Desenho Metodológico para um Sistema de Busca e Gerenciamento de Portarias: Otimização do Acesso à Informação Administrativa

1st Thiago Castro da Conceição, 2nd Given Name Surname, 3rd Given Name Surname
dept. name of organization (of Aff.), City, Country
dept. name of organization (of Aff.), City, Country if needed
thiago.conceicao@icen.ufpa.br

***Abstract*—This paper presents a methodological framework for developing a search and management system for administrative ordinances, aiming to optimize access to public information. The study explores the integration of artificial intelligence techniques, particularly Large Language Models (LLMs), with a web application to enhance the relevance of search results. The system employs Optical Character Recognition (OCR) for text extraction, CouchDB for document storage, LangChain for semantic search, and Docker for scalability and portability. The initial implementation within the Ministério Público de Contas do Estado do Pará (MPC-Pa) demonstrated improvements in efficiency and accessibility. The results indicate that the proposed solution enhances search precision, although challenges related to processing speed and intelligent classification remain. Future work includes migrating to a more robust infrastructure, improving document categorization with machine learning, and implementing a question-answering system for enhanced user interaction.**

***Index Terms*—component, formatting, style, styling, insert**

I. Introdução

O uso de novas tecnologias está cada vez mais comum nos mais diversos setores da sociedade e a inteligência artificial (IA) tem ganhado notoriedade como uma destas tecnologias emergentes. As estratégias implementadas pela Inteligência Artificial se fundamentam no modelo cognitivo, funcionando de forma semelhante ao raciocínio humano, avaliando os dados através de processos que se assemelham ao raciocínio na resolução de problemas, na interpretação de linguagem natural e na aquisição de conhecimento.

Campos e Figueiredo (2022) dizem que o avanço da inteligência artificial nos últimos anos foi impulsionado pelo"[...] avanço das estatísticas e métodos probabilísticos; pelo aumento da quantidade de dados; pelo poder computacional mais potente e mais econômico; e pela transformação de locais em ambientes propícios à tecnologia [...]" [4].

Os modelos preditivos, a identificação automática de fraudes, a avaliação de riscos baseada em dados, a segurança doméstica, a vigilância, a previsão de policiamento, o reconhecimento facial, a assistência médica, a triagem de pacientes, o suporte jurídico, o transporte, o processamento de linguagem natural e as relações públicas são exemplos de como a inteligência artificial tem sido empregada em diversos setores ao redor do mundo.

A aplicação atual da inteligência artificial (IA) na gestão pública do Brasil, devido aos seus benefícios, tem auxiliado na consolidação do princípio da eficiência. Este, que é a base do Direito Administrativo brasileiro, busca alcançar o melhor resultado possível utilizando o mínimo de recursos, sejam eles orçamentários ou de infraestrutura.

Desde a Emenda Constitucional n°19 de 1998, o princípio da eficiência passou a integrar o grupo de princípios que norteiam a Administração Pública, conforme claramente definido no artigo 37, caput, da Constituição Federal. A inclusão do princípio citado simboliza o reforço de um dever específico do administrador público de cumprir suas obrigações com rapidez, exatidão e eficácia operacional. A eficiência na gestão é fundamental para avaliar os custos de atender às necessidades públicas em relação ao grau de utilidade obtido. [3].

No entanto, frequentemente, devido à escassez de pessoal e recursos financeiros, a Administração Pública não alcança a eficiência desejada. Isso resulta, além da insatisfação pela falta de qualidade no serviço público, em desafios para controlar possíveis fraudes e comportamentos corruptos dentro das entidades públicas. Portanto, o administrador público precisa recorrer a métodos alternativos para atingir a eficiência administrativa almejada [2].

Dado esse contexto, a seguinte pesquisa busca examinar como a aplicação da Inteligência Artificial pode auxiliar na realização do princípio da eficiência na Administração Pública. A tecnologia trouxe inúmeros benefícios para várias áreas, incluindo o setor público, que pode utilizar tecnologias inovadoras para melhorar suas atividades.

Várias entidades governamentais no Brasil já estão adotando sistemas de Inteligência Artificial para aprimorar a eficiência operacional, como é o caso do Tribunal de Contas da União, que utiliza os robôs Alice, Sofia e Mônica para detectar possíveis irregularidades em contratos públicos que envolvem fundos federais. Alice já auxiliou os auditores na interrupção

Identify applicable funding agency here. If none, delete this.

de várias licitações irregulares em todo o país, evidenciando sua contribuição para a melhoria, rapidez e eficácia do serviço público fornecido pelo órgão. Este e outros exemplos serão examinados ao longo do trabalho [5].

Com isso, o presente artigo tem como objetivo realizar uma análise do uso de uma aplicação web desenvolvida para buscar portarias (documentos de caráter administrativo) do Ministério Público de Contas do Estado do Pará (MPC-Pa) e sua integração com um modelo de LLM (Large Language Model) para melhorar a relevância da pesquisa, gerando um entendimento para o significado semântico das consultas realizadas no sistema, e não apenas capturando a correspondência das palavras pesquisadas dentro desse sistema. As estratégias metodológicas abordadas pela pesquisa foram divididas em diversas etapas, com o intuito de desenvolver resultados mensuráveis acerca da seleridade, eficiência e economia que aplicação traz, porque até então tal processo é feito de forma manual dentro do MPC-Pa.

