**	**17s**	**11,7 %**	20,3%	**81,5 %**	91,2%
Cenário 3	**19s**	23s	13,2%	**12%**	86,5%	**85,2 %**
Cenário 4	76s	**37s**	24,7%	**20,8 %**	91,8%	**91,6 %**

A Tabela 3 apresenta resultados que seguem a mesma tendência da Tabela 2, nas quais inicialmente não há uma diferença significativa no tempo de envio, porém os microsserviços levam maior vantagem nos cenários finais, 7 e 8, que envolvem uma quantidade maior de e-mails, reforçando o desempenho superior dos microsserviços em cenários nos quais a carga é maior. No entanto, vale ressaltar o alto consumo de memória dos microsserviços em detrimento a monolítica, que ultrapassa 90% em todos os casos, o que pode indicar uma sobrecarga causada pela comunicação entre os serviços e suas execuções simultâneas.

Table 3: Resultados obtidos através do envio de 5 notícias

2*	Tempo total de envio		Consumo CPU		Consumo memória	
	MO	**MS**	**MO**	**MS**	**MO**	**MS**
Cenário 5	**33s**	36s	**13,5 %**	21%	**77,6 %**	93,1%
Cenário 6	**22s**	28s	21,9%	**19,8 %**	**81,4 %**	91,2%
Cenário 7	37s	**30s**	**12,2 %**	12,5%	**82,6 %**	91,8%
Cenário 8	111s	**44s**	**14,4 %**	19,8%	**84,8 %**	91,2%

No que diz respeito ao desempenho das aplicações no envio de 10 notícias, verifica-se através da Tabela 4 uma maior diferença entre os resultados obtidos pelas duas aplicações, uma vez que o consumo de recursos de CPU e memória foi menor na aplicação monolítica em todos os casos testados, com destaque para os cenários 9 e 11, onde a aplicação monolítica consumiu menos da metade dos recursos de CPU requeridos pelos microsserviços. Por outro lado, os cenários desses testes apontaram que os microsserviços necessitaram de menos tempo para envio das notícias em 3 dos 4 cenários simulados e um empate, com a diferença de tempo em relação ao monolítico aumentando paralelamente ao número de endereços de e-mail.

Table 4: Resultados obtidos através do envio de 10 notícias

2*	Tempo total de envio		Cons. CPU		Cons. memória	
	MO	**MS**	**MO**	**MS**	**MO**	**MS**
Cenário 9	**50s**	**50s**	**11,2 %**	24,3%	**90,1 %**	93,7%
Cenário 10	45s	**39s**	**11,8 %**	19,6%	**80,3 %**	89,9%
Cenário 11	57s	**44s**	**12,1 %**	29%	**83,2 %**	92,2%
Cenário 12	116s	**62s**	**11,9 %**	19,3%	**83,6 %**	87,1%

Observando os resultados outrora apresentados, é possível identificar uma tendência caracterizada pelo maior tempo de envio das notícias pela aplicação monolítica, juntamente com um consumo mais elevado de recursos dos microsserviços. Segundo [20], tal cenário é esperado, visto que a arquitetura de microsserviços pode ser composta por diversas aplicações e bancos de dados em execução simultaneamente. No entanto, vale ressaltar que houveram desvios no padrão identificado; como nos cenários 3 e 4, onde a aplicação monolítica consumiu mais recursos de hardware que os microsserviços, quando o contrário é esperado. Uma hipótese para esse ocorrido é um estresse da aplicação ao realizar o envio dos e-mails.

O gráfico da Figura 11 demonstra a tendência dos resultados anteriormente apresentados, onde o desempenho, com relação ao custo por tempo de envio, das duas aplicações é parelho nos estágios iniciais, que exigem uma carga menor. Porém, a aplicação de microsserviços se mostrou mais eficiente nos testes de maior estresse, tendo, em alguns casos, a capacidade de enviar todas as notícias com menos da metade do tempo que a solução monolítica, como observado no cenário 12 da Figura 13.

