INSTITUTO FEDERAL DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA DO CEARÁ
IFCE CAMPUS ARACATI
COORDENADORIA DE CIÊNCIA DA COMPUTAÇÃO
BACHARELADO EM CIÊNCIA DA COMPUTAÇÃO

IGOR MARTINS GALDINO

UM *BOT* PARA OBTENÇÃO AUTOMÁTICA DE DADOS PÚBLICOS USANDO AS TÉCNICAS DE *WEB CRAWLING* E *WEB SCRAPING*

ARACATI
2020

IGOR MARTINS GALDINO

UM *BOT* PARA OBTENÇÃO AUTOMÁTICA DE DADOS PÚBLICOS USANDO AS TÉCNICAS DE *WEB CRAWLING* E *WEB SCRAPING*

Trabalho de Conclusão de Curso (TCC) apresentado ao curso de Bacharelado em Ciência da Computação do Instituto Federal de Educação, Ciência e Tecnologia do Ceará - IFCE - *campus* Aracati, como requisito parcial para obtenção do título de Bacharel em Ciência da Computação.

Orientadora: Profa. Ma. Érica de Lima Gallindo.

Coorientador: Prof. Dr. Mário Wedney de Lima Moreira.

Aracati
2020

Dados Internacionais de Catalogação na Publicação
Instituto Federal do Ceará - IFCE
Sistema de Bibliotecas - SIBI
Ficha catalográfica elaborada pelo SIBI/IFCE, com os dados fornecidos pelo(a) autor(a)

G149b Galdino, Igor Martins.
Um bot para obtenção automática de dados públicos usando as técnicas de web crawling e web scraping / Igor Martins Galdino. - 2020.
69 f. : il. color.

Trabalho de Conclusão de Curso (graduação) - Instituto Federal do Ceará, Bacharelado em Ciência da Computação, Campus Aracati, 2020.
Orientação: Profa. Ma. Erica de Lima Gallindo.
Coorientação: Prof. Dr. Mário Wedney de Lima Moreira.

1. Transparência ativa. 2. Mineração de dados. 3. Web bots. I. Titulo.

IGOR MARTINS GALDINO

UM *BOT* PARA OBTENÇÃO AUTOMÁTICA DE DADOS PÚBLICOS USANDO AS TÉCNICAS DE *WEB CRAWLING* E *WEB SCRAPING*

Trabalho de Conclusão de Curso (TCC) apresentado ao curso de Bacharelado em Ciência da Computação do Instituto Federal de Educação, Ciência e Tecnologia do Ceará - IFCE - *Campus* Aracati, como requisito parcial para obtenção do Título de Bacharel em Ciência da Computação.

Aprovada em 26 de Março de 2020

BANCA EXAMINADORA

Profa. Ma. Erica de Lima Gallindo (Orientadora)
IFCE

Prof. Dr. Mário Wedney de Lima Moreira (Coorientador)
IFCE

Prof. Me. Silas Santiago Lopes Pereira
IFCE

Prof. Me. Ricardo Lenz César
IFCE

A Deus
Ao meu esforço
Aos meus professores

AGRADECIMENTOS

Primeiramente, agradeço a Deus por me fornecer saúde, força e paciência para concluir este trabalho.

Ao IFCE por proporcionar essa fase da minha formação, as oportunidades, a ótima qualidade de ensino e o abrigo, já que eu passava mais tempo nesta instituição do que em minha própria residência.

Ao LAR pelo ambiente agradável e pela “mística”, que foi de grande ajuda em meu crescimento pessoal e profissional. Este incentivo levou-me a chegar até ao final do curso.

Aos professores Reinaldo Braga e Carina de Oliveira pelas oportunidades nos projetos do laboratório e pelo grande apoio e confiança que me motivaram a permanecer no curso.

À minha professora e orientadora Erica Gallindo por todo apoio na realização desse trabalho, pela total paciência e compreensão, por acreditar no meu potencial e pela grande motivação para essa conclusão. Quando eu pensava em desistir, lembrava-me sempre da frase que me inspirou a concluir esse trabalho: "*Jede anstrengung hat ihre belohnung*" (Todo esforço tem sua recompensa).

Ao professor e coorientador Mário Wedney que me ajudou muito nas dicas de escrita e na normalização desse trabalho. Foi de grande ajuda e aprendi bastante.

Aos meus colegas e amigos, Leonardo Freitas, Rômulo Henrique, Guilherme Martins, Ruan Gondim, Vinícius Nunes, Edgar dos Santos, Renato Alves e Roberta Alencar, constantes motivadores uns dos outros para seguirmos todos sempre fortes. Estes amigos foram de grande importância para o meu crescimento acadêmico.

À minha namorada Michelle Moura pela paciência nos dias ausentes, estressantes e cansativos, por cuidar de mim, apoiar nas minhas escolhas e acreditar no meu potencial.

À Escola Virtual.Gov (EV.G) por ter me possibilitado atuar na solução de problema prático, aplicando os conhecimentos adquiridos ao longo de minha formação acadêmica.

Agradeço a todos os meus professores que de alguma forma contribuíram nos conhecimentos obtidos.

À banca avaliadora pelas ricas contribuições.

RESUMO

A Escola Virtual.Gov (EV.G) é uma iniciativa coordenada pela Escola Nacional de Administração Pública (Enap) para centralizar a oferta de cursos de capacitação profissional a distância. Diversos órgãos governamentais demandam à EV.G o desenvolvimento de cursos para capacitar seus profissionais, e para viabilizar estas ações, repassam recursos financeiros à Enap (EV.G). Para promover uma transparência ativa, e em conformidade com a Lei de Acesso à Informação, a prestação de contas da aplicação desses recursos precisa estar disponível ao cidadão comum. A EV.G gerencia os recursos através de um sistema próprio no qual se armazenam os documentos financeiros associados à execução da despesa pública, tais como: notas de empenho e ordens de pagamento. Tais documentos são obtidos a partir de buscas manuais no Portal da Transparência, em um procedimento demorado e propenso a erros. O principal objetivo deste trabalho é automatizar a obtenção dos dados do Portal da Transparência e disponibilizar *dashboards* com base nestas informações. Assim, foram criados mecanismos para obtenção automática dos dados, através dos *bots* da *web*, utilizando as técnicas *web crawling* e *web scraping*. Além disso, foi criada uma API, acionada por meio de uma *interface web* para o uso desses *bots*. Por meio dessa *interface web*, um servidor da Enap (EV.G) consegue fazer *download* dos documentos obtidos automaticamente para inserção no sistema de armazenamento da Enap (EV.G). Por fim, foi desenvolvida uma aplicação que realiza diariamente a sincronização dos dados do sistema de armazenamento da Enap (EV.G) para a fonte que alimenta o portal Em Números.

Palavras-chaves: Transparência ativa. Mineração de dados. *Web Bots*.

ABSTRACT

The Escola Virtual.Gov (EV.G) is an initiative coordinated by the Escola Nacional de Administração Pública (Enap) to centralize the offer of distance training courses. Several government agencies demand the EV.G to develop courses to train their professionals and to make these stations feasible, transfer financial resources to Enap (EV.G). In order to promote active transparency, the information about how the public budget was used must be available to the brazilian population. The Enap(EV.G) manages resources through its systems in which the financial documents associated with the execution of public expenditure are stored, such as payment orders. Such documents are obtained from manual searches on the Portal da Transparência, in a lengthy and error-prone procedure. The main objective of this work is to automate the retrieval of data from the Portal da Transparêncial and make it available dashboards based on this information. Thus, mechanisms were created for automating data retrieval, using bots with web crawling and web scraping techniques. Besides, an API was created, triggered through a Web interface for the use of these bots. Through this Web interface, an Enap server (EV.G) manages to download the documents obtained automatically for insertion into the Enap storage system (EV.G). Finally, an application was developed that performs the daily synchronization of data from Enap's storage system (EV.G) to the source that feeds the Em Números website.

Keywords: Active transparency. Data mining. Web Bots.

LISTA DE ILUSTRAÇÕES

LISTA DE ABREVIATURAS E SIGLAS

AJAX	*Asynchronous JavaScript And XML*
API	*Application Programming* Interface
AWS	*Amazon Web Services*
CGU	Controladoria-Geral da União
CNPq	Conselho Nacional de Desenvolvimento Científico e Tecnológico
CSS	*Cascading Style Sheets*
CSV	*Comma Separated Values*
Enap	Escola Nacional de Administração Pública
e-SIC	Sistema Eletrônico do Serviço de Informações ao Cidadão
EV.G	Escola Virtual.Gov
GCP	*Google Cloud Plataform*
HTTP	*HyperText Transfer Protocol*
HTML	*Hypertext Markup Language*
IFCE	Instituto Federal de Educação, Ciência e Tecnologia do Ceará
IoT	*Internet of Things*
JSON	*JavaScript Object Notation*
LAI	Lei de Acesso à Informação
NC	Nota de Crédito
NE	Nota de Empenho
OB	Ordem Bancária
PDF	*Portable Document Format*
SIAFI	Sistema Integrado de Administração Financeira
SIAPE	Sistema Integrado de Administração de Pessoal
SQL	*Structured Query Language*

STN	Secretaria do Tesouro Nacional
TCC	Trabalho de Conclusão de Curso
TED	Termo de Execução Descentralizada
UG	Unidade Gestora
URL	*Uniform Resource Locator*
W3C	*World Wide Web Consortium*
WSGI	*Web Server Gateway Interface*
XML	*Extensible Markup Language*

SUMÁRIO

1 INTRODUÇÃO

De acordo com a Constituição Federal de 1988[1], é direito do cidadão participar da gestão pública e poder supervisionar a aplicabilidade dos recursos utilizados, viabilizando o controle social, ou seja, permitindo que a sociedade também possa atuar na fiscalização dos recursos financeiros executados pelo Estado. O controle social das ações dos governantes e funcionários públicos é importante para assegurar que os recursos públicos sejam bem empregados em benefício da coletividade[2].

A Lei nº 12.527, de 18 de novembro de 2011, conhecida como Lei de Acesso à Informação (LAI), regulamenta o direito constitucional de acesso às informações públicas. É obrigação dos governos federal, estaduais e municipais garantir o direito de acesso a informação de modo transparente, ágil, claro e com linguagem de fácil compreensão. Para permitir que os cidadãos pudessem solicitar as informações regulamentadas pela LAI, a Controladoria Geral da União (CGU)[3] criou o Sistema Eletrônico de Informação ao Cidadão (e-SIC). Usando o e-SIC, qualquer cidadão pode solicitar informações aos órgãos federais que, por sua vez, terão a obrigação de fornecê-las, excetuando-se casos específicos previstos na LAI. Em 2012, ano de início de funcionamento do e-SIC, foram registrados 55.212 pedidos de acesso à informação em todos os órgãos da administração federal[4]. Em 2018, este número já era 129.309, demonstrando um aumento significativo do interesse da população pelas ações governamentais.

Para lidar com a alta demanda de solicitações, cada vez mais órgãos da administração pública estão aderindo a iniciativas de transparência ativa (BRASIL. Senado Federal, 2017), ou seja, iniciativas de divulgação de informações de interesse geral, independentemente de terem sido solicitadas por algum cidadão. O Governo Federal, especificamente os órgãos de controle da administração pública, vêm incentivando a criação de *sites* com informações que possam ser consultadas e analisadas pela população em geral, viabilizando mais um meio que permita o combate à corrupção (SANTOS, 2018). O Portal da Transparência do Governo Federal, lançado pela Ministério da Transparência e pela CGU, em 2004, é o principal exemplo de aplicação do conceito da transparência ativa em *sites* governamentais.

O Portal da Transparência é um canal de acesso livre, significando que o seu

[1] Disponível em: http://www.planalto.gov.br/ccivil_03/constituicao/constituicao.htm
[2] Disponível em: http://www.portaltransparencia.gov.br/pagina-interna/603399-controle-social
[3] A CGU é um órgão do Governo Federal responsável pela defesa do patrimônio público e pelo incremento da transparência da gestão, através de atividades de controle interno, auditoria pública, correição, prevenção e combate à corrupção e ouvidoria.
[4] Disponíveis em: https://esic.cgu.gov.br/sistema/Relatorios/Anual/RelatorioAnualPedidos.aspx

acesso é realizado sem requerer nem usuário e nem senha. Este portal é referente ao poder executivo, e objetiva permitir que o cidadão acompanhe os gastos e investimentos dos recursos federais que são arrecadados por meio de impostos. A partir dele, pode-se buscar dados referente ao orçamento, licitações, contratos, órgãos, despesas, servidores, dentre outros assuntos da Administração Pública Federal. Os dados são originários de diversas fontes, tais como: i) Sistema Integrado de Administração Financeira do Governo Federal (SIAFI), que realiza acompanhamento do processo de controle e execução financeira do Governo Federal e; ii) Sistema Integrado de Administração de Pessoal (SIAPE), o qual processa o pagamento dos servidores ativos, aposentados e pensionistas.

