Web Crawler para portais da área do Direito*

*Projeto de TFG do curso de Sistemas de Informação e Projeto de Pesquisa Bolsa PROBIT-UFN.

Eduardo Pavani Palharini
Curso de Sistemas de Informação
Universidade Franciscana
Santa Maria, Brasil
e.pavani@ufn.edu.br

Alexandre de Oliveira Zamberlan
Curso de Sistemas de Informação
Universidade Franciscana
Santa Maria, Brasil
alexz@ufn.edu.br

***Resumo*—Este artigo apresenta o projeto e a construção de um sistema automatizado de rastreamento de conteúdo em sites e/ou portais jurídicos. O sistema utiliza um *crawler* (bot de pesquisa) para coletar metadados, como título, descrição e data de publicação. Ele foi desenvolvido para facilitar a busca de informações por profissionais do Direito nos Tribunais de Justiça do Brasil. O projeto empregou a metodologia Scrum juntamente da técnica Kanban para gestão de atividades, a linguagem Python, os frameworks Django, Bootstrap e Scrapy, a biblioteca Beautiful Soup e a API Selenium. Como resultado há um sistema piloto disponível em https://buscadir.lapinf.ufn.edu.br/ com gestão de usuários, áreas do Direito, fontes de pesquisa, instituições e buscas.**

***Index Terms*—Beautiful Soup; Django; Scrapy; Selenium; Web Crawler;**

I. Introdução

Um rastreador da *Web* (Web Crawler) é um software robô denominado *bot* que navega sistematicamente em servidores *Web*, com a finalidade de recuperar e indexar conteúdos (páginas), a partir de critérios de pesquisa informados em mecanismos de busca. Esses mecanismos de pesquisa que são associados a rastreadores, copiam (baixam) as páginas de interesse para serem processadas localmente com maior relevância e eficiência [1]. Todo esse processo ocorre de forma autônoma e quando um rastreador entra em processamento, ele acessa páginas de servidores com critérios de busca, baixando e indexando dados de inúmeras páginas, como informações de Decisões judiciais divulgadas nos sites dos Tribunais.

Dessa forma, a pesquisa ou a busca passa ter um caráter de maior relevância, uma vez que pode elencar páginas mais precisas de conteúdo da pesquisa. Um rastreador da *Web* trabalha com uma lista de URL (endereços *Web* em navegadores) a serem visitados (neste contexto, sites de Tribunais Federais). Esses URL são chamados de sementes e à medida que o rastreador visita esses URL, ele identifica todos os hiperlinks nas páginas da *Web* recuperadas e os adiciona à lista de URL a serem visitadas. Essa lista de URL comporta-se de acordo com um conjunto de políticas e se o rastreador estiver realizando o arquivamento, ele copia e salva as informações à medida que avança. Os arquivos são geralmente armazenados de forma que possam ser visualizados, lidos e navegados localmente [1].

A. Objetivo

Projetar, desenvolver e implantar um sistema *Web* para realizar buscas relevantes em sites e/ou portais jurídicos.

Para que o objetivo geral seja alcançado, identificaram-se alguns objetivos específicos:

- Entender, testar e aplicar Web Crawlers;
- Estudar e avaliar bibliotecas do universo Python que promovem buscas relevantes;
- Mapear e compilar trabalhos relacionados que usaram técnicas de rastreamento em portais;
- Estudar e testar mecanismos de CAPTCHA; Estudar e testar APIs de proteção de portais que bloqueiam sistemas *bots*.

B. Justificativa

A área do Direito da instituição vem trabalhando com questões que envolvem Tecnologia da Informação, em especial a Inteligência Artificial. São os projetos:

1) Técnicas de Inteligência Artificial Aplicadas ao Direito: Representação de Conhecimento e Raciocínio Automatizado [2];

2) O Direito da Criança e do Adolescente e a Experiência de Extensão Universitária Durante a Pandemia: [3].

Os projetos possuem fases que necessitam de buscas relevantes em sites específicos, como Tribunais de Justiça. Até então, toda a busca é realizada manualmente, site a site, palavra-chave por palavra-chave. Assim, automatizar o processo é importante, além de qualificar o resultado da busca, justificando portanto este projeto. Além disso, o sistema proposto obedeceu à estrutura e à dinâmica presentes em sistemas já construídos (SISGEP-COMIC, SISGEP-SADEPI, TFGOnline, Residencia On-line), todos parte do Laboratório de Práticas da Computação e registrados formalmente junto ao INPI. Destaca-se, também, que a empresa parceira, ER Sistemas, como nos outros sistemas, também auxiliou no desenvolvimento, principalmente nas questões de disponibilizar

na Internet com total segurança os dados da base, além de fornecer boas práticas de desenvolvimento de software.

Finalmente, para auxiliar na compreensão da proposta, o texto está dividido em 5 seções. A seção de Revisão bibliográfica apresenta conceitos, fundamentos e processos referentes ao foco do trabalho (Web Crawler), dos recursos tecnológicos necessários e de trabalhos relacionados. A seção Metodologia aborda como a pesquisa é conduzida. Na sequência, Resultados, Conclusões e Referências Bibliográficas.

II. Revisão bibliográfica

Esta seção trata dos principais fundamentos e processos trabalhados no texto. Inicialmente, apresenta os temas foco da pesquisa e na sequência as tecnologias necessárias para a construção da solução.

A. Tribunais de Justiça

O Poder Judiciário Brasileiro é formado por Justiça Comum e Justiça Especializada [4].

- Justiça Comum é mais ampla e abrange a maioria das demandas processuais existentes: cível, criminal, de família, do consumidor, de sucessões, de falências e recuperações judiciais, da infância e juventude, entre outras. É composta pelas justiças Federal e Estadual, possuindo tribunais regionais e juizados federais;
- Justiça Especializada é responsável por tratar de assuntos específicos das três áreas: trabalhista, militar e eleitoral. Todas essas demandas específicas são tratadas por tribunais próprios vinculados ao Tribunal Superior do Trabalho, Superior Tribunal Militar e Tribunal Superior Eleitoral, suas instâncias máximas [5].

1) Tribunais Federais: Os tribunais federais de justiça são responsáveis por julgar casos de natureza federal, como Constituição Federal, leis federais, tratados internacionais, entre outros assuntos de competência exclusiva da União [6]. Os tribunais federais mais importantes do Brasil são:

- Supremo Tribunal Federal (STF): Mais alto órgão do poder judiciário brasileiro e tem a função de proteger a Constituição Federal. O STF é responsável por julgar questões constitucionais e atua como última instância para decisões judiciais;
- Superior Tribunal de Justiça (STJ): Órgão responsável por uniformizar a interpretação das leis federais no território brasileiro. O STJ julga principalmente questões infraconstitucionais, como direito civil, direito penal e direito administrativo;
- Tribunal Regional Federal (TRF): Existem cinco TRF, cada um responsável por uma região geográfica do Brasil. Eles são responsáveis por julgar os recursos provenientes de decisões proferidas pelos juízes federais de primeira instância;
- Justiça Federal de Primeira Instância: São os juízes federais de primeira instância que atuam em varas federais e julgam os casos de competência federal dentro de suas respectivas regiões.