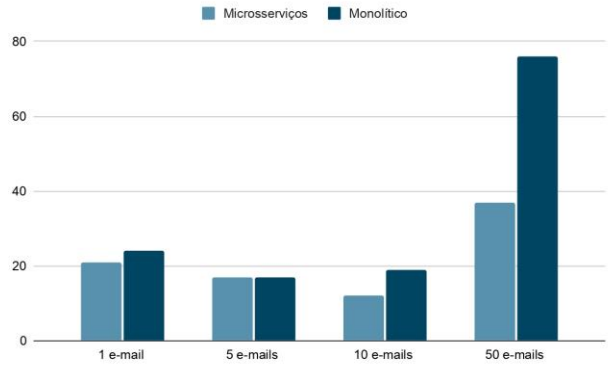

Figure 11: Gráfico com tempo de envio de uma notícia para cada número de destinatários.

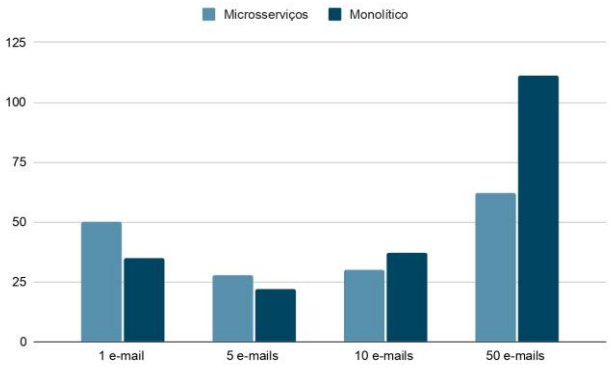

Figure 12: Gráfico com tempo de envio de cinco notícias para cada número de destinatários.

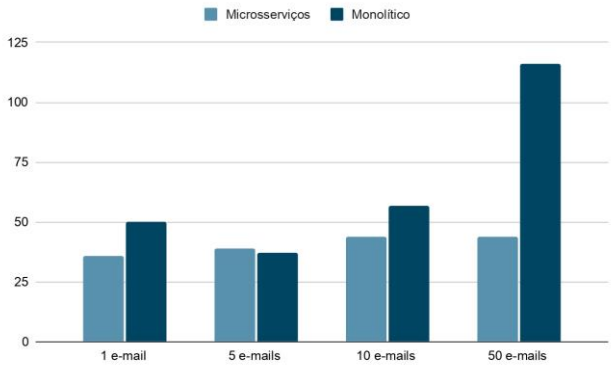

Figure 13: Gráfico com tempo de envio de dez notícias para cada número de destinatários.

Por fim, os resultados apresentados permitiram inferir que, apesar da aplicação monolítica demonstrar um consumo de hardware inferior em maior parte dos casos, os microsserviços se mostraram ser mais eficientes em relação ao tempo de envio das notícias, especialmente em cenários com maior carga de destinatários. Tais diferenças podem ser atribuídas à característica descentralizada da arquitetura de microsserviços, que permite a maior segmentação de tarefas, em contraste com a simplicidade ao modelo monolítico, que apesar de menor eficiência no tempo de envio, demonstrou maior economia de recursos computacionais.

5. CONCLUSÃO

O presente trabalho apresentou como proposta a criação de uma solução de software para extração e disseminação de notícias oriundas de dois sites do Instituto Federal do Pará, com o objetivo de alavancar a divulgação de informações úteis à comunidade acadêmica do campus de Altamira. Visando alcançar os objetivos propostos, primeiramente, foi desenvolvida uma página web com a finalidade de viabilizar a coleta dos endereços de e-mail daqueles interessados em receber as novas notícias inseridas nos sites institucionais. Em seguida, a técnica de *Web Scraping* foi utilizada para desenvolver o mecanismo de extração do conteúdo contido nos portais web da instituição, além do envio automático de e-mails para divulgação do conteúdo.

A aplicação responsável pela extração e envio das notícias foi desenvolvida seguindo uma arquitetura de software monolítica, na qual todas as funcionalidades do software se concentram em uma base de código única [20]. Posteriormente, a aplicação foi migrada para uma arquitetura de microsserviços, com o objetivo de proporcionar maior escalabilidade e modularidade para o sistema. Com base nessa evolução, foi realizada uma análise comparativa entre as duas versões da aplicação, visando avaliar os impactos dessa mudança. Tal comparação gerou materiais de estudo que poderão auxiliar desenvolvedores e arquitetos de software no contexto da utilização do Web Scraping com envio de e-mails, aplicados tanto em uma aplicação com arquitetura monolítica quanto em uma aplicação baseada em microsserviços.