Dessa forma o trabalho se estrutura em diversas etapas. A primeira é referente a fundamentação teórica da pesquisa, a segunda é metodologia adotada para a pesquisa. A terceira está associada aos resultados da pesquisa, a quarta é a discussão do trabalho, a quinta refere-se as limitações encontradas na pesquisa. A sexta e sétima abordam respectivamente as conclusões e trabalhos futuros.

II. Fundamentação Teórica

Nesta Seção, alguns conceitos fundamentais para compreensão do trabalho serão apresentados. Conceitos a respeito da inteligência Artificial (IA), Large Language Model (LLM), Lang Chain, Docker e dos princípios da eficiência pública.

A. Inteligência Artificial

A inteligência artificial é a ciência que tem a capacidade de desenvolver sistemas computacionais inteligentes que possam utilizar computadores para compreender a inteligência humana. No entanto, a I não necessita de métodos biológicos observáveis [6].

A inteligência artificial baseia-se em quatro métodos, que são: o de pensar como seres humanos, agir como seres humanos, pensar racionalmente e agir racionalmente. Em outras palavras a IA tenta compreender e criar entidade inteligentes. É um ramo da ciência computacional que surgiu recentemente que possui diversos subcampos nessa área, dentre esses a aprendizado de máquina [7].

A Inteligência Artificial busca o aumento constante da eficiência, simulando habilidades humanas como:

- resolução de problemas;
- compreensão de linguagem natural;
- visão robótica;
- sistemas especialistas e aquisição de conhecimento;
- metodologias de representação de conhecimento

Com base nisso, os estudo em IA buscam recriar o raciocínio humano acompanhados das seguintes definições:

- Sistemas especializados ou fundamentados em conhecimento (programas criados para obter e compartilhar o conhecimento operacional de um especialista humano, prontos a oferecer sugestões e comparações entre circunstâncias similares);
- sistemas de aprendizagem inteligente;
- interpretação/criação de voz;
- análise de imagem e cena em tempo real;
- programação automática

B. Large Language Models

Os LLM são modelos de aprendizagem de máquina capazes de realizar várias funções de Processamento de Linguagem Natural (PLN), tais como a criação de textos automatizados, a resposta a questões do usuário (chatbots), a tradução de textos, entre outras [8]. As suas implementações geralmente se fundamentam na arquitetura. As redes transformers [9], em contraste com as RNN (Recurrent Neural Network) que empregam a recorrência como estratégia principal para estabelecer relações entre tokens de uma sequência, empregam o mecanismo de self-attention para estabelecer tais relações. A partir da soma ponderada de uma sequência específica, o modelo determina dinamicamente quais tokens na sequência são mais pertinentes entre si.

Como o próprio nome indica, os LLM se destacam pela sua grandeza e habilidade de lidar com grandes quantidades de texto. Assim, eles conseguem compreender e produzir uma resposta compreensível na linguagem humana, habilitando-os a serem utilizados em diversas situações [10]. Os resultados eficazes vêm da elevada habilidade de produzir uma saída com uma interpretabilidade satisfatória, graças à sua arquitetura sólida, que foi treinada através de aprendizagem supervisionada sobre um vasto conjunto de dados textuais.

C. LangChain

LangChain é um framework para o desenvolvimento de aplicações baseadas em modelos de linguagem. Este framework facilita a criação de aplicações caracterizadas por [11]:

1) **Consciência de Contexto**: Estabelecer uma conexão entre um modelo de linguagem e várias fontes de contexto, como instruções de prompt, exemplos de poucos disparos ou conteúdo contextual, permite que a aplicação fundamenta suas respostas de maneira eficaz.
2) **Capacidade de raciocínio**: Aproveitar um modelo de linguagem para raciocínio, incluindo a determinação de como responder com base no contexto fornecido e decidir sobre as ações apropriadas, capacita as aplicações a navegar por cenários complexos.

Em aplicações do mundo real, as interações com o LLM não são isoladas, mas sim constituem uma série de etapas em que resultados intermediários exigem processamento lógico para iniciar a etapa subsequente. Essa sequência de interações combinadas é chamada de Chains (Cadeias), que normalmente envolve integrações com um ou mais provedores de modelos, sistemas de armazenamento de dados e APIs, entre outros. Esses módulos podem ser combinados para construir aplicações mais complexas ou usados individualmente para aplicações mais simples. LangChain se revela fundamental no

design de aplicações como assistentes pessoais, chatbots, questionamento e resposta (QA) baseado em documentos, resumo de documentos extensos, extração de informações estruturadas de texto não estruturado, consulta de dados tabulares [11].

D. Docker

O Docker surge para fornecer aplicações de forma mais ágil do que o habitual, empregando a técnica de encapsulamento, que envolve a criação de um objeto dividido que se ajusta a todos os níveis de qualquer ambiente, racionalize a quantidade de serviço e a responsabilidade da empresa que gerencia softwares ágeis.