2) Tribunais Estaduais: Os tribunais estaduais de justiça são responsáveis pelo julgamento de casos de competência estadual, ou seja, questões relacionadas às leis estaduais e municipais, assim como ações cíveis e criminais que ocorrem dentro do âmbito dos estados brasileiros [6]. Os principais tribunais estaduais são:

- Tribunal de Justiça (TJ): Cada estado brasileiro (incluindo a unidade federativa do Distrito Federal) possui um Tribunal de Justiça, responsável pelo julgamento de casos de competência estadual. O TJ é a instância final para a maioria dos processos que tramitam dentro do estado;
- Justiça Estadual de Primeira Instância: São os juízes de direito de primeira instância que atuam nas comarcas e julgam casos de competência estadual dentro de suas respectivas áreas de jurisdição.

B. Acórdãos

Acórdãos jurídicos são decisões colegiadas proferidas por tribunais, tanto federais quanto estaduais, em resposta a recursos ou ações que são submetidas a eles. Essas decisões representam o entendimento do tribunal sobre a matéria discutida e têm o objetivo de estabelecer precedentes jurídicos para casos futuros semelhantes [7].

Os acórdãos são elaborados por um grupo de juízes ou desembargadores que compõem o órgão colegiado do tribunal. Geralmente, um relator é designado para analisar o caso e elaborar um relatório sobre os fatos, argumentos apresentados pelas partes e fundamentos legais aplicáveis. Em seguida, os demais membros do tribunal participam da discussão e da votação, expressando seus votos e fundamentando suas posições.

Os acórdãos contêm a decisão final do tribunal, que pode ser a confirmação ou a modificação de uma sentença proferida em primeira instância. Além disso, trazem os fundamentos jurídicos que embasaram a decisão, destacando os pontos relevantes do caso, a interpretação da lei aplicada e a motivação para o resultado alcançado [8].

Esses documentos têm grande importância no sistema jurídico, pois estabelecem a jurisprudência, ou seja, a interpretação uniforme do direito aplicada pelos tribunais. Os acórdãos servem como precedentes para orientar futuras decisões de juízes e tribunais, proporcionando maior segurança jurídica e coerência na aplicação das leis. Podem, também, ser acessados publicamente nos tribunais de justiça (TJ) e são fontes de consulta para advogados, estudantes de direito, pesquisadores e demais interessados em compreender a jurisprudência e o entendimento dos tribunais sobre determinadas questões jurídicas. Esse, inclusive, é o foco principal do artigo: trabalhar com pesquisa automatizada de acórdãos.

C. Web Crawlers

Web Crawler [1], também conhecido como Web Spider, é um robô automatizado de busca, organização e armazenamento de dados provenientes da Internet, ou seja, tem comportamento de rastrear e obter dados por meio de sistemas digitais automatizados. É, também, sinônimo de Web Scraper, porém

este apresenta métodos mais rebuscados de busca que o Web Crawler. Um Web Crawler é semelhante aos buscadores Google (Googlebot), Microsoft (Bingbot), Baidu (Baidu Spider) e DuckDuckGo. Todavia, tem por objetivo atender um nicho de busca (fontes de consulta) que o usuário selecionará - entre as opções disponíveis no sistema - previamente.

Diferente do mundo real, procurar e catalogar a *Web* é um processo com certa complexidade (dependente de recurso computacional) e que muitas vezes deixa informação relevante fora dos resultados da busca. Os mecanismos de pesquisa visam rastrear determinados sites e/ou palavras-chaves para refinar o processo e deixá-lo direcionado as suas necessidades. As áreas que mais se beneficiam de Web Crawlers é pesquisa científica, mercado de ações e marketing digital [9].

A principal atribuição de um *bot* de pesquisa é o processo chamado de indexação. Nele, toda informação que foi encontrada pelo sistema é catalogada e armazenada em uma base de dados (seja ela banco de dados, ou arquivos), gerando assim um conjunto de metadados de busca [9].

D. API, Bibliotecas e Frameworks

Application Programming Interface são interfaces padronizadas que permitem a integração entre diferentes plataformas, definições e/ou protocolos. Uma forma de compreender API é como se fosse um contrato entre duas partes. Nesse contrato são ditas as regras (normas e protocolos) que irão permitir a troca de informações entre os sistemas. Normalmente, a aplicação interessada em ter a integração entre sistemas é a responsável por se padronizar e adequar respeitando os critérios estabelecidos pela outra parte [10].

No sistema proposto, API para a *Web* são utilizados, uma vez que devem processar dados, regras e protocolos entre servidor e um navegador. Registra-se, porém, que todos os serviços da *Web* são API, mas nem todas as API são serviços da *Web* [10].

E. Frameworks

Conhecidos como arcabouços ou esqueletos de sistemas, são coleções de funcionalidades genéricas para reutilização de código e partes do sistema em desenvolvimento. Isso ocorre por meio de pacotes, classes, interfaces e métodos, por exemplo. Ao contrário das bibliotecas, é o *framework* que assume o fluxo de controle da aplicação. Entre as principais vantagens de se utilizar *frameworks* estão: facilidade para detecção de erros, concentração na abstração do problema e otimização de recursos [11].

Desenvolver um *Web Crawler* não é uma tarefa trivial, porém, existem ferramentas que auxiliam no processo de criação de um *bot* automatizado, como:

- Beautiful Soap: biblioteca Python de código aberto e gratuito que extrai dados de arquivos HTML e XML (utilizados em arquivos de sites). Considerada fácil de integrar é utilizada por desenvolvedores iniciantes. Ela cria sua própria estrutura do site analisado, com base em critérios específicos, configuráveis pelo desenvolvedor, que podem ser usados para extrair, navegar, pesquisar e modificar dados do HTML. Além disso, com poucos comandos é possível exportar os dados para um arquivo no formato CSV ou JSON (utilizados para base de dados de sistemas ou sites) [12];
- Scrapy: *framework* de código aberto e gratuito desenvolvido na distribuição Anaconda da linguagem Python para extrair dados da *Web*. Conta com uma estrutura de organização que pode interagir com várias bibliotecas da mesma linguagem, inclusive com ferramentas de processamento e mineração de dados tornando a pesquisa mais eficiente e direcionada ao interesse do projeto. Tem como vantagem utilizar requisições assíncronas tratando eventuais erros sem interromper a execução [13];
- Selenium: API e biblioteca gratuita e de código aberto disponível em várias linguagens, inclusive Python. O Selenium é frequentemente utilizado para testar a funcionalidade de um site, automatizando a interação com os elementos da interface, preenchendo formulários, clicando em botões, navegando por páginas e verificando os resultados. Para utilizá-lo no Python é preciso instalar o pacote, através do *pip*, e fazer o download do *driver* específico do navegador que deseja controlar, como o ChromeDriver para o Google Chrome ou o GeckoDriver para o Firefox. O *driver* é responsável por estabelecer a comunicação entre a aplicação que contém a API Selenium e o navegador (motor de busca). Com os campos identificados previamente pelo Beautiful Soup e/ou Scrapy o Selenium injeta os dados fornecidos pelo usuário nos campos desejados. [14].