Após o seu desenvolvimento, as duas aplicações foram expostas ao experimento comparativo, que foi realizado através da utilização das seguintes métricas: consumo de CPU, consumo de memória RAM e tempo de envio das notícias. Como observado na Tabela 1, os testes foram realizados em 12 cenários distintos, que buscaram avaliar o efeito gerado pela simulação do aumento gradual no volume de dados processados pelas duas aplicações.

Ao final da análise comparativa, é possível inferir que a aplicação monolítica obteve um menor consumo de CPU e memória na maioria dos cenários simulados, devido a sua estrutura simplificada, que exige menor carga de hardware. Com isso, ela se mostrou ser a opção mais adequada para cenários com recursos computacionais limitados e com menor fluxo de dados. Por outro lado, a aplicação baseada em microsserviços se mostrou mais eficiente com relação ao tempo de envio das notícias, especialmente nos cenários onde o volume de dados era mais elevado. Esse comportamento se deve à eficiência na comunicação interna entre os serviços existentes. Como resultado, essa arquitetura permite que um desenvolvedor adicione novas funcionalidades à aplicação sem a necessidade de modificar o código dos serviços já existentes ou limitar o desenvolvimento a uma única linguagem de programação. Essa flexibilidade é particularmente útil em cenários onde diferentes serviços exigem soluções tecnológicas específicas.

Logo, nesse modelo arquitetural, um microsserviço de postagens das mesmas notícias para uma rede social poderia ser desenvolvido com a linguagem Python, mesmo que todo o restante da aplicação seja desenvolvido em C#, bastando que a nova funcionalidade se conectasse à fila de mensagens RabbitMQ (Figura 10), que recebe as notícias extraídas pelo microsserviço já existente. A mesma versatilidade não pode ser aplicada a aplicação monolítica, uma vez que suas funcionalidades estão contidas inteiramente em um único bloco de código.

A partir das análises realizadas, pode-se concluir que a proposta de desenvolvimento de uma aplicação para a disseminação de notícias utilizando Web Scraping, assim como a análise comparativa das duas arquiteturas utilizadas na implementação das aplicações, tem validade no que tange o aumento da distribuição de informação no contexto do IFPA, trazendo consigo as seguintes contribuições:

- Fornecimento de uma ferramenta para automatização do processo de divulgação das notícias do Instituto Federal do Pará;
- Desenvolvimento de uma metodologia replicável para a implementação de sistemas de extração e envio de notícias, que pode ser adaptada por outras instituições de ensino ou organizações;
- Fornecimento de material no que tange a análise comparativa entre a arquitetura monolítica e de microsserviços no contexto da utilização do *Web Scraping* com envio automático de e-mails, podendo ser utilizado para estudo ou apoio para decisões arquitetônicas;
- Contribuição para a literatura acadêmica sobre *Web Scraping* e arquiteturas de software.

Com o objetivo de dar continuidade a pesquisa desenvolvida, a seguir são elencados possíveis trabalhos futuros:

- Implementação de uma funcionalidade para o envio automático das notícias via WhatsApp;
- Disseminação de notícias de outros campi do IFPA;
- Permitir que o usuário selecione assuntos de interesse, para seleção personalizada de conteúdo;
- Postagem automática das notícias em redes sociais;
- Conectar a aplicação em plataforma especializada em envio de e-mails em massa;
- Envio de relatório mensal com as principais notícias;
- Adição de outros testes na etapa de análise entre as duas arquiteturas, com o objetivo de reforçar a confiabilidade dos resultados, além de acrescentar outras métricas de desempenho como insumo para esses testes.

6. REFERÊNCIAS

[1] C. C. Baggio, H. Costa, and U. Blattmann. Seleção de tipos de fontes de informação. *Revista de Ciência da Informação*, 2016.

[2] L. Bass, P. Clements, and R. Kazman. *Software Architecture in Practice*. SEI series in software engineering. Addison-Wesley, 2003.

[3] V. Benavente, L. Yantas, I. Moscol, C. Rodriguez, R. Inquilla, and Y. Pomachagua. Comparative analysis of microservices and monolithic architecture. In *2022 14th International Conference on Computational Intelligence and Communication Networks (CICN)*, pages 177–184, 2022.