A sua abrangência proporciona vários benefícios, como a utilização cuidadosa de especificações, e atua de maneira distinta ao separar os níveis processuais através do kernel do computador, ao invés de utilizar todo o sistema operacional virtualizado. A portabilidade permite que os pacotes sejam executados em todos os *host's* do Docker. O servidor não é notado sobre todas as atividades que ocorrem no contêiner, graças à previsibilidade das interfaces que seguem um padrão e mantêm uma comunicação consistente. A disponibilidade de uma nuvem pública permite a criação de ambientes personalizados de acordo com as necessidades individuais, sempre respeitando um padrão de configurações [12].

Um aspecto do seu funcionamento é o empacotamento de aplicações em contêineres. Este pacote se converte em uma imagem Docker que pode ser executada em diversas plataformas. Se a execução for bem-sucedida em um ambiente mais simples, como um computador doméstico, a imagem Docker pode ser reproduzida em diferentes plataformas para ser automaticamente executada em um servidor sem qualquer dificuldade. Os Contêineres do Linux (LXC) contribuem para a ampliação de serviços disponíveis e a simplificação de sua utilização, atuando na separação de procedimentos e programas, assemelhando-se a um ambiente virtual. contudo, é bastante leve e ligado ao *host* [12].

Os procedimentos para isolar no ambiente Docker envolvem a virtualização do sistema operacional, onde é permitida a execução simultânea de vários processos que são processados de maneira independente por um único *host*. Adicionalmente, o LXC permite a formação de grupos *namespaces* para as partes essenciais, que são separadas do cgroups encarregado de dividir o processamento das solicitações.

Em ambientes virtualizados, as máquinas virtuais criam duas réplicas completas de *hardware* e *software* do *host*, uma no local físico e outra no ambiente que será virtualizado. Portanto, as máquinas virtuais apresentam uma perda de desempenho em comparação com os contêineres. O Docker serve para alinhar as configurações fundamentais de um sistema operacional, permitindo sua execução sem a necessidade de um servidor separado. Ele opera com base em três características: contêineres, registros e imagens.

E. Princípios da eficiência pública

Ao se falar em eficiência na administração pública, refere-se em como o gestor público deve gerir as obrigações públicas, com economia, transparência e moralidade visando cumprir todas as metas estabelecidas em cada setor público.

Como colocou Maria Sylvia Zanella Di Pietro,"o princípio apresenta-se sob dois aspectos, podendo tanto ser considerado em relação à forma de atuação do agente público, do qual se espera o melhor desempenho possível de suas atuações e atribuições, para lograr os melhores resultados, como também em relação ao modo racional de se organizar, estruturar, disciplinar a administração pública, e também com o intuito de alcance de resultados na prestação do serviço público" ... [13].

A autora ainda fala que "a eficiência é um princípio que se soma aos demais princípios impostos à administração, não podendo sobrepor-se a nenhum deles, especialmente ao da legalidade, sob pena de sérios riscos à segurança jurídica e ao próprio Estado de direito" ... [13].

É obrigatório para todo funcionário público executar suas funções com rapidez, precisão e eficiência profissional. Trata-se do princípio mais atual da função administrativa, que já não se satisfaz apenas com o cumprimento da lei, demandando resultados positivos para o serviço público e um atendimento adequado às necessidades da comunidade e de seus integrantes. Ele ainda afirma que "a obrigação de eficiência está alinhada à obrigação de uma administração eficaz" [14].

III. Metodologia

Nesta Seção, serão apresentados os métodos utilizados para o desenvolvimento da pesquisa como mostra a figura 1 . Processos como a catalogação de todas as portarias (documentos administrativos) presentes no site MPC-Pa, a realização do processo de OCR (Optical Character) para reconhecimento de caracteres dos documentos administrativos do MPC-Pa, posteriormente a criação do banco de dados no CouchDB para configurar com a aplicação web criada no quasar para fazer as consultas das portarias, a implementação de LangChain na aplicação web e por fim o uso do docker garantindo portabilidade e escalabilidade do sistema.

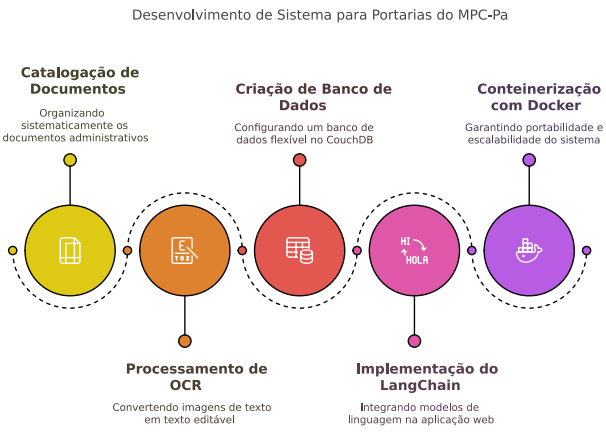

Fig. 1. Diagrama da metodológico

Inicialmente, foi realizada uma análise detalhada dos documentos existentes (portarias) para compreender sua estrutura

e formato. A partir dessa análise, foram definidos os casos de uso que orientariam o desenvolvimento e os testes do sistema. O objetivo do trabalho é catalogar todas as portarias (arquivos PDF) presentes no site do Ministério Público de Contas do Estado do Pará (MPC-PA). Para isso, foi desenvolvido o módulo `pdf_downloader.py` em Python, responsável por realizar o download e a organização desses arquivos.