Registra-se que neste trabalho, foram utilizados a biblioteca Beautiful Soap, o *framework* Scrapy e a API Selenium, porque trabalham e têm integração com vários recursos importantes para o projeto da linguagem Python, além de terem uma documentação bem elaborada e estrutura de código mais amigável.

F. Sistema CAPTCHA

É o Teste de Turing público automatizado para distinguir entre computadores e pessoas, isto é, um mecanismo de segurança que desafia o usuário a preencher um campo com a sequência formada por letras e números desenhados em uma imagem aleatória. Tem por objetivo proteger sistemas contra robôs automatizados, como *Web* Crawlers, evitando quebra de criptografia e SPAM.

Atualmente versões mais modernas do CAPTCHA apresentam imagens quadriculadas sobre coisas, paisagens e objetos que instigam o usuário a identificar onde aparecem determinados itens, selecionar e validar para então acessar o sistema ou funcionalidade. Existem diferentes CAPTCHAs, desenvolvidos por diferentes empresas. O mais famoso e utilizado por desenvolvedores de sistemas é o reCAPTCHA da Google. Sua licença é gratuita e basta você ter uma conta de e-mail do Google, para ele entrar em funcionamento, com alguns poucos comandos incorporados ao *front-end* do sistema.

G. Python e Django

Python é uma linguagem de programação de alto nível, gratuita, interpretada, multiplataforma, tipada dinamicamente e orientada a objetos. É uma das linguagens mais demandas por empresas e organizações do mundo inteiro por ser simples de aprender e aplicar, ter uma grande comunidade engajada e contar com muitas bibliotecas e *frameworks* [15].

Django é um *framework* gratuito para desenvolvimento rápido e seguro de aplicações na *Web* escrito em Python. Fornece inúmeros modelos de *design*, *drivers* de conexão com banco de dados, de processos de validação (como *login* e senhas), de aplicativos com CRUD *(Create, Retrieve, Update, Delete)* completo. Trabalha no modelo MVT (Model-View-Template):

- *Model*: responsável pelo mapeamento do banco de dados via classes e objetos do sistema;
- *Template*: interface (página HTML - visual) que o usuário vai interagir com a aplicação;
- *View*: parte lógica do sistema, onde se especifica a interação da parte visual com o modelo de dados, ou seja, as regras do negócio.

Além dessas características, Django inclui vários recursos e funcionalidades prontas que são facilmente incorporadas à aplicação, reduzindo consideravelmente o tempo de desenvolvimento e as linhas de código [16].

H. Github

Github é uma ferramenta, da Microsoft do tipo Git (sistema de controle de versões distribuído), de apoio aos desenvolvedores, analistas de sistemas, engenheiros de *software* e demais profissionais da área de TI. Trata-se de um *Cloud Service* que hospeda um sistema de versionamento de aplicações colaborativas e tem compatibilidade com sistemas Linux, Windows e MacOS. Ou seja, armazena trabalhos e sistemas que estão em evolução, constantemente sendo atualizados. O versionamento deve manter histórico de desenvolvimento dos códigos-fontes e também da documentação produzida, controlando conflitos, produtividade, status das versões, entre muitos outros [17].

Suas principais qualidades são a segurança e a inovação. Entre as principais características é possível destacar a possibilidade de retornar o sistema para uma versão estável caso haja problema, por exemplo. Paralelamente a isso, novas soluções podem continuar sendo implementadas independentemente das alterações salvas no repositório principal.

I. SQLite

Considerado um Sistema de Gerenciamento de Banco de Dados relacional, é conhecido por sua leveza, simplicidade, portabilidade e eficiência, além de não requerer uma configuração complexa ou um servidor separado para funcionar. Além disso, vem associado ao *framework* Django para testes em pilotos construídos em computadores locais. Dessa forma, é utilizado em aplicativos e dispositivos que precisam de um banco de dados local, como aplicativos móveis, navegadores da web e software embarcado [18]. Ao contrário de bancos de dados tradicionais, o SQLite armazena todo o banco de dados em um único arquivo, que pode ser facilmente movido ou compartilhado entre sistemas.

Apesar de ser um banco de dados mais simples, o SQLite oferece recursos consistentes, como suporte transacional, integridade referencial, índices, criptografia, disparadores (*triggers*) e visões (*views*). Ele é distribuído como um software de código aberto e está disponível para várias linguagens de programação [18].

J. NGINX e GUNICORN

O paradigma de computação em nuvem pode ser definido como um conjunto de recursos com capacidade de processamento, armazenamento, conectividade, plataformas, aplicações e serviços disponibilizados na Internet. Uma arquitetura em nuvem é formada por múltiplos servidores. Entre os vários conceitos que abrangem a computação em nuvem destacam-se balanceamento de carga, banco de dados aglomerados e escalonamento de armazenamento. Nesse quesito de balanceamento de carga, surge então o serviço NGINX, que faz a distribuição de requisições de maneira equitativa sobre os nós de um ambiente distribuído [19]. Em síntese, o NGINX é em uma aplicação *Web* o responsável por gerenciar a disponibilidade da aplicação ao usuário.

Além disso, esse serviço oferece certificado digital SSL (Secure Sockets Layer), permitindo que o tráfego entre o cliente final e o servidor seja utilizando conexão segura através do protocolo HTTPS (Hypertext Transfer Protocol Secure). O NGINX também foi projetado para servir e fazer cacheamento de arquivos estáticos como imagens, CSS (Cascading Style Sheets) e Javascript, além de servir o conteúdo dinâmico encaminhado pelo GUNICORN.

Por sua vez, GUNICORN é considerado um servidor Python para páginas e/ou sistemas via protocolo HTTP para sistemas UNIX (Linux). Ou seja, ele resolve de forma estática e dinâmica as solicitações HTTP de clientes via seus navegadores. Esse serviço é compatível com várias estruturas *Web*, com uso otimizado de recursos [19].

K. Scrum

Na indústria de *software*, há uma variedade de metodologias e técnicas empregadas para a construção de programas computacionais. Dentre as metodologias ágeis aplicadas, uma delas é a Scrum. Essa metodologia tem por prioridade o desenvolvimento de produto/serviço de modo menos complexo, mais eficiente e focado em resultados [20].