[4] C. M. Borges Silva de Oliveira, I. Lopes Garcia Nascimento, J. Jasão Melo da Silva, and F. R. Leite Mota. Competência em informação: análise bibliométrica e altmétrica da comunicação científica (2019-2021). *P2P E INOVAÇÃO*, 9(2):415–433, Mar. 2023.

[5] B. E. Costa. Monitoramento do envelhecimento de software relacionado à memória: Um estudo exploratório. Trabalho de Conclusão de Curso (Bacharelado em Sistemas de Informação) – Universidade Federal de Uberlândia, Uberlândia, 2018, 2018.

[6] O. De Souza. *CONSEQUÊNCIAS DO USO DE SISTEMAS AUTÔNOMOS NA GESTÃO E DISSEMINAÇÃO DE INFORMAÇÕES*, pages 43–62. 11 2023.

[7] S. S. Delfino, J. A. S. de Pinho Neto, and M. R. F. de Sousa. Desafios da sociedade da informação na recuperação e uso de informações em ambientes digitais. *Revista Digital Biblioteconomia e Ciência da Informação*, 17:1–16, Nov 2019.

[8] S. J. Fowler. *Microsserviços prontos para a produção: Construindo sistemas padronizados em uma organização de engenharia de software*. Novatec Editora, Apr. 2019.

[9] Grafana Labs. Grafana documentation, 2024. Acessado em: 22 de julho de 2024.

[10] J. Hillen. Web scraping for food price research. *British Food Journal*, 121(12):3350–3361, Nov. 2019.

[11] HtmlAgilityPack Team. *HtmlAgilityPack Documentation*. GitHub, 2024.

[12] L. Johansson and D. Dossot. *RabbitMQ Essentials: Build distributed and scalable applications with message queuing using RabbitMQ*. Packt Publishing Ltd, 2020.

[13] M. Johansson and O. Isabella. Comparative study of rest and grpc for microservices in established software architectures, 2023.

[14] M. Khder. Web scraping or web crawling: State of art, techniques, approaches and application. *International Journal of Advances in Soft Computing and its Applications*, 13(3):145–168, Nov. 2021.

[15] M. L. G. d. Lara and V. L. Conti. Disseminação da informação e usuários. *São Paulo em Perspectiva*, 17(3–4):26–34, Dec. 2003.

[16] J. P. D. Lucio. Análise comparativa entre arquitetura monolítica e de microsserviços, 2017.

[17] H. F. S. Marques. Migração de arquitetura: Monolítico para microsserviços usando domain-driven design. http://hdl.handle.net/10400.22/19534, 2021.

[18] R. C. Martin. *Clean architecture*. Prentice Hall, Philadelphia, PA, Sept. 2017.

[19] R. Mitchell. *Web Scraping with Python, 2e*. O'Reilly Media, Sebastopol, CA, Apr. 2018.

[20] S. Newman. *Criando Microsserviços.* Novatec, 2 edition, 2022.

[21] R. T. Oliveira. Documentação de produto implementado por software: Conexãoif. B.S. thesis, 2023.

[22] J. R. Pereira, J. F. d. A. Barros, R. d. S. Freire, and V. Vieira França. A qualidade da usabilidade dos portais de transparência das universidades federais do nordeste do brasil. *Revista do Serviço Público*, 72(4):803–823, dez. 2021.

[23] M. Richards and N. Ford. *Fundamentals of Software Architecture: An Engineering Approach*. O'Reilly Media, 1 edition, 2020.

[24] L. Richardson. Beautiful soup documentation, 2007.

[25] T. C. Rosa and L. A. Baptaglin. O acesso a informação a ao patrimônio documental no instituto federal de roraima: conquistas e desafios. *Pesquisa Brasileira em Ciência da Informação e Biblioteconomia*, 13(1), May 2018.

[26] M. Shaw and D. Garlan. *Software Architecture: Perspectives on an Emerging Discipline.* Pearson, Upper Saddle River, NJ, Apr. 1996.

[27] C. J. Stradolini. Migração de sistemas monolíticos para microsserviços: Estudo de caso de migração de um módulo de pagamentos de e-commerce. http://hdl.handle.net/10183/243216, 2022.

[28] F. Tapia, M. Ángel Mora, W. Fuertes, H. Aules, E. Flores, and T. Toulkeridis. From monolithic systems to microservices: A comparative study of performance. Applied Sciences, 2020.