O módulo `pdf_downloader.py` é composto por diversas funções que trabalham de forma conjunta para realizar o processo de catalogação. Abaixo, descrevemos as principais funções que compõem este módulo:

A. Download dos arquivos

1) Função Principal: `pdf_downloader.py`: O processo inicial consiste na criação de uma sessão HTTP otimizada para múltiplas requisições, que permite a reutilização da conexão para melhorar o desempenho e reduzir a sobrecarga no servidor de destino. A seguir, as principais etapas do módulo são detalhadas:

- **Inicialização das Requisições**:
  - A função principal inicia as requisições ao site do MPC-PA criando um objeto `requests.Session()`.
  - Essa abordagem otimiza as conexões HTTP, permitindo que a mesma conexão seja reutilizada para múltiplas requisições, o que melhora a eficiência do processo.
- **Extração dos Links**:
  - Após a inicialização da sessão, a função `get_pdf_first_page(url)` é utilizada para extrair os links dos arquivos PDF.
  - Durante reuniões com representantes de cada setor do órgão, foi sugerido que apenas a primeira página do arquivo fosse capturada. A função `get_pdf_first_page` realiza exatamente essa tarefa, extraindo o link correspondente à primeira página do PDF.

2) Funções Auxiliares: Além da função principal, o módulo `pdf_downloader.py` também conta com funções auxiliares que desempenham papéis específicos no processo de download e catalogação dos arquivos:

- **Função `get_pdf_links_first_page`**:
  - Esta função realiza uma requisição HTTP do tipo GET ao site do MPC-PA.
  - Utiliza a biblioteca `BeautifulSoup` para fazer a análise do HTML da página e localizar os links para os arquivos PDF.
  - Filtra os links identificando aqueles que terminam com a extensão `.pdf` ou que contêm a palavra `download`.
  - Em seguida, a função realiza a remoção de duplicatas e adiciona as URLs a um conjunto, garantindo que não haja links duplicados.
- **Função `download_pdf`**:
  - Esta função extrai o nome dos arquivos a partir de seus links, decodifica caracteres especiais presentes nas URLs e limpa o nome dos arquivos, substituindo caracteres inválidos por underscores (_).
  - Antes de realizar o download, verifica se o arquivo já existe no diretório de saída. Caso o arquivo já tenha sido baixado, o download é ignorado e uma mensagem é registrada.
  - A função também verifica o cabeçalho `content-type` da resposta HTTP para garantir que o conteúdo retornado seja, de fato, um arquivo PDF.

B. Tratamento dos Arquivos PDF (Módulo `pdf_processor.py`)

A extração de texto de documentos PDF é uma tarefa desafiadora, especialmente quando se trata de documentos que não contêm texto embutido, como arquivos escaneados ou imagens. O principal objetivo dessa etapa é realizar a extração eficiente do conteúdo textual dos arquivos PDF e salvar o resultado em formato de texto simples (`.txt`), de forma que possa ser facilmente manipulado e armazenado.

1) Função Principal: `process_pdfs`: Esta função coordena o processamento dos documentos PDF, realizando a extração de conteúdo e salvando os resultados de forma organizada.

1) **Parâmetros**:
   - `pdf_dir`: Diretório de entrada contendo os arquivos PDF que serão processados.
   - `output_dir`: Diretório de saída onde os arquivos de texto extraídos serão armazenados.
2) **Processo**:
   a) **Criação do Diretório de Saída**: Caso o diretório de saída não exista, ele é criado para garantir que todos os arquivos extraídos sejam corretamente armazenados.
   b) **Listagem dos Arquivos PDF**: A função percorre o diretório de entrada e lista todos os arquivos com a extensão `.pdf`, preparando-os para o processamento.
   c) **Iteração sobre os Arquivos**: Cada arquivo PDF é processado individualmente pela função `process_single_pdf`, que se encarrega de extrair o texto.
   d) **Controle de Progresso**: Para cada 5 arquivos processados, a função registra o progresso no console, permitindo ao usuário acompanhar a execução do processo, o que é especialmente útil em grandes volumes de dados.
3) **Retorno**: A função retorna o número de arquivos que foram processados com sucesso, oferecendo uma métrica de desempenho da operação.

2) Função Auxiliar: `process_single_pdf`: Esta função é responsável pela extração do conteúdo textual de um arquivo PDF individual. Ela lida com a extração de texto e a conversão do conteúdo para um formato legível.

1) **Parâmetros**:
   - `pdf_path`: Caminho completo para o arquivo PDF.

- `output_dir`: Diretório onde o arquivo de texto extraído será salvo.