Scrum começa com o *Product Owner* ou (Dono do Produto), ele é o agente responsável que determina o que deve e não deve fazer parte do produto final. Além de definir os requisitos do projeto, o *Product Owner* elabora uma lista com todas as exigências e implementações que o produto/serviço final deve apresentar, o que se chama de *Product Backlog*. O *Product Backlog* necessita obedecer uma ordem de prioridades, por exemplo, em um sistema *cloud service* (serviço em nuvem) é extremamente importante criar um protótipo local antes de levar o sistema ao servidor e disponibilizar à publico. Após definir-se quais são os requisitos que devem fazer parte do

Product Backlog, a equipe de desenvolvimento planeja os *Sprints*, que são períodos de tempo pré-determinados, que podem ser mensais ou semanais, onde se divide as tarefas presentes no *Product Backlog* em pequenas tarefas que são distribuídas aos membros da equipe para serem concluídas dentro de 7 ou 15 dias, por exemplo. Após o término de cada *Sprint*, é realizada um retrospectiva para analisar e discutir o que pode ser melhorado no próximo.

Há também na metodologia Scrum a figura do *Scrum Master*. Esse profissional é responsável por supervisionar o trabalho do restante da equipe de desenvolvimento do projeto, além de definir a quantidade de *Sprints* que devem ser realizados durante o mês ou semana e gerenciar o desenvolvimento como um todo (prazos, metas, responsáveis, controle de versões, *backup* do sistema, etc.). Também é função dele facilitar quaisquer problemas que possam surgir no caminho da equipe de trabalho e assegurar que os métodos e princípios da metodologia sejam mantidos e cumpridos. E por fim, existe o *Stakeholder* que são todas as partes interessadas no projeto. Responsáveis por validar o sistema normalmente são clientes.

L. Trabalhos Relacionados

O trabalho de [21] apresenta um ambiente online construído e disponibilizado para a gestão de Trabalhos Finais de Graduação da Universidade Franciscana (UFN). Mostra funcionalidades das diversas visões e cenários de usuários, bem como as boas práticas utilizadas tanto na gestão do projeto, quanto na implementação do sistema. A metodologia Scrum foi utilizada com o suporte da técnica Kanban para gestão de atividades, bem como as tecnologias *frameworks* Bootstrap e Django que estão associados à linguagem Python para desenvolvimento *Web* do sistema. O principal objetivo do projeto foi agilizar os processos de submissão, avaliação e emissão de pareceres, pois há vários documentos a entregar, prazos a serem cumpridos, avaliações obedecendo critérios e *feedbacks* obrigatórios aos alunos. Ademais, promove mecanismos de busca de assuntos e/ou tecnologias trabalhadas em pesquisas conduzidas por alunos e seus orientadores.

No trabalho de [22], para tornar a extração dos dados de uma plataforma *Web* mais ágil, bem como garantir pertinência nos dados, fora desenvolvido em Python, com *framework* Django, um sistema automatizado de busca de informações (Web Crawler) de indicadores institucionais. Para gestão dos processos e atividades foi utilizado a metodologia Scrum juntamente da ferramenta Trello (Kanban). A construção de um sistema para a extração desses dados fora proposta de duas maneiras: com acesso direto a base de dados da plataforma em questão - através de API - ou ao próprio arquivo XML dos usuários, realizando o processamento das informações através de um Parser *C#* (analisador), armazenando-as em uma base de dados MySQL, e posteriormente, apresentando essas informações em uma interface.

No Trabalho Final de Graduação de [9], o enfoque foi utilizar os benefícios do Web Scraper para chegar a uma solução viável para a indexação de artigos científicos e seus atores publicados na biblioteca IEEE Xplore. Para isso, foi desenvolvido um *script* em Python, juntamente da ferramenta *crawler* Selenium, que acessa a página da IEEE e extrai dois arquivos de formato CSV. Um arquivo sobre os artigos com as informações de título, idioma, resumo, número de páginas e palavras-chaves. E o outro arquivo sobre os autores com as informações de nome, afiliação e país.

Os três trabalhos citados utilizaram a linguagem de programação Python para construção dos sistemas. sendo que cada um apresenta suas peculiaridades no desenvolvimento, com uso de diferentes ferramentas para diferentes propósitos. Todavia, esses trabalhos foram importantes para dar o embasamento necessário para desenvolver e nortear o projeto Web Crawler do Direito, uma vez que todos estão no universo Python de programação, com suas bibliotecas e seus *frameworks*.

III. Metodologia e proposta de trabalho

Este trabalho é baseado em pesquisa exploratória com desenvolvimento de piloto computacional, tendo como referência os portais de Tribunais (acórdãos jurídicos ou sentenças). No projeto e no desenvolvimento do Web Crawler foi utilizado um conjunto de boas práticas do Scrum. Os encontros entre aluno e orientador, ao longo do semestre, foram semanais, seguindo orientações da metodologia, e sempre houve apresentação, e posteriormente discussão, de atividades desenvolvidas. As atividades *Product Backlog* foram geridas pela ferramenta *Web* Trello (técnica Kanban) para controle de prazos.

Vale ressaltar que a equipe de desenvolvimento é formada unicamente pelo aluno e professor orientador. O aluno, além de desenvolvedor do sistema, comportou-se como *Product Owner* e, neste caso, o professor orientador adquiriu o papel de *Scrum Master*. O trabalho é, também, uma Bolsa de Iniciação Científica da Universidade Franciscana - UFN (Probit 2022/2023) em parceria com o Curso de Direito da instituição. Os professores da área do Direito, que colaboraram com o projeto, são os *Stakeholders* e a participação deles foi imprescindível para o trabalho, fornecendo o conteúdo relacionado à área do Direito para a pesquisa e, posteriormente, validando funcionalidades, *layouts* e dinâmica de funcionamento do *software* desenvolvido.

Para elucidação do sistema, foi utilizada a ferramenta Astah de diagramação UML. Quanto as tecnologias computacionais foram e estão sendo usadas:

- linguagem de programação Python;
- *frameworks* Bootstrap, Django e Scrapy;
- biblioteca Beautiful Soup;
- API Selenium;
- Sistema Gerenciador de Banco de Dados(SGBD) SQLite;
- GUNICORN e NGINX;
- Github.

A. Ideia do sistema

Nesta seção, buscou-se apresentar especificações de alguns aspectos funcionais e estruturais. A Figura 1 ilustra os atores do sistema e as principais funcionalidades que se buscou modelar. Ressalta-se que os casos de uso em destaque (vermelho)

não fazem parte deste trabalho, mas ideias futuras em uma nova versão do sistema. Na Figura há um conjunto de Casos de Uso (figuras elípticas), que são os principais serviços que o sistema deve realizar.

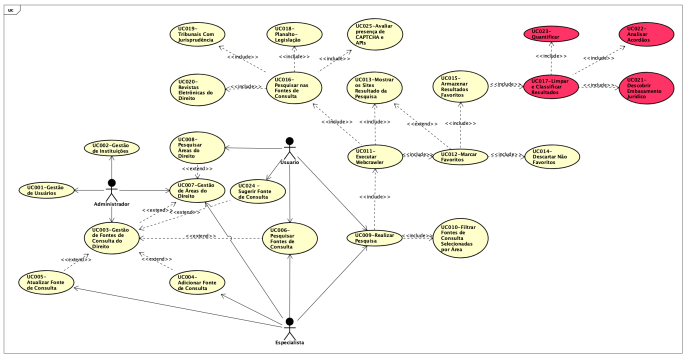

Figura 1. Diagrama de Casos de Uso.