Em 2018, o Governo Federal lançou o novo Portal da Transparência, proporcionando melhor usabilidade, buscas mais intuitivas, painéis gráficos e agregação de redes sociais. Embora permita que o cidadão comum obtenha dados de seu interesse, por ter sido construído como uma ferramenta de propósito geral e com visualizações de dados padronizadas, o portal não atende necessariamente às necessidades específicas de todos os órgãos da administração pública, que precisam construir outras visualizações dos mesmos dados que estão disponibilizados no referido ambiente. A Escola Nacional de Administração Pública (Enap), é um exemplo de entidade da administração pública com requisitos próprios para disponibilização de informação sobre a execução de recursos orçamentários e financeiros que ela gerencia.

A Enap coordena a Escola Virtual.Gov (EV.G), que oferta cursos a distância para capacitação profissional de servidores públicos de todo país. No ano de 2018 foram realizadas cerca de 442.000 inscrições em cursos ofertados na EV.G, tendo este número aumentado para 940.00 em 2019[5], demonstrando o crescente interesse pelo tipo de capacitação ofertada.

Diversos órgãos da administração pública utilizam a EV.G como principal mecanismo de capacitação de seus servidores. Com frequência, os cursos ofertados pela EV.G são demandados por instituições parceiras, confeccionados e posteriormente disponibilizados em seu portal[6]. Para viabilizar a confecção e o desenvolvimento dos cursos, as instituições parceiras descentralizam recursos à Enap (EV.G), por meio de um Termo de Execução Descentralizada (TED)[7], para execução de um plano de trabalho previamente acordado, permitindo a instituição parceira acompanhe a aplicação dos recursos.

[5] Disponível em: https://emnumeros.escolavirtual.gov.br/indicadores/

[6] Disponível em: https://www.escolavirtual.gov.br

[7] Cada TED, definido no Decreto nº 6.170, de 25 de julho de 2007, é acompanhado por um conjunto de informações, tais como: plano de trabalho, definição do objeto, propósitos a serem alcançados, etapas e recursos envolvidos, permitindo o acompanhamento da unidade que descentralizou o recurso.

A Enap (EV.G) gerencia vários TEDs simultâneos e para isso possui uma solução de *software* para armazenar as informações dos principais documentos associados à execução financeira, tais como: TED, nota de crédito, nota de empenho, notas de liquidação, ordem de pagamento, entre outros. Estas informações são inseridas manualmente no sistema, utilizando para isso dados que precisam ser obtidos junto às instituições que receberam recursos da Enap (EV.G). Como a frequência de lançamentos de novos dados é muito alta, o tratamento manual não garante que os registros estejam sempre atualizados. Logo, faz-se necessário um mecanismo para atualização automática dos dados armazenados no sistema da Enap (EV.G), com base nas fontes de dados oficiais existentes.

Nesse contexto, o presente trabalho tem como objetivo propor uma solução para automatizar a atualização dos dados de execução de despesa pública, necessários ao monitoramento realizado pela Enap (EV.G), usando para isso informações do Portal da Transparência. Os dados necessários ao monitoramento das ações da Enap (EV.G) serão extraídos automaticamente do referido portal, inseridos nos sistemas de registros existentes na Enap (EV.G), e publicados por meio de *dashboards*[8] disponíveis no *site* da entidade. Isso viabilizará a transparência ativa incentivada pelos órgãos de controle, viabilizando o controle social e uma visão global da aplicação dos recursos que circulam na Enap (EV.G).

1.1 Objetivos

1.1.1 Objetivo Geral

Desenvolver uma aplicação para obtenção automática de dados da execução de despesas públicas, a partir do Portal da Transparência, e criar um mecanismo para armazenamento em nuvem dos dados extraídos por meio das técnicas de *web crawling* e *web scraping*.

1.1.2 Objetivos Específicos

- Criar *bots* da *web* para obter automaticamente dados do Portal da Transparência;
- Criar uma API responsável pelo acionamento de *bots*;
- Criar uma *interface web* para fazer uso da API desenvolvida;

[8] O *dashboard* é um painel visual que oferece visualizações rápidas dos principais indicadores de desempenho relevantes para um objetivo ou processo de negócios específico.

- Extrair os dados obtidos, por meio das técnicas supramencionadas, para um arquivo CSV; e
- Criar um *script* responsável pela sincronização da fonte de dados do portal Em Números com uma área de armazenamento na nuvem.

1.2 Organização do Trabalho

O restante desse trabalho está organizado em como se segue. No Capítulo 2, para uma compreensão da proposta de solução adotada, detalha-se o contexto do trabalho com ênfase nos conceitos associados à execução da despesa pública e à realidade computacional no qual este trabalho se insere.

No Capítulo 3 são analisados outros trabalhos que obtêm dados por meio do uso de aplicações com *bots* da web.

A metodologia utilizada para desenvolver este trabalho é descrita no Capítulo 4 enquanto que no Capítulo 5 são relatados os resultados obtidos no decorrer dos testes realizados a partir da solução proposta.

Por fim, no Capítulo 6, são apresentadas as considerações finais acerca do trabalho realizado, bem como sugestões de possíveis trabalhos futuros para promover a sequência e o aprimoramento do presente trabalho.

2 FUNDAMENTAÇÃO TEÓRICA

Nesta seção são apresentados diferentes métodos de extração de dados na *Web*, as técnicas utilizadas por *bots* da internet, *frameworks* e bibliotecas disponíveis para solução desse trabalho. Em seguida, são descritos os serviços do Google para armazenamento automatizado de dados no *Google Drive*, pois com o *Google Drive* foi encontrado uma solução para sincronização automática e de forma gratuita com a ferramenta que gera os *dashboards*.

Esta parte técnica, que na *Web*, as técnicas utilizadas por *bots* da internet, *frameworks* e bibliotecas disponíveis para estes fins. Em seguida, são descritos os serviços do Google para armazenamento automatizado de dados no *Google Drive*, vez que esta estratégia foi usada como parte da solução proposta. Esta parte técnica, que subsidia a proposta de solução apresentada neste trabalho, está detalhada nas seções 2.1 e 2.2.

Para conceituar minimamente a área do negócio no qual este trabalho se insere, na Seção 2.3 são detalhados os processos de elaboração do orçamento público e os documentos que formalizam as fases de execução da despesa pública. Finaliza-se com uma explicação da execução financeira no âmbito da Enap (EV.G) e do processo existente para proporcionar a transparência ativa no contexto dessa instituição.

2.1 Extração de dados da *web*

2.1.1 Bots da web

O *bot* da *web*, também chamado de *spider*, é um algoritmo usado para analisar e extrair informações dos *websites*, de forma sistemática e automatizada (OMARI; SHOHAM; YAHAV, 2016). Os *bots* capturam os textos das páginas e cadastram os *links* identificados, para que possam ser posteriormente utilizados na localização de novas páginas. Eles são bastantes utilizados pelos motores de buscas para criar cópias das páginas pesquisadas e armazena em suas bases, com isso eles conseguem oferecer respostas mais rápidas. Além do mais, é possível extrair dados contidos nas páginas pesquisadas e estruturá-los para serem analisados a posteriori.

Estes *bots* podem realizar dois tipos de técnicas, rastreamento de dados e raspagem de dados, conhecidas respectivamente como: *web crawling* o procedimento de indexação dos *links* que serão visitados e *web Scraping* processo para obtenção do conteúdo das páginas. Os *bots* podem ser criados utilizando estas técnicas de

forma simultânea ou como tarefas distintas (KHALIL; FAKIR, 2017). Entenda-se como *bot crawler* o *bot* que utiliza a técnica *web crawling* e *bot scraper* que usa a técnica *web scraping*.

2.1.2 Web crawling

O *web crawling* é o processo responsável por efetuar as buscas das páginas *web* e indexá-las. Ele captura informações dos *websites* e cadastra os links encontrados, de forma que possam ser localizados futuramente (D'HAEN et al., 2016). O processo é iniciado a partir de uma *seed* (semente), que consiste de uma lista inicial de endereços a serem visitados. À medida que o *bot* visita estes endereços, os *hyperlinks* são identificados e adicionados à lista de endereços a serem visitados (KHALIL; FAKIR, 2017), proporcionando a localização de outras páginas, mantendo assim o banco de dados atualizado. Como exemplos de *bots* pode-se citar: (i) ***Googlebot*** do Google; (ii) ***Msnbot* e *Bingbot*** da Microsoft; e (iii) ***Yahoo! Sluro*** do Yahoo! (Algiryage; Dias; Jayasena, 2018).

O Google, *site* de busca mais utilizado no mundo[1]. Segundo Eric Schmidt, executivo-chefe do Google, o Google levaria em torno de 300 anos para conseguir rastrear e catalogar todas informações de páginas *web* disponíveis na internet (MOTOMURA, 2016). O Google utiliza três etapas básicas para gerar resultados a partir de páginas da *web*:

- **rastreamento** que descobre quais páginas existem, e, de forma iterativa, localiza-se e busca-se *links* na *web*, a partir da lista inicial de URLs a serem visitadas (*seeds*). Em seguida, as novas URLs identificadas são adicionadas a uma lista de URLs a visitar.
- **indexação** ocorre após uma página ter sido descoberta, em seguida, o Google analisa seu conteúdo, associando arquivos de imagens e vídeos agregados, tentando catalogá-las a partir do assunto tratado. A partir disso, essas informações são indexadas no banco de dados em diversos computadores.
- Veiculação ocorre quando o usuário efetua uma consulta, o Google busca a resposta mais relevante com base em vários fatores, tais como localização, idioma e dispositivo utilizado para acesso à página (computador ou *smartphone*) (GOOGLE, 2019a).

Além das questões de volume de dados disponíveis na internet, cabe destacar que o conteúdo destas páginas está em constante atualização, levando a necessidade

[1] Disponível em: https://neilpatel.com/br/blog/sites-de-busca/

de se estabelecer, em um dado instante de tempo, quais páginas serão priorizadas (NIÑO-MORA, 2014). As políticas de eficiência na implementação de um *bot crawler*, descritas a seguir, podem ser utilizadas para auxiliar no processo de priorização.

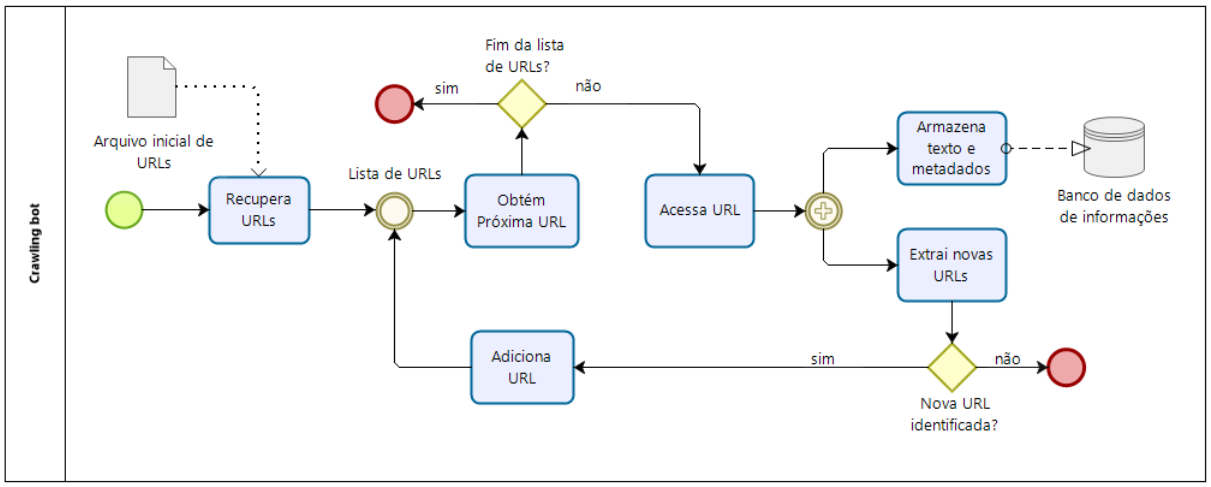
- **Política de seleção**: Define quais páginas serão indexadas. Devido à grande quantidade de conteúdo na *web*, é muito difícil armazenar todas estas informações. Então, é desejável que se armazene apenas as páginas consideradas mais relevantes.
- **Política de revisitação**: Define a forma de atualização dos registros das páginas. Como as páginas *web* estão em constante atualização, existem duas políticas de revisitação possíveis. A **política uniforme** estabelece que todas as páginas serão revisitadas com a mesma frequência enquanto que a **política proporcional** estabelece que serão revisitadas as páginas que são mais frequentemente atualizadas.
- **Política de cordialidade**: Existe para evitar a sobrecarga dos *sites*. Um *crawler* pode efetuar, no mesmo intervalo de tempo, muito mais requisições do que um ser humano, podendo consumir bastante banda e sobrecarregar os servidores. Uma maneira seria colocar um acréscimo no tempo para cada requisição de um *bot crawler*.
- **Política de paralelização**: Existe para coordenar *crawlers* distribuídos, separando as responsabilidades entre os *crawlers* de uma mesma rede para visitação de páginas *web*. Essa política pode ser implementada de duas formas distintas. No modo **associação dinâmica**, no qual existe um coordenador que contém uma lista de URLs a visitar. Ele responsável por encaminhar esta lista aos *crawlers* responsáveis pela visitação. Este coordenador pode atribuir ou remover URLs da lista, dinamicamente. No modo **associação estática**, por sua vez, não existe um coordenador, porém, os *crawlers* podem trocar URLs entre si.