2) **Processo**:
   a) **Extração do Nome do Documento (OCR)**: A função tenta obter o nome do documento, inicialmente realizando OCR sobre uma área destacada do PDF, e, se falhar, usa o nome do arquivo PDF como nome padrão.
   b) **Abertura do PDF e Extração do Texto**: Utilizando a biblioteca `fitz` (parte do PyMuPDF), o PDF é aberto, e o texto é extraído de cada página. Se o texto não for extraído corretamente, a função recorre ao OCR utilizando o `pytesseract` para tentar recuperar o conteúdo visualmente presente.
   c) **Construção do Conteúdo**: O texto extraído de cada página é concatenado em uma única string que representa o conteúdo completo do documento.
   d) **Escrita no Arquivo de Texto**: O conteúdo extraído é salvo em um arquivo de texto com a codificação UTF-8, garantindo a preservação de caracteres especiais e acentuação, o que é essencial para o manuseio adequado de textos em diferentes idiomas.
3) **Retorno**: A função retorna `True` se o processamento for bem-sucedido e `False` caso contrário, proporcionando um mecanismo de verificação de sucesso para cada arquivo.

3) Função Auxiliar: `extract_document_name`: Esta função executa uma tarefa específica de extração do nome do documento utilizando OCR, essencial para categorizar e organizar os documentos de forma mais eficiente.

1) **Parâmetros**:
   - `pdf_path`: Caminho completo para o arquivo PDF.
2) **Processo**:
   a) **Extração da Imagem da Primeira Página**: A primeira página do PDF é convertida em uma imagem utilizando a função `get_pixmap()` da biblioteca `fitz`, possibilitando a aplicação de OCR diretamente sobre uma representação visual do conteúdo.
   b) **Recorte da Imagem**: A área onde se espera encontrar o nome do documento é recortada manualmente, com as coordenadas ajustáveis conforme o layout do documento. Isso requer conhecimento prévio sobre a estrutura dos arquivos PDF em questão.
   c) **Aplicação de OCR**: A imagem recortada é submetida ao `pytesseract` para a extração de texto, utilizando parâmetros ajustados para melhorar a acurácia, como o modo de segmentação de página e a definição de uma lista de caracteres permitidos.
   d) **Limpeza do Texto**: O texto extraído passa por um processo de limpeza para remover caracteres especiais, espaços em branco excessivos e outros ruídos, visando obter uma versão mais precisa e legível do nome do documento.
3) **Retorno**: Retorna o nome do documento extraído ou `None` caso o OCR falhe.

C. Upload dos Documentos para o Banco de Dados (Módulo `db_uploader.py`)

Após a extração e organização do conteúdo textual, a próxima etapa é o upload dos dados para um banco de dados CouchDB, que oferece uma maneira eficiente de armazenar e consultar grandes volumes de documentos em formato JSON. O uso de um banco de dados NoSQL como o CouchDB é especialmente adequado para lidar com documentos não estruturados e metadados associados.

1) Função Principal: `recreate_database`: Esta função coordena a criação de um banco de dados no CouchDB e o upload dos documentos extraídos.

1) **Parâmetros**:
   - `text_dir`: Diretório contendo os arquivos de texto a serem carregados.
   - `database_name`: Nome do banco de dados onde os documentos serão armazenados.
2) **Processo**:
   a) **Inicialização**: A função cria uma instância da classe `CouchDBUploader`, que é responsável por gerenciar a comunicação com o CouchDB, incluindo autenticação e upload de dados.
   b) **Gerenciamento do Banco de Dados**: Caso o banco de dados especificado já exista, ele é excluído e recriado para garantir que os dados sejam carregados em um novo ambiente limpo. Em seguida, índices são criados nos campos mais relevantes, como conteúdo, tags e embeddings, para otimizar as consultas futuras.
   c) **Upload dos Documentos**: A função percorre os arquivos de texto no diretório de entrada e, para cada documento, gera um dicionário JSON com o conteúdo, metadados e tags. Esses documentos são carregados em lotes para o CouchDB, o que aumenta a eficiência do processo.
3) **Retorno**: A função retorna `True` se o upload for bem-sucedido, e `False` caso contrário.

2) Classe Auxiliar: `CouchDBUploader`: A classe `CouchDBUploader` encapsula as operações de comunicação com o CouchDB, proporcionando um mecanismo robusto para o gerenciamento de dados.

1) **Atributos**:
   - `base_url`: URL do servidor CouchDB.
   - `auth`: Credenciais de autenticação para garantir a segurança na comunicação.
   - `batch_size`: Tamanho dos lotes para o upload em massa, o que é crucial para otimizar a performance ao lidar com grandes volumes de dados.
2) **Métodos**:
   a) Métodos para gerenciar o banco de dados (`delete_database`, `create_database`, `_create_indexes`).

b) Métodos para upload de documentos individuais ou em lotes (`upload_document`, `upload_batch`).
c) Simulação de geração de embeddings e tags para o texto (`_generate_simulated_embedding`, `_generate_simulated_tags`).

As seções A, B e C descritas acima referem-se as duas primeiras etapas da figura 1. Agora faleremos da criação do banco de dados e da implementação do Lanchain.

D. Criação da Base de Dados (CouchDB)

O CouchDB foi escolhido como o sistema de gerenciamento de banco de dados (SGBD) para este projeto, desempenhando um papel crucial no armazenamento e na organização dos dados extraídos das portarias. Sua natureza NoSQL, orientada a documentos, oferece flexibilidade e escalabilidade, características essenciais para lidar com a diversidade e o volume de informações presentes nos documentos administrativos.