O piloto desenvolvido, neste trabalho, possui apenas três atores: Administrador, Usuário e Especialista. O Administrador é o usuário responsável por toda a parte de gestão do sistema. Usuário e Especialista apresentam atribuições parecidas, porém o Especialista tem, a mais, a capacidade de Gerir Fontes do Direito e Gerir Áreas do Direito.

As Fontes do Direito são os portais *web* que o Especialista, designado por sua instituição, terá a atribuição de cadastrar. As Áreas do Direito, por sua vez, são diferentes esferas do Direito, como: Civil, Criminal, Sucessão, Ambiental, Previdenciária, etc. A principal funcionalidade do sistema para os profissionais do Direito é o ato de realizar a pesquisa, que é o foco do desenvolvimento.

Um diagrama importante, no quesito de aspectos funcionais, é o Diagrama de Atividades, que ilustra o fluxo de trabalho pretendido com o piloto proposto (Figura 2). Na figura, é possível acompanhar a dinâmica de funcionamento do sistema Web Crawler, em que um usuário do sistema, depois do *login*, dentro da gestão básica, pode realizar diferentes consultas como Áreas do Direito e Fontes de Consulta. Em seguida, esse usuário, no ambiente de pesquisa, pode filtrar áreas e executar o rastreador (*crawler*). Uma vez disparado o rastreador, o sistema busca nas fontes cadastradas se o site ou o repositório possui algum tipo de segurança, como o CAPTCHA. Se o site pesquisado possuir um mecanismo de proteção *bot* o sistema Web Crawler irá fazer a identificação no cadastro da fonte. Caso o site não possua mecanismo de segurança ou exista uma API válida, os resultados são devolvidos ao sistema, em forma de arquivos baixados que serão armazenados em um banco de dados.

Há dois processos considerados críticos:

- verificar se o site ou fonte tem recurso CAPTCHA;
- identificar os campos presentes no site ou fonte.

Assim, decidiu-se tratar esses processos no momento em que uma fonte ou site de pesquisa for cadastrado no sistema. Ao observar a Figura 6, há duas colunas que identificam se

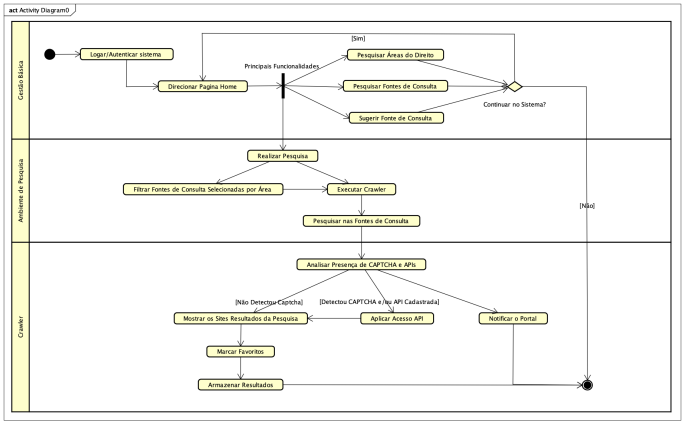

Figura 2. Diagrama de Atividade.

o site ou fonte tem CAPTCHA e se campos foram ou não identificados.

Quado um site indicar a presença de CAPTCHA, isso é sinalizado, como uma etiqueta, permitindo que o usuário saiba que essa fonte possui CAPTCHA e, consequentemente, evitando o uso desse site na pesquisa.

Uma vez definidas e especificadas as funcionalidades, segue a especificação de aspectos estruturais. Um diagrama importante para isso é o de Classe, mas com uma característica de pacote, ilustrando pacotes necessários, classes e suas relações (Figura 3). Na figura, existem três pacotes: Gestão Básica, Ambiente de Pesquisa e *Crawler*. O pacote Gestão Básica tem a responsabilidade de gerenciar Usuários, Instituições, Fontes de Consulta e Áreas do Direito, comumente chamado de *Create, Retrieve, Update, Delete* (CRUD). O pacote Ambiente de Pesquisa tem como objetivo tratar dos resultados baixados, favoritados e/ou armazenados. Por fim, o pacote *Crawler* é o principal foco desta pesquisa sendo implementadas técnicas de rastreamento digital.

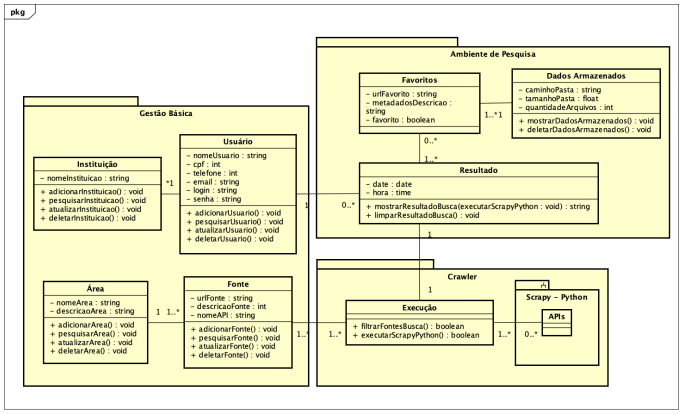

Figura 3. Diagrama de Classe.

B. Resultados e discussões

Os resultados do trabalho, referem-se a modelagem de todos aspectos funcionais e estruturais do sistema proposto, bem como a implementação e implantação das principais funcionalidades do sistema, caracterizando-o como um piloto ou um *Minimum Viable Product* (MVP)[1]. Os aspectos funcionais e estruturais podem ser visualizados em diagramas das Figuras 1, 2 e 3.

Concomitante ao processo de modelagem, protótipos foram construídos ao longo de todo o processo. O sistema já pode ser acessado pelo endereço https://buscadir.lapinf.ufn.edu.br. Há algumas funcionalidades básicas que foram implementadas e que podem ser visualizadas nas Figuras 4, 5 e 6.

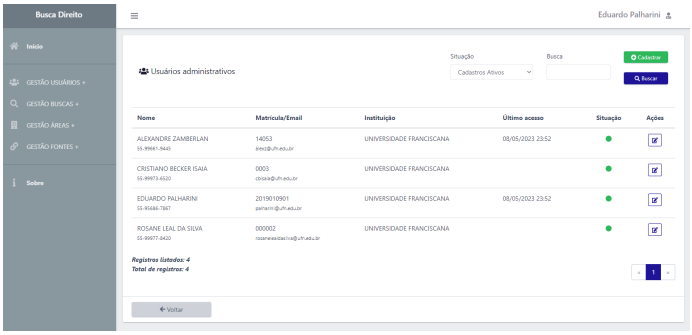

Figura 4. Gestão de Usuários.