A Figura 1 ilustra o processo realizado por um *crawler*. Inicialmente, são recuperadas as URLs a partir de um arquivo inicial (*seed*) e adicionadas a uma lista. Em seguida, o *crawler* recupera dessa lista a próxima URL a ser visitada. Então, para acessar a URL, é necessária a existência de um cliente HTTP[2] que envie uma solicitação e obtenha uma resposta (UDAPURE; KALE; DHARMIK, 2014). O arquivo HTML resultante da solicitação realizada pode conter novas URLs, pois o conjunto de URLs são extraídas e podem ser adicionadas a lista de URLs a serem visitadas. Além disso,

[2] HTTP (*HyperText Transfer Protocol* - Protocolo de Transferência de Hipertexto) é um protocolo de comunicação para sistemas de informação de hipermídia, permite a transferência de dados na *World Wide Web*.

são indexados e armazenados os metadados da URL visitada no banco de dados. Esse ciclo continua enquanto tiver URLs a serem visitadas ou alguma restrição de parada.

Figura 1 – Funcionamento de um *web crawling*



Fonte: Elaborado pelo autor.

2.1.3 *Web scraping*

O *web scraping* é o processo de extração utilizado para coletar dados relevantes de *sites* de forma automática, convertendo as informações desestruturadas em estruturadas, para serem posteriormente analisadas (ZHAO, 2017), em um procedimento conhecido como raspagem de dados.

O *bot* que utiliza a técnica *web scraping* é programado para efetuar requisições a um servidor *web*, a partir de uma lista predefinida de URLs ou retornado pela técnica *web crawling*, após a solicitação são extraídos os dados necessários. As informações obtidas são copiadas e podem ser exportadas em arquivos nos formatos JSON[3], CSV[4], entre outros. Normalmente, este processo simula uma navegação humana na utilização de um *site*, porém o *bot* consegue efetuar mais requisições do que um ser humano.

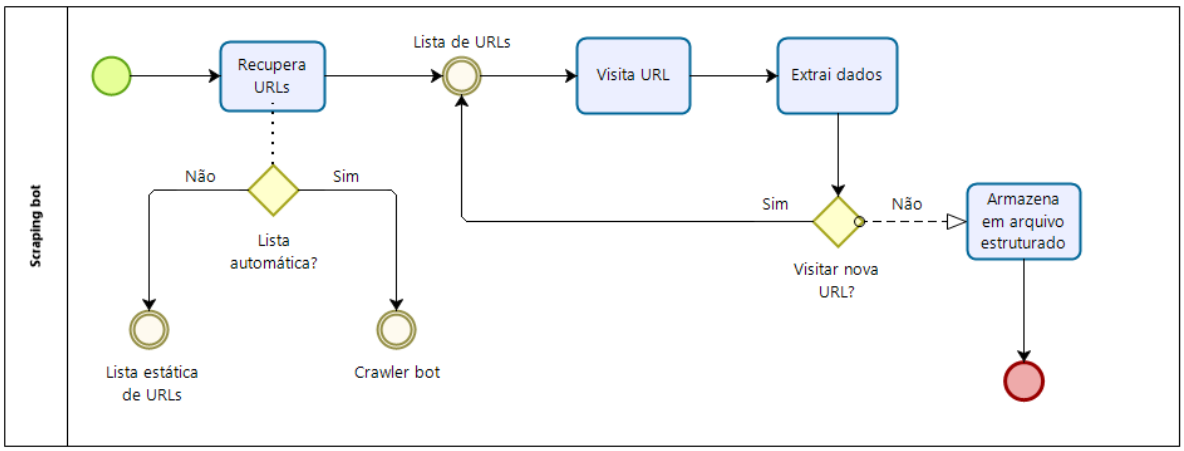
A Figura 2 demonstra o comportamento de um *bot* aplicando a técnica *web scraping*. De início, são obtidos as URLs e adicionadas à lista de solicitações. As URLs podem ser recuperadas por um *crawler bot*, dinamicamente, ou de uma lista estática predefinida. Após a lista de URLs ser atualizada, o *scraper bot* envia uma solicitação HTTP GET para as páginas *web* e faz o *download* do arquivo HTML. A partir

[3] JSON (*JavaScript* Object Notation - Notação de Objetos *JavaScript*) é um formato leve de troca de dados. Para seres humanos, é fácil de ler e escrever.

[4] CSV (*Comma separated value* - valores separados por vírgulas) é um tipo de arquivo de texto fundamental para transferência de informações entre aplicativos diferentes.

disso, o *scraper bot* faz o processo de extração de dados. Estes dados podem ser apanhados por meio do CSS *Selector* ou através do *XPath*[5], pois, o arquivo baixado possui informações que são desnecessárias e precisa ser filtrados (MAHTO; SINGH, 2016). Se houver mais URLs para visitar, ele retorna para a lista e continua o ciclo de solicitação HTTP e extração. Por fim, após a extração de dados, essas informações podem ser armazenadas em arquivos estruturados para futura análise, tais como CSV, JSON, XML e outros.

Figura 2 – Comportamento do *web scraping*



Fonte: Elaborado pelo autor.

Para a utilização do *web crawling* e *web scraping* é necessário ter alguns cuidados. A má utilização dessas técnicas pode se tornar uma prática ilegal, pois, muitos *sites* possuem políticas e ações específicas para proibir ou atrapalhar o garimpo de dados, e o uso dessas técnicas podem ser considerados ilegal[6] Existe alguns pontos de atenção na aplicação desses processos:

- **Robots.txt**: é um arquivo em formato de texto que fica localizado na raiz do *site*. Ele gerencia o tráfego de um *crawler* desse *site*, informando quais páginas ou arquivos podem ser solicitados. Esse recurso é utilizado para evitar sobrecarga do *site* com solicitações (GOOGLE, 2019b);
- **Termo de serviço**: respeitar os termos de serviços dos *sites*. Caso alguém entre em processo na justiça, as afirmações desses termos podem valer;
- **Leis do local em que o *site* está hospedado**: se a raspagem é feita em *sites* de outros países, o cuidado deve ser redobrado para não infringir as leis locais de proteção aos dados;

[5] *XPath* pode ser usado para navegar pelos elementos e atributos em um documento XML, que também podem ser usados com HTML. O *XPath* foi definido pelo W3C (*World Wide Web Consortium*).
[6] Disponível em https://rockcontent.com/blog/web-scraping/

- **Taxa de rastreamento**: o *bot* efeitua mais requisições ao servidor do que uma pessoa. Existe uma maior possibilidade do *site* identificar como um ataque. Aumente o tempo para o intervalo das requisições;
- **Identificação do *scraper***: criar um arquivo de identificação do *bot*, com as informações do proprietário e para que a utilização dos dados;
- **Proteção dos dados coletados**: se os dados que deseja capturar têm proteção de direitos autorais, evitar a coleta.

Na construção de um *bot* utilizando as técnicas de rastreamento e raspagens de dados não dependem de uma API (MITCHELL, 2018), normalmente é desenvolvido uma aplicação automatizada que consulta um servidor *web*. Algumas linguagens de programação dão suporte a ferramentas que possibilitam a realização das técnicas, como *Storm Crawler* do *Java*, o *Apify SDK* do *NodeJS*, *Kimurai* do *Ruby*, *Scrapy* e *BeautifulSoup* do *Python*.

Para este trabalho foi realizada uma análise de duas ferramentas que permitem a utilização das técnicas *web crawling* e *web scraping*, a saber: *BeautifulSoup* e *Scrapy*. *BeautifulSoup* é uma biblioteca da linguagem *Python* para extrair informações de documentos HTML e XML[7]. Com essa biblioteca é possível obter links das páginas, por meio da aplicação da técnica de *web crawling*, ou raspar o conteúdo delas usando a técnica de *web scraping*). O *Scrapy* é *framework* da linguagem *Python* que utiliza as técnicas supramencionadas, possibilitando a extração de informações e estrutura[8]. A biblioteca *Beautifulsoup* possui recursos limitados, enquanto o *Scrapy* é recomendado para projetos mais robustos (CHERKESOV et al., 2017). Em vista disso, para o desenvolvimento desse trabalho, optou-se pela utilização do *framework Scrapy*.

2.1.4 Scrapy

O *Scrapy* é um *framework* de código-fonte aberto e colaborativo para extrair os dados necessários de várias fontes (KOUZIS-LOUKAS, 2016). Ele pode ser usado para uma ampla variedade de aplicativos úteis, como mineração de dados, processamento de informações ou arquivamento histórico, de maneira rápida, simples e extensível.

Embora o *Scrapy* tenha sido criado para raspagem na *web*, ele também pode ser usado para extrair dados usando APIs (como *Amazon Web Services* - AWS) ou como um rastreador da *web* de uso geral. No projeto *Scrapy* são criados *spiders*

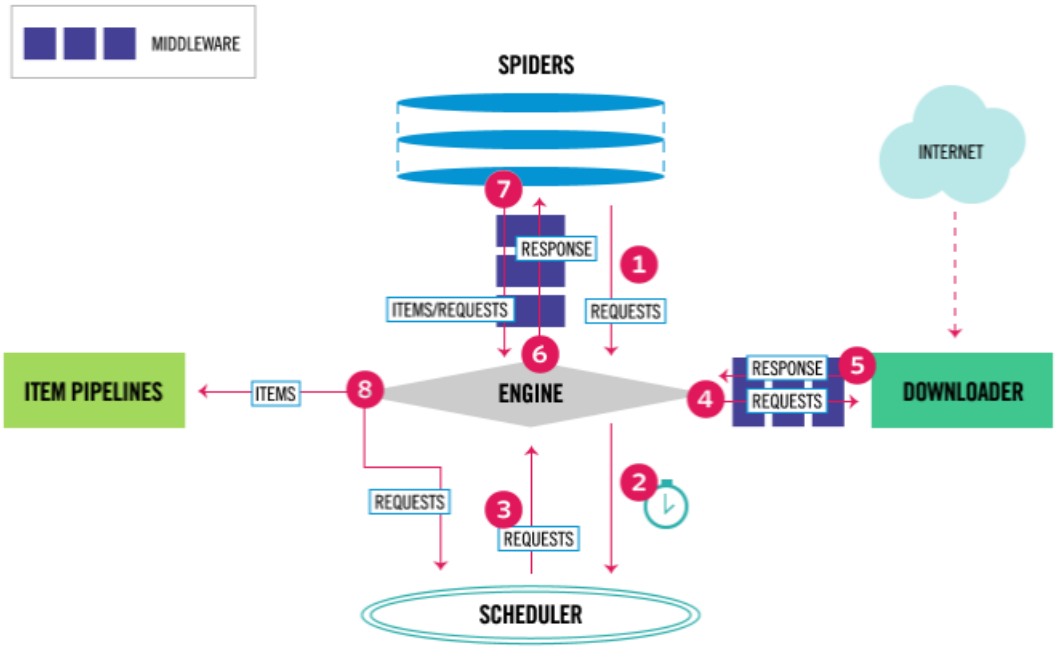
[7] Disponível na documentação do *BeautifulSoup* https://www.crummy.com/software/BeautifulSoup/bs4/doc/
[8] Disponível na documentação https://docs.scrapy.org/en/latest/

que recebem um conjunto de instruções para realizarem técnicas de rastreamento e raspagem de dados de forma assíncrona (RAMÍREZ-CORONEL et al., 2019).

As *Spiders* são classes que definem metodologicamente como o *site* será raspado (NISAFANI; HENDRAWAN; WIBISONO, 2017), ou seja, ela é uma estrutura de métodos que são estabelecidas regras de como os dados serão rastreados e capturados. Em um projeto *Scrapy* pode existir inúmeros *spiders*, como complemento ou propósitos diferente.

Na Figura 3 apresenta-se o diagrama com a arquitetura do *Scrapy* e seus componentes[9], cada um sendo responsável por um conjunto de atribuições descritas a seguir:

Figura 3 – Diagrama da arquitetura do *Scrapy*



Fonte: http://docs.scrapy.org/en/latest/topics/architecture.html

1. o *spider (bot)* efetua uma requisição a uma URL;
2. o mecanismo de execução (*engine*) do *Scrapy* agenda esta solicitação no Agendador e solicita o rastreamento das próximas solicitações;
3. o Agendador envia as próximas solicitações ao mecanismo;
4. o mecanismo envia uma requisição para o *downloader* da página solicitada, passando pelos *middlewares*[10];

[9] Disponível em http://docs.scrapy.org/en/latest/topics/architecture.html

[10] Os *middlewares* no *Scrapy* são interceptadores que processam solicitações específicos antes de uma requisição ou depois de uma resposta.

5. é enviada a resposta com a página baixada para o mecanismo, após serem processadas pelos *middleware*;

6. o mecanismo envia a resposta ao *bot* para processamento, passando pelo *middleware*;

7. o *bot* retorna os itens raspados e novas solicitações ao mecanismo, passando pelo *middleware*;

8. o mecanismo envia os itens raspados para os *pipelines* de itens, fazendo o tratamento do que foi raspado e armazenando em arquivos ou banco de dados, a seguir, envia solicitações processadas para o Agendador e solicita que as próximas solicitações sejam rastreadas. Por fim, o processo se repete a partir da etapa (i) até acabar as solicitações do Agendador (FAROOQ; HUSAIN; SUAIB, 2018).