E. Modelo de Dados

Cada portaria é representada como um documento JSON individual no CouchDB. A estrutura do documento é projetada para acomodar tanto o conteúdo textual extraído quanto os metadados relevantes, além de informações adicionais geradas durante o processamento, como tags e embeddings simulados. Essa abordagem permite a fácil adaptação a diferentes formatos de portarias e a inclusão de novos campos no futuro, garantindo maior flexibilidade ao sistema.

1) Processo de Criação e Indexação:

1) **Conexão com o CouchDB:**
   - O script `db_uploader.py` estabelece uma conexão com o servidor CouchDB utilizando a biblioteca `requests`, autenticando-se com as credenciais fornecidas.
2) **Criação do Banco de Dados:**
   - O script verifica se o banco de dados especificado já existe. Caso exista, ele é excluído para garantir a consistência dos dados.
   - Em seguida, um novo banco de dados é criado para armazenar os documentos atualizados.
3) **Criação de Índices:**
   - A criação de índices é fundamental para otimizar o desempenho das consultas.
   - O script define índices nos campos `content`, `tags` e `embedding`, permitindo que o CouchDB execute buscas eficientes com base nesses critérios.
   - A criação dos índices é realizada através de requisições HTTP POST para o endpoint `/index` do CouchDB.

A implementação do Langchain encontrou algumas limitações no ambiente de desenvolvimento e de recursos computacionais. Tendo em algumas partes do código apenas a estrutura para simular a utilização de alguns dos principais componentes do Langchain que pode ser melhor estruturado em uma futura implementação em um ambiente mais adequado.

F. Langchain

O Langchain foi implementado para extração, organização e busca semântica de informações contidas nas portarias. As etapas são descritas a seguir.

1) Configuração do Ambiente: Configurar o ambiente para execução dos módulos do Langchain, preparando as bibliotecas e dependências necessárias.

- **Implementação:**
- Configuração de um ambiente Python com as bibliotecas necessárias do Langchain.
- Utilização da biblioteca `sentence-transformers` para geração de embeddings.
- Integração com o banco de dados vetorial `Chroma` para armazenamento e busca por similaridade.

2) Carregamento e Divisão dos Documentos: Carregar os documentos PDF e dividi-los em partes menores para facilitar o processamento por modelos de linguagem.

- **Implementação:**
- Utilização do `PyPDFLoader` do Langchain para carregar os documentos PDF.
- Implementação do `RecursiveCharacterTextSplitter` para dividir os documentos em partes menores, preservando a estrutura do texto.
- **Benefícios:**
- Permitir o processamento de documentos muito grandes.
- Melhorar a precisão da busca semântica, focando em partes menores e mais relevantes do documento.

3) Geração de Embeddings: Representar o significado semântico dos documentos e das consultas de pesquisa em um espaço vetorial.

- **Implementação:**
- Utilização do modelo `HuggingFaceEmbeddings` para gerar embeddings vetoriais para cada parte do documento.
- **Benefícios:**
- Permitir a busca por similaridade semântica.
- Melhorar a precisão da classificação de tópicos.

4) Armazenamento Vetorial: Armazenar os embeddings vetoriais de forma eficiente para permitir a busca por similaridade.

- **Implementação:**
- Utilização do `Chroma` como banco de dados vetorial para armazenar os embeddings.
- Indexação eficiente para permitir buscas por similaridade.
- **Benefícios:**
- Permitir a busca por similaridade em grandes volumes de dados.
- Oferecer recursos avançados de indexação e filtragem.

5) Busca Semântica: Permitir a recuperação de documentos relevantes com base na similaridade semântica.

- **Implementação:**
- Criação de uma função que recebe a consulta do usuário.
- Geração do embedding correspondente à consulta.
- Realização de uma busca por similaridade no banco de dados vetorial.
- **Benefícios:**

- Recuperação eficiente de documentos relevantes
- Busca semântica mais precisa em comparação à busca por palavras-chave.

6) Classificação de Tópicos: Classificar os documentos em categorias predefinidas para facilitar a organização e recuperação.

- **Implementação:**
- Utilização do Langchain para classificar os documentos utilizando um modelo de classificação de texto pré-treinado.
- Atribuição automática de tags aos documentos com base nos tópicos identificados.
- **Benefícios:**
- Organização automatizada dos documentos.
- Melhora na eficiência da busca e recuperação de informações.

G. Conteinerização do Docker

A aplicação de busca de portarias foi desenvolvida utilizando contêineres Docker para garantir modularidade, escalabilidade e facilidade de implantação. A arquitetura da aplicação é composta por três serviços principais, cada um isolado em um contêiner Docker, permitindo uma execução independente e a comunicação entre serviços por meio de redes internas.

O serviço de banco de dados utiliza o CouchDB, responsável por armazenar os documentos relacionados às portarias. O backend da aplicação é representado pelo Python Updater, um script desenvolvido em Python que realiza o download, processamento e carregamento dos dados no banco de dados. Por fim, o Frontend, desenvolvido em Vue.js, fornece a interface do usuário para consulta e interação com as informações.