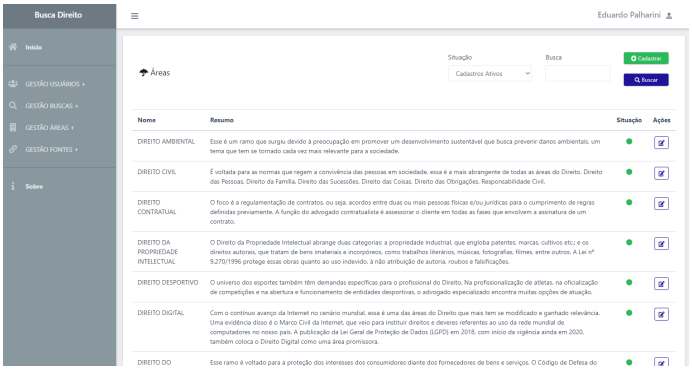

Figura 5. Gestão de Áreas do Direito.

Uma vez cadastrada uma fonte ou um site de Tribunal de Justiça, o sistema identifica a presença de CAPTCHA e de campos presentes para que o Web Crawler possa injetar dados na pesquisa. Portanto, as Figuras 7 e 8. Isso é importante, pois os sites dos Tribunais possuem campos de pesquisa e são eles que deve receber a injeção do Web Crawler. A Figura 9 ilustra um exemplo de pesquisa em Tribunal. E a Figura 10 mostra o código construído no processo de validar o formulário (antes de persistir no banco de dados).

[1]O Mínimo (ou Mais) Produto Viável (MVP), é uma versão de um novo produto que permite que uma equipe colete o máximo de aprendizado validado sobre os clientes com o mínimo de esforço [23].

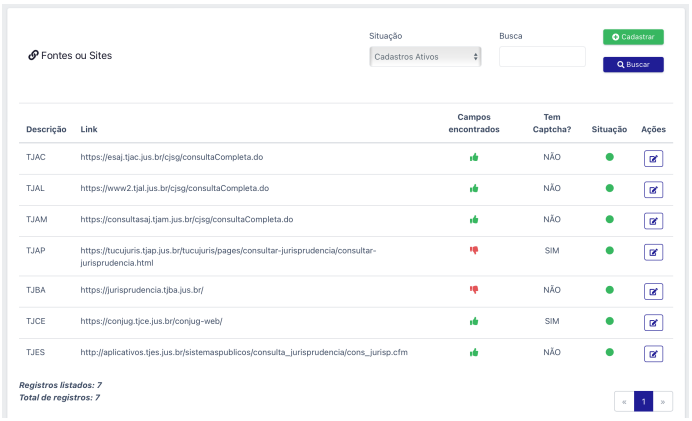

Figura 6. Gestão de Fontes no BuscaDir.

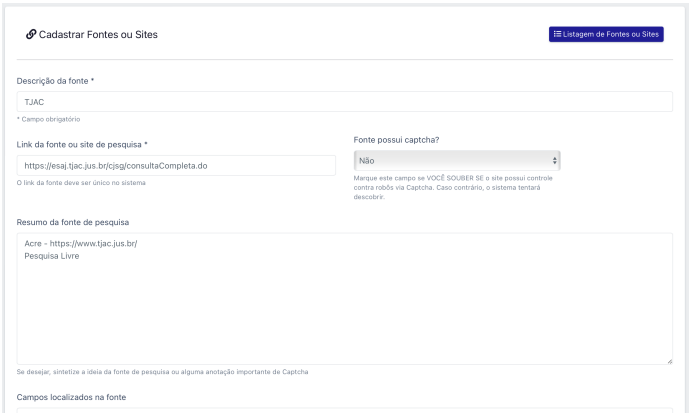

Figura 7. Cadastro da Fonte. Primeira Parte.

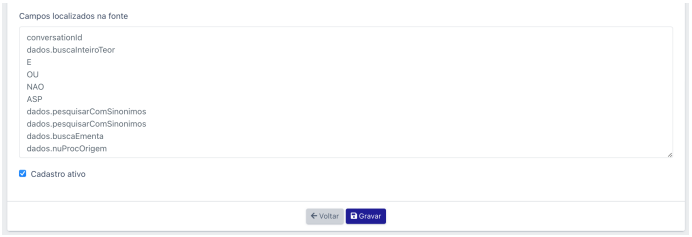

Figura 8. Cadastro da Fonte. Segunda Parte.

Entre as linhas 58 e 69 há a rotina para verificar a presença de CAPTCHA. E entre as linhas 71 e 87 há o uso da biblioteca Beautiful Soup que identifica a presença de *forms* no site e dentro desses *forms*, identifica *inputs*.

Uma das funcionalidades mais importantes é a Gestão de Consultas. É nela que o usuário do sistema vai poder pesquisar por palavras-chave nos sites ou fontes cadastrados no sistema e, depois, selecionar as fontes onde pretende que a pesquisa seja realizada. No piloto, foram cadastrados alguns Tribunais de Justiça do Brasil (Figura 12). Dessa forma, a Figura 11

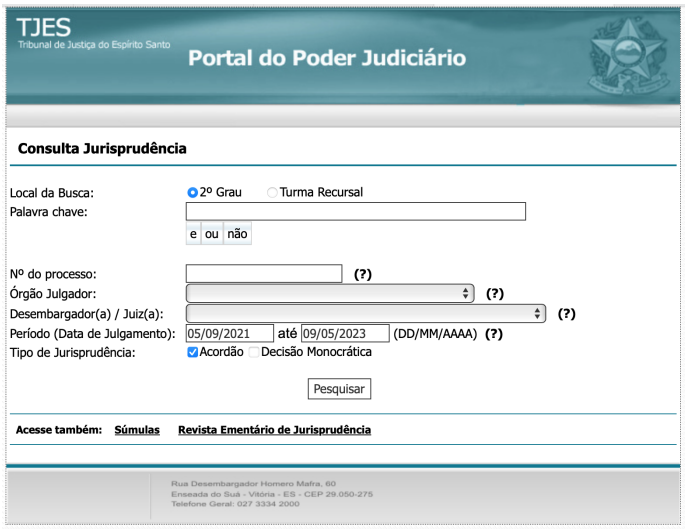

Figura 9. Site de pesquisa do Tribunal de Justiça do Espírito Santo.

```
56      def form_valid(self, form):
57          self.object = form.save(commit=False)
58          try:
59              requisicao = requests.get(self.object.link)
60
61              html = requisicao.text
62
63              if "captcha" in html.text:
64                  self.object.tem_captcha = 'SIM'
65              else:
66                  self.object.tem_captcha = 'NÃO'
67          except:
68              self.object.tem_captcha = 'NÃO SE SABE'
69              messages.error(self.request,"Não foi possível descobrir corretamente se o link possui Captcha"
70
71          try:
72              requisicao = requests.get(self.object.link)
73
74              html = requisicao.text
75
76              # Cria um objeto BeautifulSoup a partir do HTML obtido
77              soup = BeautifulSoup(html, "html.parser")
78
79              self.object.campo_fonte = soup.title.string + "\n"
80              # print(soup.title.string )
81              formulario = soup.find('form')
82              entradas = formulario.find_all('input')
83
84              for e in entradas:
85                  self.object.campos_fonte += e.get('name') + "\n"
86          except:
87                  messages.error(self.request,"Não foi possível descobrir corretamente os campos da fonte!")
88
89          self.object.save()
90          return super().form_valid(form)
```

Figura 10. Código no View de Fonte a ser executado no cadastro ou na atualização de uma fonte.

ilustra como as consultas são armazenadas.