Uma vantagem do *Scrapy* são suas solicitações agendadas e processadas de forma assíncrona. Ele não precisa esperar que uma solicitação seja concluída e processada, pode enviar a próxima solicitação ou fazer demais tarefas enquanto espera a resposta. Isso também significa que outras solicitações podem continuar, mesmo se alguma solicitação falhar ou ocorrer um erro ao manipulá-la.

Embora isso permita fazer rastreamentos muito rápido, enviando várias solicitações simultâneas ao mesmo tempo, de maneira tolerante a falhas, o *Scrapy* também oferece a qualidade do rastreamento por meio de algumas configurações. Pode ser definido um atraso no *download* entre cada solicitação, limitar a quantidade de solicitações simultâneas por domínio ou por IP e até mesmo usar uma extensão de aceleração automática que tenta descobrir isso automaticamente. Alguns recursos do *Scrapy* para tornar a raspagem de dados mais eficiente[11]:

- Suporte interno para selecionar e extrair dados de fontes HTML/XML usando seletores CSS estendidos e expressões *XPath*, com métodos auxiliares para extração usando expressões regulares.

- Um console de *shell* interativo para testar as expressões CSS e *XPath* para raspar dados, muito útil ao gravar ou depurar os *spiders*.

- Suporte integrado para gerar exportações em vários formatos e armazená-las em vários *back-ends*.

- Detecção automática, para lidar com declarações de codificação estrangeiras, quebradas e fora do padrão.

[11] Documentação do *scrapy*, disponível em http://docs.scrapy.org/en/latest/intro/overview.html

- Forte suporte à extensibilidade, permitindo conectar sua própria funcionalidade usando sinais e uma API bem definida.
- Vasta gama de extensões e *middlewares* integrados para manipulação.
  - Manipulação de *cookies* e sessões;
  - Recursos HTTP como compactação, autenticação, *cache*;
  - Falsificação de agente de usuário;
  - Robots.txt;
  - Restrição de profundidade de rastreamento e outros.
- Um console *Telnet* para conectar-se a um console *Python* executando dentro do seu processo *Scrapy*, para observar e depurar seu rastreador.
- Outras vantagens, como *spiders* reutilizáveis para rastrear *sites* a partir de *Sitemaps* e *feeds* XML/CSV, um *pipeline* de mídia para baixar automaticamente imagens (ou qualquer outra mídia) associadas aos itens raspados, um resolvedor de DNS em cache.

Os *websites* estão cada vez dinâmicos, por meio do *JavaScript*. Através do *JavaScript* é possível acelerar o carregamento de páginas da *web* (ZHOU et al., 2018). O *Scrapy* não executa naturalmente *JavaScript*, então pode-se utilizar algum complemento para fazer a execução do *JavaScript*, como o *Splash*.

2.1.4.1 Splash

O *Splash* é um serviço de renderização em *JavaScript*. É um navegador leve, implementado no *Python3* usando *Twisted* e *QT5* (ZHOU et al., 2018). O *reator QT* (*twisted*) é usado para tornar o serviço totalmente assíncrono, permitindo tirar vantagem da simultaneidade do *webkit* via *loop* principal do *QT*. Alguns recursos do *Splash*[12]:

- Processa várias páginas da *web* em paralelo;
- Obtém resultados em HTML e faz capturas de tela;
- Desativa imagens ou usa regras do *Adblock Plus* para tornar a renderização mais rápida;
- Executar *JavaScript* customizado no contexto da página;

[12] Disponível em https://splash.readthedocs.io

- Reconhece *scripts* de navegação em Lua;
- *Scripts Splash Lua* em blocos de anotações *Splash-Jupyter*;

2.1.4.2 Docker

Para simplificar inicialização do serviço do *Splash* foi utilizado o *Docker*. O *Docker* possui mecanismo de virtualização de *containers* que possui dependências necessárias para uma aplicação, ou seja, sem a necessidade de configuração de outros programas para execução da aplicação. Com base nisso, foi adquirido essa ferramenta que facilita na operação do *Splash* sem precisar da configuração do serviço, apenas execução do *Docker*.

O *Docker* é um *software* que fornece virtualização de *container* leve, que ajuda a fornecer aplicativos mais rapidamente (JARAMILLO; NGUYEN; SMART, 2016). Ele possui um conjunto de ferramentas que é necessário para executar, código, tempo de execução, ferramentas do sistema e bibliotecas. Isso ajuda os desenvolvedores a implantar e gerenciar aplicativos com mais facilidade.

O *Docker* empacota a aplicação e suas dependências para dentro de um *contaiter*, desse modo o *container* se torna portátil sendo capaz de ser utilizado em qualquer lugar que possua o *Docker* instalado. Isso garante que aplicativo sempre execute o mesmo conjunto de aplicações, independente do sistema operacional. O *Docker* possui algumas vantagens como:

- Cria novas instâncias e faz cópias do código de produção para execução local, sem a necessidade de configurar ambiente de desenvolvimento;
- Configurações e dependências em um *container* isolado, sem conflitos de aplicativos;
- Elimina as inconsistências do ambiente, ou seja, é possível executar em qualquer sistema operacional que o *Docker* esteja instalado.

2.1.5 Flask

O *Flask* é um *framework Web* escrito em *Python* baseado no WSGI[13]. O *Flask* foi projetado como uma estrutura extensível desde o início, fornecendo um núcleo

[13] *A Web Server Gateway Interface* (WSGI) é uma especificação que descreve como um servidor da *Web* se comunica com aplicativos da mesma e como os aplicativos podem ser encadeados para processar uma solicitação (https://wsgi.readthedocs.io/en/latest/what.html).

sólido com os serviços básicos, enquanto as extensões fornecem o restante (GRINBERG, 2018). Este *framework* foi desenvolvido para criação de aplicações *Web* simples, no entanto, possui a capacidade de expandir para aplicativos complexos.

O *Flask* não impõe nenhuma dependência ou *layout* do projeto, portanto, o desenvolvedor escolhe as ferramentas e bibliotecas que deseja usar[14], dessa forma, ele possui apenas os recursos necessários para sua execução, entretanto, um novo pacote pode ser adicionado para incrementar as funcionalidades da aplicação. Existem muitas extensões fornecidas pela comunidade que facilitam a adição de novas funcionalidades.

Código 2.1 – Exibindo uma mensagem em *Flask*

```
1 from flask import Flask
2 app = Flask(__name__)
3
4 @app.route("/")
5 def main():
6     return 'Olá Flask!'
7
8 if __name__ == "__main__":
9     app.run(debug=True)
```

2.2 Publicação automática de dados na nuvem

Entre as APIs de computação em nuvem, o *Google Cloud Plataform* (GCP) vem se destacando no mercado devido sua coleção de serviços de nuvem pública (CHALLITA et al., 2018). O GCP consiste em um conjunto de sistemas computacionais no qual acompanham diversos recursos, que podem ser agrupados em serviços como: autenticação no *Google Drive*, *Internet of Things* (IoT), *Structured Query Language* (SQL), entre outros. Esses serviços fornecem uma infraestrutura necessária que pode ser adicionada a uma aplicação. Todos os serviços precisam pertencer a um projeto e podem ser gerenciados através de uma *interface* gráfica no Console do GCP ou via linha de comando.

Para tornar o acesso de serviços do GCP disponíveis em uma aplicação, é necessários autenticação através de chaves de API, credencias do cliente com *OAuth 2.0*[15] ou chaves de conta de serviço, no qual, este trabalho foi utilizado o meio de autenticação *OAuth 2.0*. Para autenticar-se através do protocolo *OAuth 2.0*, pode-se

[14] Disponível em: https://palletsprojects.com/p/flask/

[15] É o protocolo padrão para autorização. O *OAuth 2.0* se concentra na simplicidade do desenvolvedor do cliente, fornecendo fluxos de autorização específicos para aplicativos (https://oauth.net/2/).

fazer o uso de bibliotecas específicas disponíveis em algumas linguagens de programação. Na linguagem *Python*, por exemplo, existe a biblioteca *oauth2client* que efetua esse procedimento para se autenticar por meio do padrão *OAuth 2.0*, ela recebe como parâmetros, as credenciais que são geradas pelo próprio serviço do GPC. Com a aplicação autenticada, é possível a manipulação de arquivos no *Google Drive*.

Com a API *gspread* é possível efetuar o manuseio de planilhas do Google. Ele pode registrar informações capturadas, pela aplicação, em planilhas e armazenalas no *Google Drive*. Para auxiliar na manipulação dos dados da planilha, existem as bibliotecas: (i) *Pandas* é uma biblioteca de código aberto, que fornece estruturas de dados de alto desempenho e operações para manipular os dados em formato de tabelas; e, (ii) a *Numpy* que possui uma larga coleção de funções matemáticas para trabalhar com *arrays* e matrizes multidimensionais. Ambas bibliotecas da linguagem *Python*.

2.3 Execução da despesa pública

Esta seção descreve o funcionamento da execução da despesa pública para o entendimento da parte do negócio no qual se insere a solução proposta neste trabalho. Inicialmente, detalha-se o conceito de orçamento público e as modalidades de aplicação do orçamento, a partir da legislação associada ao assunto. Posteriormente, são apresentadas as fases de execução da despesa pública e os tipos de documentos que formalizam cada uma destas fases.

2.3.1 Orçamento público

O orçamento público é o mecanismo utilizado pelo Governo Federal para planejar a utilização de receitas que são obtidas, principalmente, por meio da arrecadação de impostos federais. Por meio do orçamento também, é definido o recurso financeiro total que o Governo Federal espera arrecadar em receitas e fixado o valor máximo de despesas que podem ser efetuadas com dinheiro púbico (ARRUDA; ARAUJO, 2017), para que o valor gasto não seja maior que o valor arrecadado. O orçamento público ajuda na transparência das contas públicas permitindo que todo cidadão fiscalize e acompanhe a aplicação dos recursos.

A Lei Orçamentária Anual (LOA) é a lei que anualmente estabelece as despesas e as receitas que serão realizadas pelo governo no próximo ano. Ela deve ser elaborada com todo detalhamento dos gastos e receitas previstos para o ano seguinte e depende da aprovação do Congresso Nacional. Por meio da LOA, cada órgão público tem seu orçamento fixado, por meio do qual já são definidas, inclusive, os tipos

das despesas que poderão ser executadas pelos órgãos com o recurso orçamentário disponibilizado.

2.3.2 Modalidades de aplicação de recursos federais

Para que o Governo Federal preste serviços à sociedade e cumpra com os objetivos das políticas públicas, a aplicação dos recursos pode ser realizada por meio de modalidades de aplicação distintas. Uma modalidade de aplicação classifica a despesa e indica a aplicação do recurso diretamente pelas entidades de mesmo nível de governo ou indiretamente por meio de transferências para outros órgãos[16].

O orçamento público federal publicado anualmente descreve o limite máximo orçamentário que cada órgão do Governo Federal terá naquele ano. Cada órgão, ou unidade detentora do crédito orçamentário, aplica seu recurso diretamente, por meio da contratação de obras, de pagamentos de servidores, de compras de livros didáticos para estudantes; entre outros gastos. Além da aplicação direta, a unidade detentora do crédito pode realizar a descentralização de recursos para outros órgãos, para terceirizar a realização de algum serviço.

A descentralização de créditos é a forma utilizada na Administração Pública Federal para se transferir o poder do crédito orçamentário de uma unidade gestora (UG) para outra UG do mesmo órgão ou de órgão distinto. Dessa forma, a descentralização pode ser classificada como **externa** quando a movimentação dos créditos orçamentários é feita entre órgãos diferentes e **interna** quando a movimentação de créditos ocorre entre UGs de um mesmo órgão.

Conforme mencionado anteriormente, o contexto deste trabalho é na transparência ativa da execução financeira de recursos no âmbito da Enap (EV.G), que recebe recursos de órgãos para disponibilização de cursos e estabelece parcerias com outros órgãos para a realização de ações de pesquisas e desenvolvimento, sendo ambos tipos de órgãos chamados de instituições parceiras.