Cada serviço possui um arquivo Dockerfile que define o processo de construção da imagem. O Dockerfile do Frontend utiliza uma imagem base do Node.js para instalação das dependências e construção dos arquivos estáticos, que são posteriormente servidos por uma imagem de produção do Nginx. O Dockerfile do Python Updater utiliza uma imagem base do Python para instalação das bibliotecas necessárias e execução do script de atualização. Já o Dockerfile do CouchDB configura o banco de dados utilizando variáveis de ambiente para definir credenciais e portas de comunicação.

A orquestração dos contêineres é realizada por meio do Docker Compose, através de um arquivo de configuração que descreve como os serviços devem ser iniciados, as redes que utilizam, os volumes persistentes e as dependências entre eles. A comunicação entre os contêineres é feita por meio de uma rede Docker criada especificamente para a aplicação, permitindo que os serviços se reconheçam pelo nome.

Para garantir o funcionamento adequado, foram definidos healthchecks que monitoram a integridade dos serviços. O Traefik, um proxy reverso, foi configurado para realizar o roteamento do tráfego para o Frontend, utilizando labels no arquivo de configuração do Docker Compose.

O processo de execução da aplicação é iniciado pela construção das imagens Docker com o comando `docker-compose build`. Em seguida, a criação e execução dos contêineres ocorre por meio do comando `docker-compose up`, respeitando a ordem de inicialização definida nas dependências. O Python Updater se conecta ao CouchDB para armazenar os dados processados, enquanto o Frontend acessa o banco de dados para exibir as informações ao usuário.

Essa abordagem assegura a integração eficiente entre os componentes da aplicação, proporcionando uma solução robusta e de fácil manutenção.

IV. Resultados

Como resultado temos a primeira versão do sistema que está em funcionamento de forma interna no MPC-Pa. A figura 2 mostra a tela do sistema. Também foi aplicado um questionário (https://forms.gle/enuytRUfHv1Aj5ob6) para se obter as informações do usuário ao utilizar o sistema e foram obtidos as seguintes respostas:

A. Usabilidade do Sistema

Intuitividade e Navegação: 26 entrevistados consideraram o sistema intuitivo e de fácil navegação.

Dificuldades encontradas: 9 entrevistados relataram dificuldades com funcionalidades específicas, sendo as principais dificuldades relacionadas à filtragem avançada e à exportação de resultados.

Facilidade de localização de informações: A média da avaliação foi de 4,2 na escala de 1 a 5.

B. Funcionalidade do Sistema

Adequabilidade das funcionalidades de busca: 23 entrevistados afirmaram que as funcionalidades atendem às necessidades.

Precisão dos Resultados: 25 entrevistados indicaram que os resultados são precisos e relevantes.

Relevância dos resultados: 5 entrevistados encontraram resultados irrelevantes, sendo os principais problemas relacionados à exibição de documentos desatualizados.

C. Desempenho do Sistema

Velocidade de resposta: 22 entrevistados consideraram o sistema rápido, enquanto 8 mencionaram lentidão em horários de pico.

Ocorrência de erros e falhas: 8 entrevistados relataram experiências com erros ou falhas, principalmente ao carregar documentos extensos.

Sugestões de melhorias: As principais sugestões incluíram melhorias na busca por palavras-chave, otimização de carregamento de arquivos e personalização dos filtros de pesquisa.

V. Limitações

O projeto possui algumas limitações. O número de pessoas que avaliaram o sistema por meio do questionário foi muito abaixo para o número de servidores ativos no órgão. Também existem algumas melhorias, bem como:

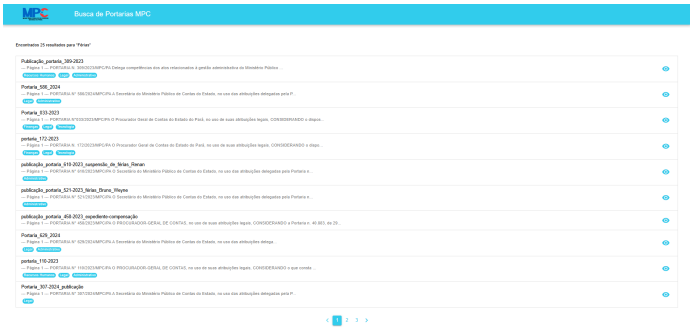

Fig. 2. Tela inicial do sistema.

- Inteligência Limitada: A busca e a organização dos documentos não são muito inteligentes, pois dependem de palavras-chave e precisam de aprimoramento na aplicação do Langchain.
- Lentidão: O sistema pode ficar lento com muitos documentos ou muitos usuários acessando ao mesmo tempo.
- Erros nos Textos: A extração de texto dos PDFs nem sempre é perfeita, o que pode levar a erros na busca e na organização.