Para o Web Crawler ter funcionamento completo, é preciso: identificar se há ou não CAPTCHA; identificar os campos (*inputs*) de formulários (*forms*) de pesquisa; injetar as palavras-chaves nesses campos de pesquisa; e trazer os metadados da busca realizada. Dessa forma, entra a API Selenium. Na Figura 13, é possível visualizar a ideia de como o processo de injetar dados em um site funciona. Registra-se que foi usado um *driver* do navegador Firefox (versão 114.0.2) para que aplicação funcionasse e pudesse realizar a busca. Há também variáveis que devem estar associadas às variáveis do portal, como *link, pesquisa, palavras_chave e botao*.

Já a Figura 14, mostra o resultado da execução do código da Figura 13, trazendo resultados da palavras_chave *inss* no

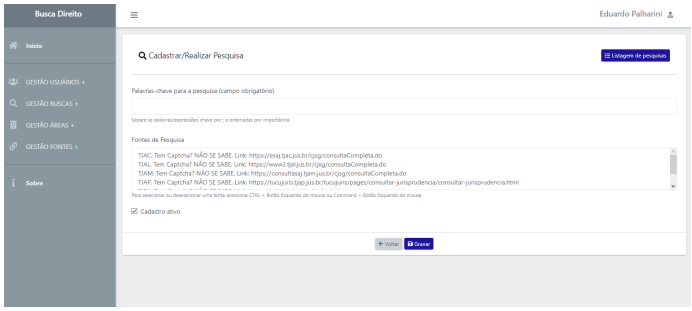

Figura 11. Gestão de Consultas.

Tribunal	Estado	Site do Tribunal	Link de Pesquisa Jurisprudencial	Termo para Pesquisar/Palavras-Chaves	Palavras-Chave utilizadas
TJAC	Acre	https://www.tjac.jus.br/	https://esaj.tjac.jus.br/cjsg/consultaCompleta.do	Pesquisa Livre	criança E adolescente E direito à educação E covid19
TJAL	Alagoas	https://www.tjal.jus.br/	https://www2.tjal.jus.br/cjsg/consultaCompleta.do	Pesquisa Livre	criança E adolescente E direito à saúde E covid19
TJAP	Amapá	https://www.tjap.jus.br/portal/	https://tucujuris.tjap.jus.br/tucujuris/pages/consultar-jurisprudencia/consultar-jurisprudencia.html	Pesquisa Livre	criança E adolescente E direito à saúde E pandemia
TJAM	Amazonas	https://www.tjam.jus.br/	https://consultasaj.tjam.jus.br/cjsg/consultaCompleta.do	Pesquisa Livre	(covid OU pandemia) E (criança OU adolescente) E saúde
TJBA	Bahia	http://www5.tjba.jus.br/portal/	https://jurisprudencia.tjba.jus.br/	Digite palavras sobre o assunto desejado	(covid OU pandemia) E (criança OU adolescente) E educação
TJCE	Ceará	https://www.tjce.jus.br/	https://conjug.tjce.jus.br/conjug-web/	Chave (s)	(covid OU pandemia) E (criança OU adolescente) E
TJES	Espírito Santo	http://www.tjes.jus.br/	http://aplicativos.tjes.jus.br/sistemaspublicos/consulta_jurisprudencia/cons_jurisp.cfm	Palavra Chave	(covid OU pandemia) E (criança OU adolescente) E educação
TJGO	Goiás	https://www.tjgo.jus.br/	https://projudi.tjgo.jus.br/ConsultaJurisprudencia	Digite um termo para a pesquisa	(covid OU pandemia) E (criança OU adolescente) E educação
TJMA	Maranhão	https://www.tjma.jus.br/	https://jurisconsult.tjma.jus.br/#/submenu	Chave de Busca	(covid OU pandemia) E (criança OU adolescente) E
TJMT	Mato Grosso	http://www.tjmt.jus.br/	https://jurisprudencia.tjmt.jus.br/catalogo	Busca Livre ou Número Único/Protocolo	(covid OU pandemia) E (criança OU adolescente) E

Figura 12. Lista de alguns Tribunais de Justiça cadastrados.

```
1   from selenium import webdriver
2
3   navegador = webdriver.Firefox()
4
5   #link retirado da gestão da fonte do site do tribunal
6   link = 'https://esaj.tjac.jus.br/cjsg/consultaCompleta.do'
7   navegador.get(link)
8
9   #campo de pesquisa do formulário identificado pelo Beautiful Soup
10  pesquisa = '//*[@id="iddados.buscaInteiroTeor"]'
11
12  #valor de pesquisa cadastrado pelo usuário na consulta
13  palavras_chave = 'inss'
14  navegador.find_element('xpath', pesquisa).send_keys(palavras_chave)
15
16  #botão de submissão do formulário identificado pelo Beautiful Soup
17  botao = '//*[@id="pbSubmit"]'
18  navegador.find_element('xpath', botao).click()
```

Figura 13. Exemplo de código Python-Selenium para injetar pesquisa em site.

Tribunal de Justiça do Estado do Acre.

IV. Conclusões

O texto apresenta a ideia do trabalho, justificativa, objetivos e alguns conceitos e dinâmica sobre Web Crawlers, ou o processo de rastreamento *Web*.

Em seguida, apresenta o processo de modelagem em que foi necessário utilizar uma abordagem metodológica eficiente para garantir o sucesso dos processos. Para isso, foram aplicadas

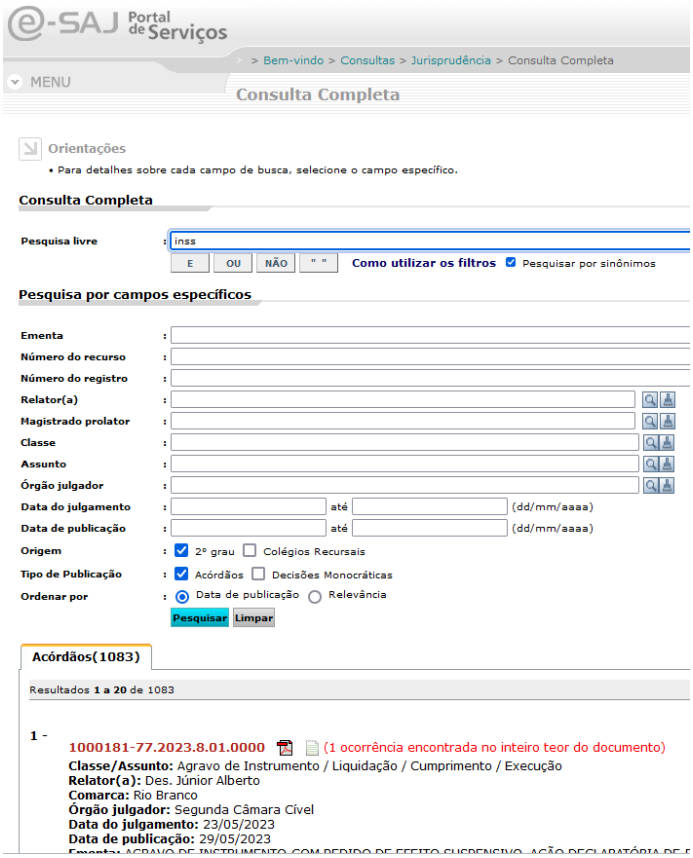

Figura 14. Resultado da aplicação Selenium na Consulta.