O repasse de recursos entre as instituições parceiras e a Enap é formalizado por meio de um Termo de Execução Descentralizada (TED), definido no Decreto nº 8.180, de 30 de dezembro de 2013, como: "o instrumento por meio do qual é ajustada a descentralização de crédito entre órgãos e/ou entidades integrantes, com objetivo executar ações de interesse da unidade orçamentária descentralizadora". Um TED contém uma descrição abrangente do objeto a ser executado a partir dos recursos descentralizados, das metas a serem alcançadas, das etapas e dos recursos envolvi-

[16] Manual do SIAFI - Classificações orçamentárias está disponível em: https://conteudo.tesouro.gov.br/manuais/index.php?option=com_content&view=article&id=1567:020332-classificacoes-orcamentarias&catid=749&Itemid=376

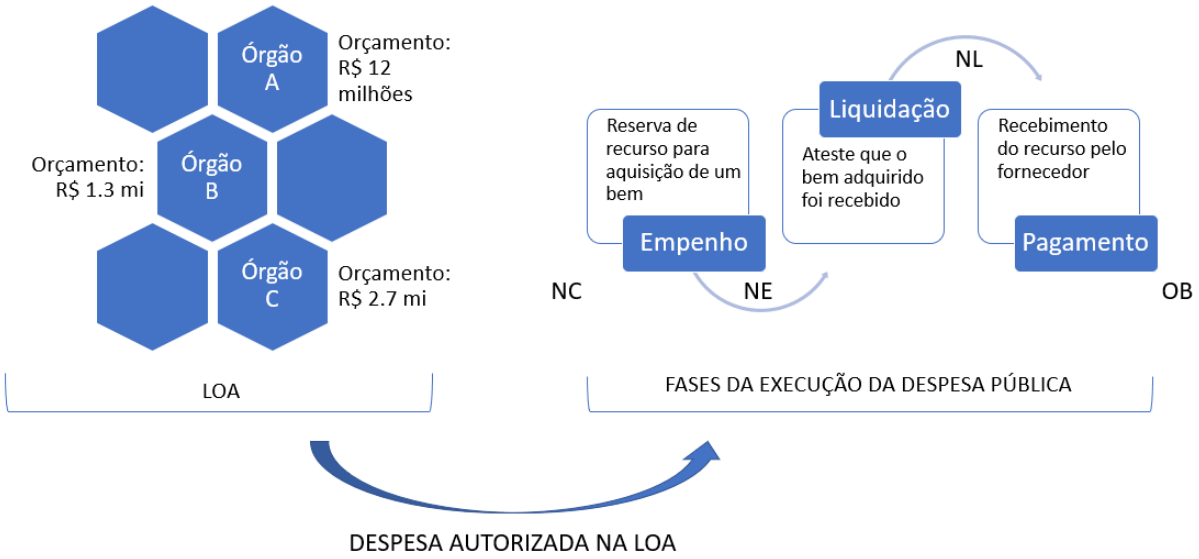
dos.

Após a assinatura do TED, a Enap é autorizada então a descentralizar o crédito previsto para a instituição parceira, sendo esta operação de descentralização formalizada por meio de um documento denominado **Nota de Crédito ou Nota de Movimentação de Crédito** (NC), apresentado em detalhas na Seção 2.3.4.

2.3.3 Fases da execução da despesa pública

Conforme estabelecido na Lei nº 4.320, de 17 de março de 1964, para executar as despesas públicas, ou seja, para aplicar o dinheiro arrecadado na prestação dos serviços públicos, são previstas as 3 (três) etapas da despesa ilustradas na Figura 4 e descritas a seguir:

Figura 4 – Fases da execução da despesa pública



Fonte: Elaborado pelo autor.

- **Empenho**: é a etapa na qual o Governo Federal reserva o dinheiro a ser pago quando for concluído um serviço ou a entrega de um produto, ou seja, é uma garantia que se faz ao fornecedor de que possua o crédito necessário para o pagamento da despesa (VIEIRA; PIOLA, 2016). No âmbito da Administração Pública, é proibida a realização de uma despesa sem empenho prévio.

  O empenho pode ser visto pelo fornecedor como uma garantia dada a ele, com valor devidamente deduzido do orçamento, de que o contrato existente será pago, desde que observadas as cláusulas contratuais. No entanto, é impor-

tante notar que o empenho pode vir a ser cancelado se não houver a entrega, por parte do fornecedor do Governo, do item estabelecido no contrato.

- **Liquidação**: Após a entrega do bem ou prestação do serviço previstos no contrato (VIEIRA; PIOLA, 2016), vem a fase da liquidação da despesa. A liquidação é o ateste, por parte do Governo, de que um bem adquirido foi entregue corretamente pelo fornecedor, ou que a etapa de uma obra foi concluída como acordado. A finalidade da liquidação da despesa é a de apurar se o item do gasto foi realizado na forma do contrato, verificando elementos como: (i) a origem e o objeto do que se deve pagar; (ii) o valor exato a ser pago; e (iii) o credor a quem se deve pagar o valor para que se extingua a obrigação.

- **Pagamento**: Depois que uma despesa for liquidada, acontece a etapa do pagamento que é o último estágio de execução da despesa pública. Esta etapa consiste na entrega de numerário ao credor da Administração Pública (NASCIMENTO, 2017), ou seja, etapa na qual os fornecedores e as empresas contratadas pelo governo recebem de fato o dinheiro pelo que realizaram. Os métodos de pagamentos precisam ser transparentes, seguros, coordenados e eficiente na saída dos recursos da Conta Única do Tesouro Nacional.

  A realização do pagamento é realizada por meio do SIAFI, através de emissão de Ordem Bancária (OB), documento que possui várias espécies e características próprias, variando de acordo com o tipo de pagamento a ser realizado.

2.3.4 Documentos das fases da despesa pública

Com o objetivo de administrar os créditos utilizados, para formalizar cada uma das fases da execução das despesas públicas, detalhadas na Seção 2.3.3, são previstos vários tipos de documentos. A seguir, apresenta-se apenas os tipos de documentos existentes que serão utilizados no contexto deste trabalho.

- **Nota de Crédito (NC):** este documento é utilizado para registrar a movimentação de créditos interna e externa e suas anulações[17], ou seja, é usado para formalizar a realização de transferências de crédito orçamentário entre unidades gestoras. A NC é o documento utilizado para registrar eventos vinculados a movimentação interna e externa de créditos para execução da despesa pública, no qual se descreve os tipos de despesas que podem ser realizadas com aquele recurso e seus respectivos valores máximos.

[17] Manual do STN - Documentos utilizados pelo sistema, disponível em $https://conteudo.tesouro.gov.br/manuais/index.php?option=com\_content\&view=article\&id=3102:020500-documentos-utilizados-pelo-sistema$

- **Nota de empenho (NE):** é o documento usado para registrar o comprometimento de despesas do recurso disponível em uma unidade gestora. A nota de empenho está vinculada a, no mínimo, uma nota de crédito. Cabe destacar também, que é prevista a possibilidade de se **reforçar** uma nota de empenho, procedimento pelo qual se aumenta os valores disponibilizados originalmente, ou de se **anular** uma nota de empenho, em função de algum compromisso desfeito.
- **Ordem Bancária (OB):** é documento que serve para registrar a formalização de um pagamento. Uma OB é o mecanismo de registro do pagamento de compromissos, gerado ao final da etapa de pagamento da execução da despesa pública.

2.4 Gestão da execução financeira da Enap (EV.G)

Para conceituar a área do negócio no qual este trabalho se insere, nesta seção detalha-se o processo de execução financeira de recursos descentralizados pelas instituições parceiras da Enap (EV.G) e a transparência ativa praticada pela instituição.

2.4.1 A Escola Virtual.Gov - EV.G

A EV.G é uma iniciativa coordenada pela Enap para centralizar a oferta de cursos à distância de capacitação profissional a servidores, empregados públicos de todo o país, bem como cidadãos e estrangeiros interessados nos cursos.

A EV.G consiste em uma solução completa para oferta de cursos à distância, composta de infraestrutura de hospedagem de cursos, sistema de gestão acadêmica próprio, catálogo de cursos, base de dados de cursos, alunos e capacitações, serviços de atendimento ao usuário de primeiro nível, entre outros. Os cursos ofertados no âmbito da EV.G são oriundos das instituições associadas. A adesão à EV.G dá-se pela assinatura de termo de adesão ao protocolo de intenções, por meio do qual a instituição associada assume um dos dois papéis descritos a seguir:

- **Conteudista**: instituição dotada de competência técnica em assunto específico, responsável pela produção de cursos, com interesse nos serviços de hospedagem fornecidos pela EV.G.
- **Certificadora**: instituição dotada de legitimidade acadêmica para expedição dos certificados de cursos disponibilizados na EV.G. Pode oferecer cursos de forma independente ou associada a alguma instituição conteudista parceira no desenvolvimento do conteúdo.

- **Patrocinadora**: instituição interessada em apoiar e incentivar as ações da EV.G, associando sua marca à Escola.
- **Gestora**: instituição interessada em apoiar e incentivar as ações da EV.G, motivada pelo interesse em dados estratégicos de capacitações ligadas às políticas públicas que estão sob sua competência institucional.

A depender do papel desempenhado, há contrapartidas previstas para as instituições associadas, que podem desde assumir os custos relativos ao desenvolvimento e revisão de cursos até à realizar repasse de recursos voltados à manutenção de serviços de hospedagem.

Diante das múltiplas formas de associações à EV.G, destaca-se que este trabalho se insere no contexto da divulgação de informações relativas aos recursos financeiros repassados ou recebidos pela Enap (EV.G), para o desenvolvimento dos cursos à distância disponibilizados pela EV.G.

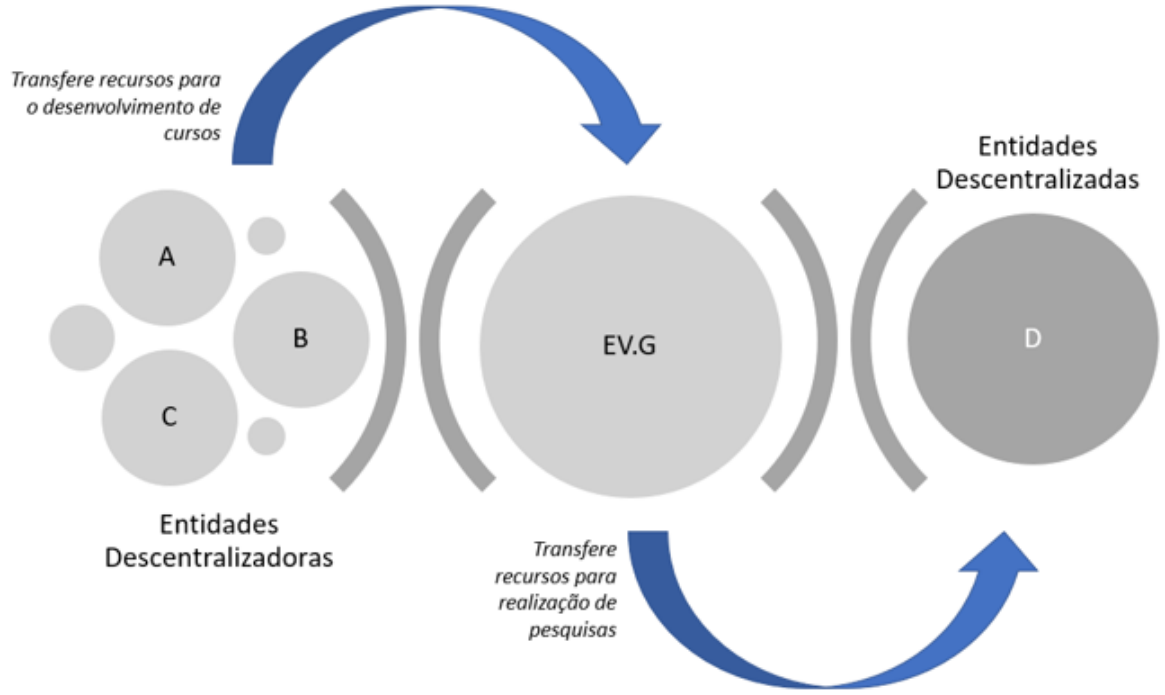
2.4.2 Papeis da Enap (EV.G) na gestão dos recursos

Como ilustrado na Figura 5, a Enap (EV.G) pode desempenhar dois papéis distintos, a depender do mecanismo de descentralização de recursos usados. Quando a Enap (EV.G) recebe recursos de outros órgão para a realização de cursos em seu ambiente, ela está exercendo o papel de **entidade descentralizada**. Por outro lado, quando a Enap (EV.G) repassa recursos para que outro órgão desenvolva pesquisas para ela, o papel exercido é de **unidade descentralizadora**.

Neste contexto, é possível definir: (i) **entidade descentralizadora** como a responsável pelo acompanhamento e a fiscalização de créditos descentralizados, ou seja, unidade repassadora de recursos; e (ii) **entidade descentralizada** como a responsável pela operacionalização dos créditos e execução dos recursos repassados, isto é, trata-se da unidade que recebe recursos de outra unidade para a realização de atividades específicas em benefício da unidade descentralizadora dos recursos.

O processo de transferências e aplicação de recursos é feito por meio de emissão do documento financeiro **NC**, descrito na seção anterior. Quando a Enap (EV.G) atua como entidade descentralizadora, para que possa disponibilizar a transparência ativa na aplicação os recursos, ela precisa ter, sob sua gestão, todos os demais documentos de formalização da despesa emitidos pela unidade descentralizada, como NEs e OBs. Inevitavelmente, tem-se um grande volume destes tipos de documentos financeiros, tornando-se necessário existir uma gestão automatizada de todos eles, tenham eles sido emitidos pela própria Enap (EV.G) ou por unidades para quem a Enap (EV.G) descentralizou recursos.

Figura 5 – Transferências de recursos entre a Enap (EV.G) e instituições parceiras



Fonte: Elaborado pelo autor.