VI. Conclusão

Em conclusão, a aplicação de busca de portarias representa um avanço significativo na gestão e acesso à informação administrativa, oferecendo uma alternativa mais eficiente e organizada aos métodos tradicionais baseados em documentos PDF. Apesar das limitações impostas pelo ambiente WebContainer, o sistema demonstra o potencial da tecnologia para otimizar processos e promover a transparência. A implementação de técnicas de extração de texto, simulação de PLN e armazenamento em um banco de dados escalável como o CouchDB, mesmo que de forma simplificada, permite aos usuários encontrar documentos relevantes de forma mais rápida e precisa.

Para aprimorar o sistema de busca e gerenciamento de portarias e superar as limitações identificadas, os seguintes trabalhos futuros são propostos:

VII. Trabalhos futuros

Para aprimorar o sistema de busca e gerenciamento de portarias e superar as limitações identificadas, os seguintes trabalhos futuros são propostos:

A. Migração para um Ambiente com Mais Recursos

A migração para um ambiente computacional mais robusto, que permita a instalação e a execução de bibliotecas externas, é fundamental para a implementação de técnicas de Processamento de Linguagem Natural (PLN) mais avançadas.

B. Utilização de um Banco de Dados Vetorial

Integrar um banco de dados vetorial especializado, como Chroma ou FAISS, para armazenar e pesquisar os embeddings de forma eficiente.

C. Aprimoramento da Classificação de Tópicos com Aprendizado de Máquina

Substituir a abordagem baseada em palavras-chave por modelos de aprendizado de máquina supervisionados ou não supervisionados para classificar os documentos em tópicos de forma mais precisa e automatizada.

D. Implementação de Técnicas de Extração de Informação Mais Robustas

Utilizar técnicas de extração de informação mais avançadas para identificar e extrair automaticamente metadados relevantes dos documentos, como datas, nomes de pessoas e organizações, e referências a leis e regulamentos.

E. Desenvolvimento de um Sistema de Resposta a Perguntas

Implementar um sistema de resposta a perguntas que permita aos usuários fazer perguntas em linguagem natural sobre o conteúdo das portarias e receber respostas precisas e concisas.

References

[1] DESORDI, D.; BONA, C. D. A inteligência artificial e a eficiência na administração pública. Revista de Direito, [S. l.], v. 12, n. 02, p. 01–22, 2020. DOI: 10.32361/202012029112. Disponível em: https://beta.periodicos.ufv.br/revistadir/article/view/9112. Acesso em: 17 fev. 2025.

[2] BREGA, José Fernando Ferreira. Governo eletrônico e direito administrativo. 2012. 336 f. Tese (Doutorado em Direito do Estado) - Faculdade de Direito, Universidade de São Paulo, São Paulo, 2012. Disponível em: http://www.teses.usp.br/teses/disponiveis/2/2134/tde-06062013-154559/publico/TESE_FINAL_Jose_Fernando_Ferreira_Brega.pdf. Acesso em 17 fev 2025.

[3] BRASIL. Emenda Constitucional n° 19, de 1998. Modifica o regime e dispõe sobre princípio e normas da Administração Pública, Servidores e Agentes políticos, controle de despesas e finanças públicas e custeio de atividades a cargo do Distrito Federal, e dá outras providências. Exposição dos Motivos. Disponível em: https://www2.camara.leg.br/legin/fed/emecon/1998/emendaconstitucional-19-4-junho-1998-372816-exposicaodemotivos-148914-pl.html. Acesso: 17 fev 2025.

[4] CAMPOS, S.L.B; FIGUEIREDO, J.M.. Aplicação de Inteligência Artificial no Ciclo de Políticas Públicas. Cadernos de Prospecção. V.15, p. 196-214, 2022.

[5] MENEZES, Ana Paula Veras Carvalho. Inteligência artificial para identificação de indícios de fraude e corrupção em compras públicas no TCU. 2022. 120 f. Dissertação (Mestrado em Administração Pública) - Instituto Brasileiro de Ensino, Desenvolvimento e Pesquisa, Brasília, 2021.

[6] MCCARTHY, John. WHAT IS ARTIFICIAL INTELLIGENCE?: Basic Questions. 2007. Acesso em: 25 fev. 2025.

[7] RUSSELL, Stuart; NORVIG, Peter. Inteligência Artificial. 3. ed. Rio de Janeiro: Elsevier, 2013.

[8] J. Kaddour, J. Harris, M. Mozes, H. Bradley, R. Raileanu, and R. McHardy. Challenges and applications of large language models, 2023.

[9] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding, 2019

[10] A. Radford, K. Narasimhan, T. Salimans, and I. Sutskever. Improving language understanding by generative pre-training. 2018.

[11] LANGCHAIN. LangChain is a framework for developing applications powered by language models. 2024. Disponível em:https://python.langchain.com/docs/get_started/introduction.

[12] Raho, M. et al. Kvm, xen and docker: A performance analysis for arm based nfv and cloud computing. In: 2015 IEEE 3rd Workshop on Advances in Information, Electronic and Electrical Engineering (AIEEE). [S.l.: s.n.], 2015. p. 1–8. 24

[13] DI PIETRO, Maria Sylvia Zanella. Direito Administrativo. São Paulo: Atlas, 2002.

[14] MEIRELLES, Hely Lopes. Direito Administrativo Brasileiro. São Paulo: Malheiros, 1996.