boas práticas do Scrum, como a definição de *sprints*, a organização do *backlog* e a realização de reuniões periódicas para acompanhar o progresso do projeto. Além disso, o uso da ferramenta Trello (Kanban) auxiliou na visualização e no gerenciamento das tarefas, auxiliando no planejamento, organização e execução do trabalho.

A diagramação do projeto realizada com o auxílio da ferramenta Astah, permitiu uma melhor compreensão do projeto como um todo. A escolha da linguagem de programação Python e do *framework* Django, se mostrou acertada, uma vez que combinados a versatilidade de um com a robustez e escalabilidade do outro foi possível desenvolver um sistema *Web* com todos os recursos necessários para seu devido funcionamento e gerenciamento.

Através da biblioteca Beautiful Soup, foi possível navegar pelas páginas, extrair informações estruturadas (como *forms* e *inputs*) e armazená-las em um formato desejado. O *framework* Scrapy desempenha um papel complementar ao Beautiful Soup, na análise e extração de dados em páginas HTML. Ele simplifica o processo de manipulação de elementos e permite pesquisas por *tags*, classes e outros critérios. A API Selenium, por sua vez, desempenha papel fundamental no sistema injetando as palavras-chave digitadas pelo usuário nos *inputs* de busca dos portais selecionados.

Em resumo, as tecnologias Python, Django, Beautiful Soup, Scrapy e Selenium foram fundamentais para o desenvolvimento deste trabalho. Elas proporcionaram uma solução piloto eficiente para a construção inicial de um Web Crawler, permitindo a coleta e extração automatizada de dados de páginas Web. Dessa forma, o sistema tem uma estrutura Web consolidada, com funções básica implementadas e disponibilizadas.

Finalmente, esta pesquisa possui alguns desafios e trabalhos futuros. Tendo em vista que há 27 Tribunais de Justiça e que a maioria não obedece um padrão de construção de sistema para pesquisas em seus sites, fica quase que inviável detectar automaticamente os campos e selecioná-los para realizar as pesquisas pelo *crawler* construído. Assim, como feito neste trabalho piloto, esses campos são detectados automaticamente, mas a seleção acontece manualmente no momento do cadastrado da fonte. Um trabalho futuro importante, é armazenar os resultados (metadados de acórdãos) que o *crawler* identificou, depois que os dados de pesquisa foram injetados nos links dos portais selecionados.

Referências

[1] J. Masanes, "Web archiving: issues and methods," in *Web archiving*. Springer, 2006, pp. 1–53.

[2] A. d. O. Zamberlan, "Técnicas de inteligência aplicadas ao direito: Representação de conhecimento e raciocínio automatizado," *XXV Simpósio de Ensino, Pesquisa e Extensão - SEPE*, p. 9, 2021.

[3] R. L. da Silva, "O direito da criança e do adolescente e a experiência de extensão universitária durante a pandemia," *Disciplinarum Scientia—Sociais Aplicadas*, vol. 17, no. 2, pp. 59–74, 2021.

[4] Sistema Judiciário Brasileiro: organização e competências. [Online]. Available: https://www.jusbrasil.com.br/noticias/sistema-judiciario-brasileiro-organizacao-e-competencias/2535347

[5] Como funciona o poder Judiciário no Brasil. [Online]. Available: https://portal.unit.br/blog/noticias/como-funciona-o-poder-judiciario-no-brasil/

[6] Panorama e estrutura do Poder Judiciário Brasileiro. [Online]. Available: https://www.cnj.jus.br/poder-judiciario/panorama-e-estrutura-do-poder-judiciario-brasileiro

[7] ACÓRDÃO. [Online]. Available: https://vademecumbrasil.com.br/palavra/acordao

[8] Cnj serviço: Saiba quando a decisão final é dada por sentença ou em acórdão. [Online]. Available: https://www.cnj.jus.br/cnj-servico-saiba-quando-a-decisao-final-e-dada-por-sentenca-ou-em-acordao/

[9] F. Tavares and L. M. Cunha, *Web Scraping, um caso de uso para coletar metadados de artigos científicos publicados na biblioteca digital da IEEE*. Instituto Federal Fluminense, 2021.

[10] R. R. Lecheta, *Google Android 4ª edição: Aprenda a criar aplicações para dispositivos móveis com o Android SDK*. Novatec Editora, 2015.

[11] B. McLaughlin, G. Pollice, and D. West, *Use a Cabeça - Análise e Projeto Orientado ao Objeto*. Alta Books Editora, 2007.

[12] Beautiful soup: Scrapper web. [Online]. Available: https://www.crummy.com/software/BeautifulSoup/

[13] Framework scrapy. [Online]. Available: https://scrapy.org/

[14] Selenium client driver. [Online]. Available: https://www.selenium.dev/selenium/docs/api/py/#contributing

[15] Programming language python. [Online]. Available: http://www.python.org/about/

[16] Framework django. [Online]. Available: https://www.djangoproject.com/

[17] L. Molinari, *Gerência de Configuração-Técnicas e Práticas no Desenvolvimento do Software*. Florianópolis: Visual Book, 2007.

[18] Db browser for sqlite. [Online]. Available: https://sqlitebrowser.org/

[19] R. E. dos Santos, A. de Oliveira Zamberlan, S. A. G. Vieira, G. C. Kurtz, and R. Frohlich, "Proposta de uma plataforma de cloud computing para disponibilização de um sistema online para consultórios e clínicas por meio do modelo saas," *Tópicos em Ciências da Saúde Volume 24*, p. 7, 2021.

[20] J. Sutherland, *Scrum: a arte de fazer o dobro do trabalho na metade do tempo*. Leya, 2016.

[21] E. Palharini and A. Zamberlan, *TFG Online: Ambiente para gestão eletrônica de trabalhos finais de graduação*. Disciplinarum Scentia, 2021.

[22] F. Prass, F. M. Boijink, and A. Zamberlan, *Parser e leitura automatizada de currículos da plataforma Lattes para extração de indicadores acadêmicos e tecnológicos*. Editora Atena, 2019.

[23] Most valuable product. [Online]. Available: https://gomainspring.com/filemaker-community/mvp-vs-mvp-minimum-viable-product-vs-most-valuable-product/,addendum=