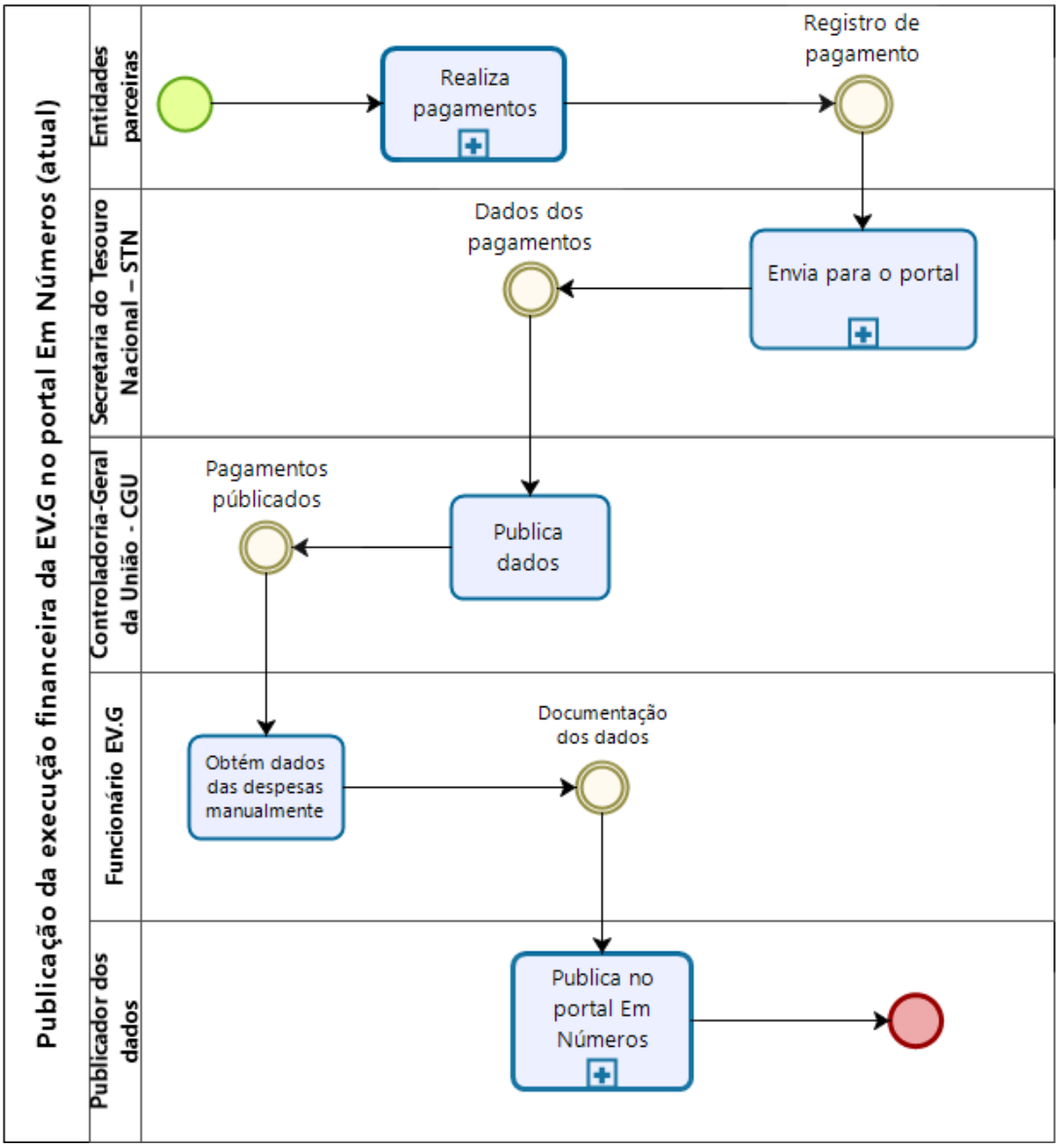
2.4.3 Gestão Financeira da Enap (EV.G)

Para tornar clara a contribuição deste trabalho, detalha-se nesta seção, o funcionamento atual do mecanismo de gestão financeira e de transparência ativa utilizado atualmente pela Enap (EV.G). Posteriormente, na Seção 4, será apresentada o funcionamento do mesmo processo, a partir da proposta de solução que procura resolver, de forma automática, algumas questões associadas ao registro dos dados de movimentação dos recursos da Enap (EV.G).

Atualmente, o funcionamento deste processo se dá como descrito a seguir. Para que as informações da utilização de recursos no âmbito da Enap (EV.G) possam ser publicadas, os servidores responsáveis por esta ação precisam, rotineiramente, solicitar os dados da execução dos recursos às entidades parceiras e atualizar manualmente estas informações em um sistema próprio.

Como ilustrado na Figura 6, inicialmente, a Enap (EV.G) descentraliza os recursos para as entidades parceiras que, realizam sua execução ao longo do desenvolvimento das atividades para as quais foram contratadas, registrando toda a movimentação orçamentária e financeira no Sistema Integrado de Administração Financeira (SIAFI). Periodicamente essas movimentações são enviadas pela Secretaria do Tesouro Nacional (STN) ao Portal da Transparência. Os tipos de informações enviadas

Figura 6 – Gestão financeira e de transparência ativa da Enap (EV.G) atual



Fonte: Elaborado pelo autor.

pela STN incluem os documentos das despesas, tais como, notas de crédito, notas de empenho, ordem de pagamento, entre outros. Após o envio, os dados dos pagamentos são publicados no portal pela Controladoria geral da União (CGU).

Embora os dados necessários à publicização da informação pela Enap (EV.G) sejam publicados periodicamente pela STN no Portal da Transparência, a Enap (EV.G) solicita as notas de empenho às entidades parceiras, para realizar uma atualização manual de sua base de dados de registros de pagamento. Adicionalmente, cabe destacar que a busca dos dados através do empenho torna-se demorado, pois algumas informações não são disponíveis para o *download* no Portal da Transparência, com isso, o profissional precisa procurar meio de copiar esses dados, tornando oneroso o seu trabalho nesta atualização. Por fim, é efetuado a publicação destes dados em seu

portal Em Números[18]. Como esclarece a própria EV.G, o portal Em Números:

> Disponibiliza dados e informações relativas à capacitação de servidores públicos, temas de maior interesse, distribuição nacional das capacitações, entre outros indicadores, de forma aberta, tempestiva e dinâmica. O acesso facilitado a dados reais e precisos permite não apenas o controle social, mas também a análise de informações de acordo com as mais variadas necessidades, por meio de *dashboards* que apresentam visualmente as informações de interesse para o cidadão comum, público alvo do portal.

[18] Disponível em https://emnumeros.escolavirtual.gov.br/

3 SOLUÇÕES QUE UTILIZAM *WEB SCRAPING*

3.1 Jusbrasil

O Brasil é um país regido por um conjunto de atos normativos interligados (leis, decretos, etc), organizados dentro do princípio da hierarquia jurídica. Em função do grande volume de atos e da dubiedade de entendimentos associados a eles, são poucos os cidadãos comuns que compreendem o funcionamento deste universo. Neste sentido, a Jusbrasil surge como uma plataforma tecnológica para conectar pessoas à justiça por meio de advogados e informação jurídica, com abordagens acessíveis a todos.

A Jusbrasil é a primeira rede social orientada a conteúdo jurídico e o portal jurídico mais visitado do mundo, segundo o próprio *site*[1]. A partir de consulta textual, usando palavras-chaves, associadas ao tema de interesse, um cidadão comum tem acesso a processos similares que tramitaram (ou estão em trâmite) na justiça, cujos pareceres podem ser usados pelos interessados na resolução de suas próprias questões. A plataforma também permite que se contrate um advogado especializado no tema abordado, simplificando a vida do usuário da plataforma. Os conteúdos do Jusbrasil são replicados a partir dos *sites* dos tribunais. Estas informações são coletadas e estruturadas pelos engenheiros, utilizando técnicas que fazem obtenção de dados automática, em seguida, publicadas no Portal.

No portal da Jusbrasil, ilustrado na Figura 7, são compartilhados gratuitamente conhecimento via artigos, notícias, jurisprudência, legislação e diários Oficiais, para que os cidadãos aprendam sobre seus direitos e deveres. A Jusbrasil tem como objetivo: (i) fazer com que as pessoas entendam seus direitos e deveres; (ii) tornar melhor e mais acessível o acesso aos advogados; (iii) proporcionar que as pessoas e os advogados possam se encontrar facilmente; (iv) ter acesso à justiça com mais eficiência.

3.2 Escavador

Escavador é uma plataforma que compartilha inúmeras e diferentes informações por meio da tecnologia. Esta plataforma agrupa dados que estão espalhados em *sites* oficiais, para facilitar o acesso dos conteúdos ao cidadão. Todo conteúdo do *site* é coletado de forma automática por meio de *bots* que atuam nas fontes públicas, seja pela Lei de Acesso à Informação ou de fontes jurídicas e de instituições

[1] Disponível em $https://sobre.jusbrasil.com.br/comofazemos2$

Figura 7 – Página inicial do *site* Jusbrasil

Fonte: https://www.jusbrasil.com.br

públicas (ESCAVADOR, 2019). O objetivo do escavador é capturar esses dados, que são de difícil compreensão para algumas pessoas, e torná-los estruturados para que pessoas comuns sejam capazes de compreender o que acontece no Brasil.

O Escavador é bem conhecido na comunidade acadêmica por disponibilizar informações sobre o currículo de pessoas, extraindo dados da Plataforma Lattes[2] do CNPq (Conselho Nacional de Desenvolvimento Científico e Tecnológico). Normalmente, no Brasil, quando se consulta o nome completo de um indivíduo nos mecanismos de busca, uma das primeiras respostas é um link para o currículo dele no *site* do Escavador.

Por meio do *site* do Escavador, cuja página inicial pode ser visualizada na Figura 8, as pessoas podem ter acesso aos diários oficiais e artigos. Além disso, o

[2] Disponível em $http://lattes.cnpq.br$

site disponibiliza uma seção monitoramento, onde se é possível encontrar informações de pessoas, empresas, processos e outros. No monitoramento pode ser efetuado um cadastro para ser notificado quando o nome da pessoa for mencionado em algum Diário Oficial. Essa notificação é gratuita, com algumas limitações, porém, existe uma versão completa com recursos adicionais.

Figura 8 – Página inicial do *site* Escavador

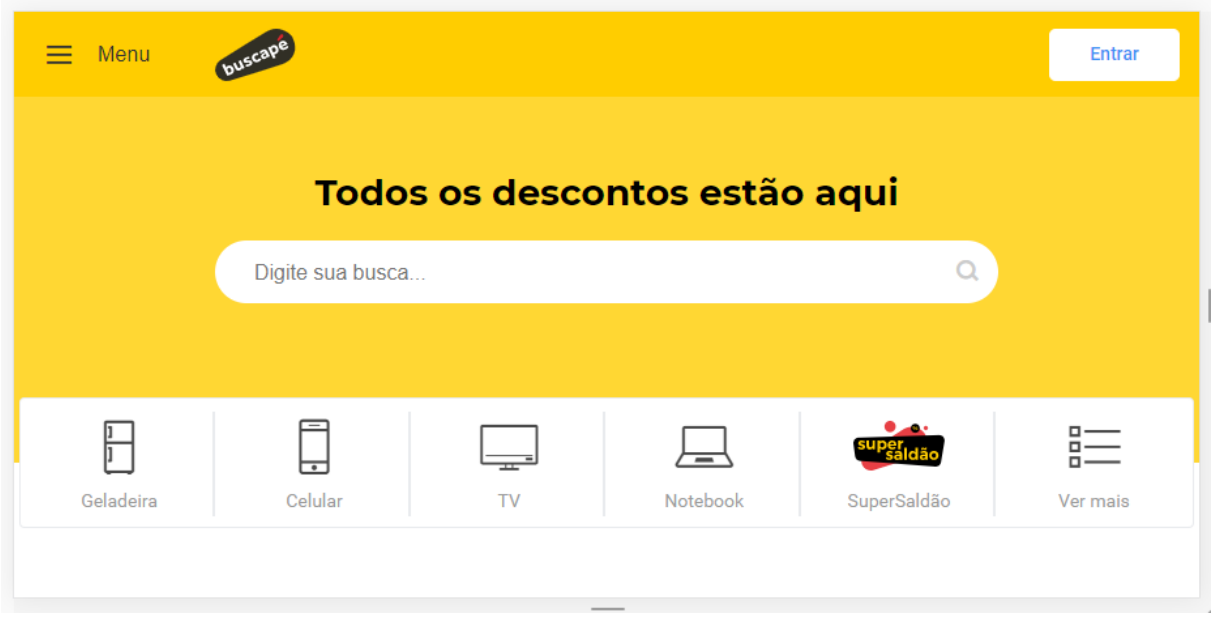
Fonte: https://www.escavador.com

3.3 Buscapé

O *site* Buscapé, ilustrado na Figura 9, é uma plataforma gratuita que trabalha juntamente com outras marcas nas diversas etapas de uma compra digital. O Buscapé compara preços, lojas e produtos. Após a comparação, é exibido as informações dos produtos anunciados com menor preço das lojas mais confiáveis. O objetivo do Buscapé é permite que o usuário não perca tempo procurando em várias lojas por preços ou informações importantes de um determinado produto.

No Buscapé existe uma seção para desenvolvedores. Os desenvolvedores podem ter acesso algumas APIs que o Buscapé disponibiliza, tais como: API de pedidos e API de ofertas. A ii) API de pedidos permite que o lojista venda seus produtos através do *site* Buscapé. Para isso, é necessário o desenvolvimento de um *web service* que será utilizado pelo carrinho de compras do buscapé para informar que um

Figura 9 – Página inicial do *site* Buscapé



Fonte: https://www.buscape.com.br

novo pedido foi realizado ou que um pedido teve seu pagamento confirmado. A i) API de oferta serve para integração no envio de ofertas entre o buscapé e o lojista, com objetivo de diminuir o tempo de atualização das ofertas no *site*. Para ter acesso a essa API é necessário possuir um *token* de acesso.

3.4 Trivago

O Trivago[3] é um buscador de preços voltado para hospedagem, com propósito de que o cliente possa aproveitar dos serviços disponíveis de boa qualidade e com o menor preço (KAUR, 2020). Ele permite que o usuário compare os preços de quartos em vários *sites* e hotéis antes da reserva. A uso da plataforma é gratuito e estão disponíveis nas plataformas *web*, *Android* e *iOS*.

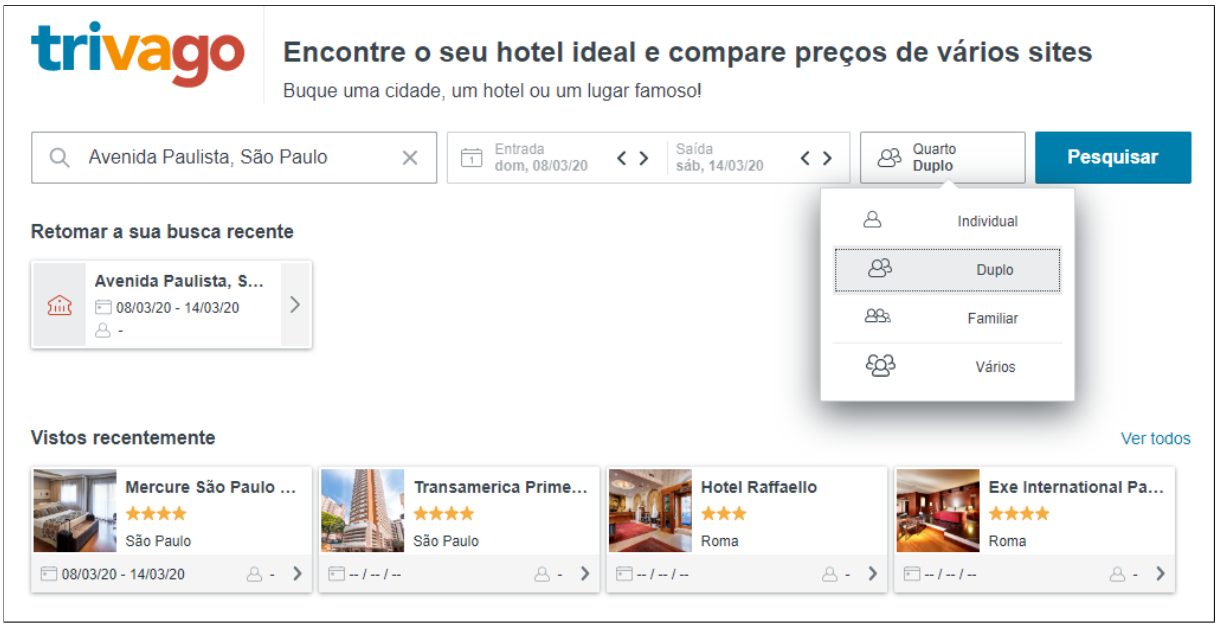
Os hotéis realizam cadastros de preços em diversos *sites*, como decolar, booking, hoteis.com entre outros. Os registros dos valores do mesmo hotel podem variar para cada *site*, desse modo, o Trivago faz a seleção, com base nesses dados, resultando na busca pelos quartos com menor preço.

Na plataforma do Trivado, mostrada na Figura 10, o usuário consegue buscar hospedagem através do nome do hotel, endereço de destino, data de entrada, data de saída e tipo do quarto. A aplicação retornará os preços mais acessíveis de acordo com as informações passadas pelo usuário, além disso, o usuário pode filtrar algumas

[3] Disponível em $https://www.trivago.com$

informações resultantes da busca, como ordenação (preço, distância, por exemplo), avaliação do hotel, complementos na hospedagem (piscina, estacionamento, café da manhã) entre outros.

Figura 10 – Página inicial do *site* Trivago



Fonte: https://www.trivago.com.br

4 AUTOMATIZAÇÃO DA GESTÃO FINANCEIRA DA ENAP (EV.G)

O trabalho aqui apresentado tem como objetivo simplificar o processo de atualização do Portal Em Números, automatizando as atividades de alimentação manual hoje realizadas pela Enap (EV.G) e publicando as informações que foram obtidas na fonte de dados do portal. Antes do início do trabalho, foi solicitada uma autorização para a realização desta pesquisa junto à Enap, sendo devidamente aprovada por meio do documento apresentado no Anexo A.

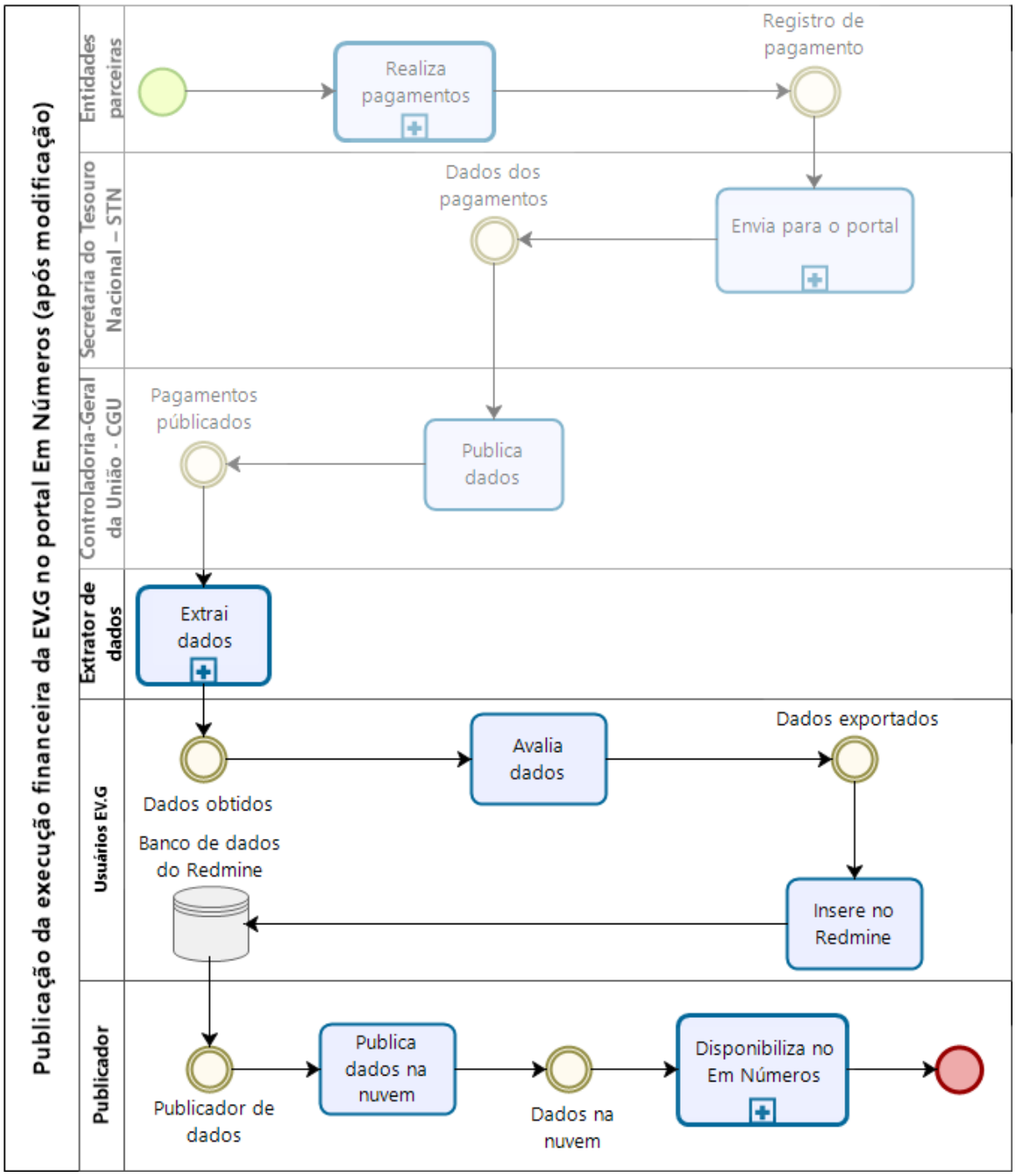
Com a inserção da proposta apresentada neste trabalho, o processo será alterado essencialmente na parte de obtenção e registro dos dados no sistema próprio da Enap (EV.G), que será automatizado na solução proposta. Na Figura 11, a seguir, ilustra-se como fica o processo existente atualmente após a aplicação da solução proposta.

A solução proposta atua principalmente em duas partes distintas: na extração de dados do Portal da Transparência e em sua inserção no sistema de armazenamento da Enap (EV.G). Nesta proposta, o rastreio dos documentos é realizado por meio de *bots* da *web* que buscam as informações de interesse, oriundas do Portal da Transparência, extraindo e estruturando os conteúdos obtidos em seguida. Os documentos obtidos, por sua vez, são armazenados na nuvem e publicados no portal Em Números. A visualização no portal, com a implementação desta proposta, continua sendo por meio de *dashboards*, por se tratar de uma linguagem mais direta e acessível que facilita a compreensão e possibilita que qualquer pessoa possa entender os dados apresentados.

Quanto às tecnologias a serem utilizadas, a proposta de automatização foi desenvolvida na linguagem de programação *Python*, que disponibiliza um conjunto de *frameworks* e bibliotecas para propósitos específicos. Para auxiliar nesse trabalho, foram utilizados os *frameworks Scrapy*, *Flask* e *gspread*, como também as bibliotecas *Pandas*, *Numpy* e *oauth2client*, todas da linguagem *Python*.

O *framework Scrapy* disponibiliza vários recursos com suporte à tecnologia *web crawling* e *web scraping*. A utilização do *Scrapy* teve como finalidade aplicar as estruturas de rastreamento e raspagem de dados, com objetivo de solicitar as informações desejadas do portal da Transparência. O *Flask* foi utilizado para o desenvolvimento da API por se tratar de um *framework* com modelo simples e a capacidade de expansão para aplicações complexas. A biblioteca *Pandas*, por outro lado, ajudou no tratamento e na manipulação de uma grande quantidade de dados, trabalhando as informações em formato de tabelas e tornando o manuseio mais fácil.

Figura 11 – Gestão financeira e de transparência ativa da Enap (EV.G) com a proposta



Fonte: Elaborado pelo autor.

O *Numpy*, outra biblioteca *Python* utilizada, fornece manipulação de números em formato de lista. O *oauth2client* contribuiu com a autenticação no *Google Drive*, por meio do serviço do *Google Cloud* e o *gspread* foi utilizado para o manipulação das planilhas armazenadas no *Google Drive*.

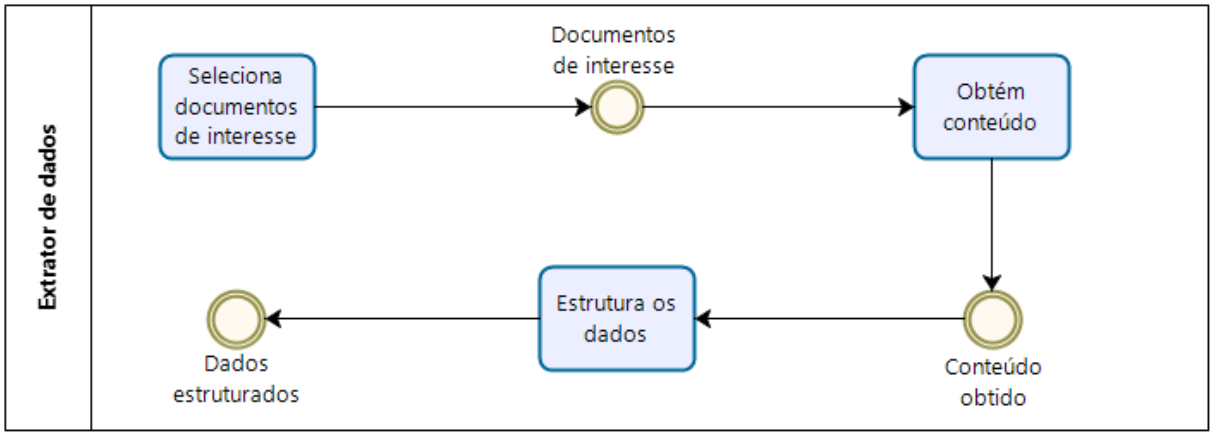
4.1 Extração de dados do Portal da Transparência

O processo de extração de dados, apresentado na Figura 12, é realizado através de *bots*, utilizando as técnicas de *web crawling* e de *web scraping* que fazem a rastreabilidade e a obtenção dos dados, respectivamente. O acionamento destes *bots* se dá a partir de uma *Application Programming Interface* (API) criada especialmente para o desenvolvimento deste trabalho.

Os *bots* fazem a busca dos registros no Portal da Transparência e selecionam quais são os documentos de interesse. Posteriormente, eles realizam a varredura dos dados, obtendo os conteúdos necessários. Para que o servidor não seja sobrecarregado, os *bots* foram desenvolvidos incluindo um tempo de espera entre uma requisição e outra.

Os dados são obtidos a partir de várias URLs distintas, resultando em um conjunto de arquivos separados. Os dados de pagamento (OB) são obtidos a partir de uma URL e os de nota de empenho (NE) de outra. Então, faz-se necessário que a aplicação contemple um serviço responsável por estruturar os dados em um arquivo único, disponibilizando esse arquivo para que o usuário - funcionário da Enap (EV.G) que anteriormente obtinha os dados de forma manual - efetue o seu *download*.

Figura 12 – Processo de extração de dados



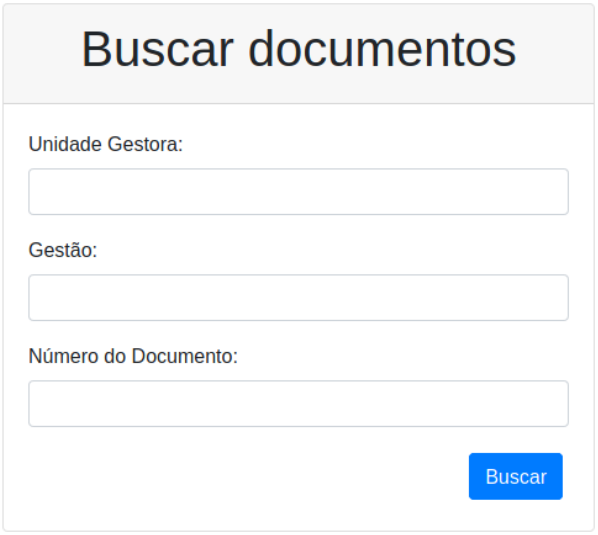
Fonte: Elaborado pelo autor.

Na solução proposta, foi criada uma aplicação *web* que aciona os *bots*. Por meio desta aplicação, ilustrada na Figura 13, o usuário informa os parâmetros de busca que são fundamentais para a realização das consultas, tais como:

- **código da UG** responsável pela descentralização ou pela execução dos créditos recebidos;

- **código gestão** que trata-se de uma codificação utilizada no sistema SIAFI para permitir que uma mesma UG possa movimentar, de forma individualizada, os recursos comuns de manutenção recebidos do Tesouro Nacional e os recursos com finalidade específica provenientes de um fundo especial[1];
- **número do documento** que deseja solicitar, a saber: ordem bancária (OB) ou nota de empenho (NE). Com estes números, a aplicação consegue realizar a busca, iniciando o processo de extração de dados.

Figura 13 – Interface *web* para o usuário



Fonte: Elaborado pelo autor.

O processo de extração dos dados se inicia com a etapa de seleção dos documentos, utilizando-se como base as informações supramencionadas repassadas pelo usuário, tais como: código da unidade gestora, código da gestão e número do documento. Com essas informações, os *bots* criam a URL para efetuar a requisição no documento de interesse. As URLs que são construídas por esse trabalho seguem um padrão, variando apenas o código do documento nelas contidos. Dessa forma, a aplicação consegue montá-las através dos parâmetros obtidos do usuário.

Após a requisição realizada usando a URL formada, utiliza-se o *Splash* para a renderização dos dados que são obtidos do servidor, por meio do *JavaScript*, tem como resposta o documento HTML com todas essas informações. Terminado esse procedimento, a raspagem dos conteúdos necessários continua. Para a busca dos documentos, podem ser utilizados os três tipos de *bots* descritos a seguir:

[1] Glossário do Senado Federal, disponível em $https://www12.senado.leg.br/orcamento/glossario/gestao$

- ***Bot* do pagamento**: este busca os dados de um pagamento (OB) específico. São retornadas as informações do pagamento e todos os favorecidos[2];
- ***Bot* do empenho**: este busca os dados do empenho, devolvendo as informações sobre ele.
- ***Bot* dos pagamentos**: este busca e retorna todos os pagamentos (OBs) vinculados a um empenho.

No Código 1 do Apêndice A é mostrada a codificação do ***bot* dos pagamentos** que obtém as informações das OBs vinculadas a um empenho. Na execução inicial do *bot* dos pagamentos, é construída a URL e realizada a requisição.

No Código 4.1, mostrado a seguir, podem ser observadas as funções que são executadas inicialmente. A princípio, na função `__init__`, é atribuído o código do documento recebido pela API, que, por sua vez, foi construído com os parâmetros passados pelo usuário. Em seguida, por meio da função `getDataEmpenho`, são obtidas as informações do empenho para agrupar com os dados das OBs. Após a execução desta função, a `start_requests` é executada.

Código 4.1 – Funções de inicialização do *bot* de pagamento

```
1  URL = 'http://www.portaltransparencia.gov.br/despesas/pagamento/{}?
       ordenarPor=fase&direcao=asc'
2
3  def __init__(self, code='', *args, **kwargs):
4      self.codigo = code
5      #Obtem informações do empenho
6      self.getDataEmpenho(self.codigo)
7      super(PagamentosEmpenhoSpider, self).__init__(*args, **kwargs)
8
9  def start_requests(self):
10     codes = self.getCodesPay(self.codigo)  #Pega uma lista de códigos de OB
           relacionada ao empenho
11     for code in codes:
12         yield SplashRequest(
13             url = self.URL.format(code),
14             callback = self.parse, args={'wait': 5}
15         )
```

A função `start_requests` é responsável por realizar as requisições. Inicialmente, são adquiridos todos os códigos relacionados ao empenho que são extraídos

[2] Em uma mesma ordem de pagamento (OB) podem haver mais de um favorecido (CPF ou CNPJ) associado.

de um arquivo gerado pelo ***bot* do empenho**, responsável por buscar as informações do empenho.

Posteriormente, são realizadas várias requisições nas URLs construídas com o código de cada OB, para buscar os documentos desejados. Além disso, é passado um valor decimal para o argumento `wait`, informando o tempo (em segundos) para aguardar a renderização da página. Após a conclusão da requisição, é executada uma função de *callback*, denominada `parse`, que fará o procedimento de captura dos dados.

Com base nesse procedimento realizado pelo *bot*, continua-se a execução do mecanismo para obtenção dos dados. Nesse procedimento é realizada a obtenção de conteúdo por meio da raspagem de dados (*web scraping*). Nessa etapa, são obtidos os dados relevantes do documento utilizando-se seletores (através do CSS ou *XPath*). Em virtude do HTML obtido na requisição possuir algumas informações irrelevantes, no processo de raspagem é efetuada a seleção dos dados necessários ao usuário.

No procedimento de captura de dados, o *bot* segue com a execução da função `parse`, ilustrada no Código 4.2, que recebe a resposta da requisição como um documento HTML renderizado. A partir disso, são obtidos todos os favorecidos daquela OB, e, em seguida, são utilizados métodos para efetuar a raspagem de dados, selecionando os dados necessários e os armazenando em um arquivo CSV. Cabe destacar que a função `parse` é executada para cada requisição efetuada pelo *bot*, ou seja, para cada código de OB repete-se o mesmo procedimento de raspagem dos dados.

Código 4.2 – Função do procedimento de raspagem de dados do *bot*

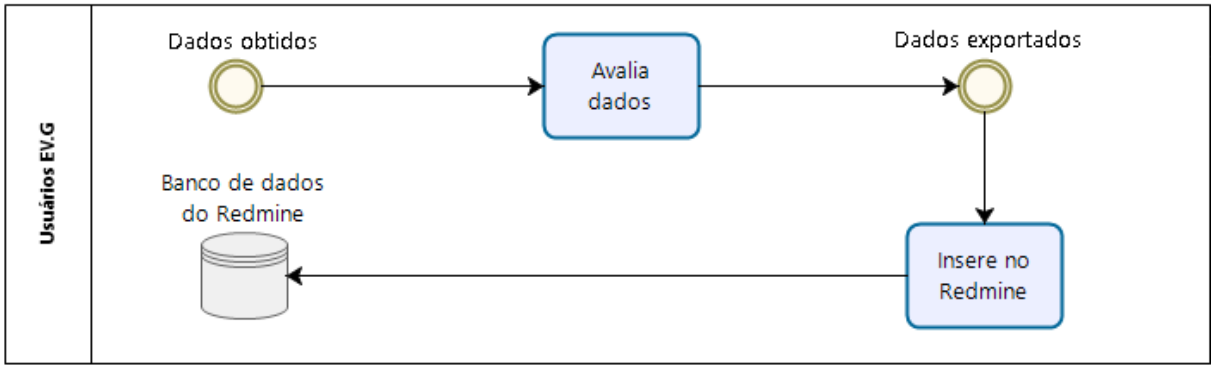
```
1  def parse(self, response):
2      code = response.url.split('?')[0].split('/')[-1:][0]
3      # Lista de favorecidos da OB
4      self.codigoOB = code
5      os.system(self.SRC_FAVORECIDOS.format(code))
6      if not self.init:
7          self.init = True
8          header = response.css('section.dados-tabelados strong::text').
               extract()
9          yield Portalltem(row = self.H_EMPENHO + header)
10
11     row = response.css('section.dados-tabelados span::text').extract()
12     yield Portalltem(row = self.R_EMPENHO + row)
```

Visto que a obtenção dos dados é selecionada individualmente pelo *bot*, o conjunto de informações obtidas está fragmentado. Com isso, aplica-se o método de

estruturação. O procedimento para estruturar os dados é feito no agrupamento das informações obtidas e armazenadas em arquivos CSV. A estruturação dos dados é efetuada toda vez que o *bot* realiza a raspagem. Após os dados serem estruturados, eles são disponibilizados para *download*.

Com a obtenção dos dados estruturados, realizado pelo processo anterior, segue-se com o armazenamento dessas informações. O funcionário da Enap (EV.G) é o usuário responsável pelo processo de inserção de dados ilustrado na Figura 14. Em resumo, este processo se inicia com a aquisição dos dados por meio da utilização da aplicação *web* proposta. Os dados coletados por meio da aplacação são posteriormente avaliados pelo usuário público-alvo da aplicação (funcionário da Enap). Após os dados serem validados, o usuário obtém os arquivos que são importados no sistema de gerenciamento da Enap (EV.G), que é um *Redmine* customizado para àquela instituição.

Figura 14 – Processo de armazenamento dos dados



Fonte: Elaborado pelo autor.

O *Redmine* é uma plataforma *web open source* que disponibiliza recursos para gerenciamento de projetos. Embora o *Redmine* seja uma ferramenta concebida para gerenciar atividades de um projeto de desenvolvimento de software, o recurso de criação de campos personalizados associados aos itens de um projeto, a torna uma ferramenta bastante flexível, podendo ser usada para o registro de dados em contextos onde não existem sistemas de informação específicos para resolver um determinado problema.

No contexto da Enap (EV.G), o *Redmine* é utilizado para o gerenciamento das informações financeiras, por meio do qual se registram todos os dados associados a empenhos e pagamentos realizados por meio dos TEDs existentes com órgãos parceiros. Atualmente estas informações de OB e NE são inseridas no sistema individualmente e de forma manual. Com o mecanismo de obtenção dos dados do Portal da Transparência, criado neste trabalho, o usuário passará a realizar a importação em lote no *Redmine* dos dados obtidos por meio do *bot*, visto que a aplicação já dispõe

de uma funcionalidade para este fim.

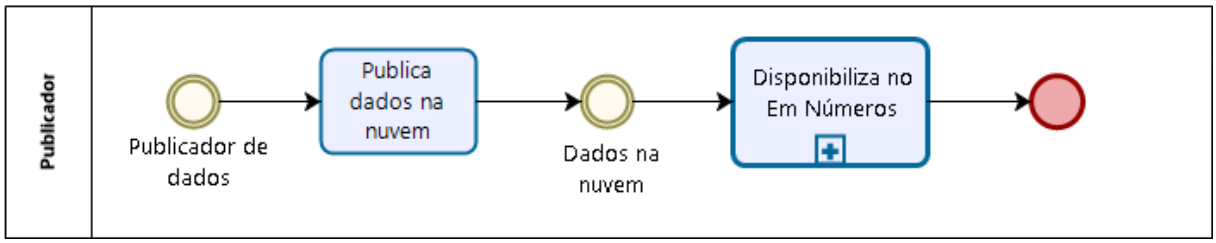
Após o processamento de dados realizado pelo *bot*, o usuário avalia os documentos para validar a integridade dos dados. Este procedimento consiste na visualização dos dados deforma que o usuário possa verificar se estes estão conforme o solicitado. Com a validação dessas informações, o usuário pode realizar um *download* dos arquivos que são disponibilizados (no formato CSV).

Como mencionado, a Enap (EV.G) utiliza o *Redmine* como sistema para o gerenciamento dos documentos financeiros. Atualmente, os documentos financeiros são cadastrados um a um na ferramenta, pelo funcionário responsável, após a consolidação dos dados por meio de um processo manual de busca e agrupamento de informações. Esta forma de trabalho é lenta e suscetível a erro humano na inserção individualizada de cada item. Por meio do *bot* desenvolvido neste trabalho, este procedimento pode vir a ser rapidamente realizado, dependendo muito mais da velocidade de acesso à internet do que da agilidade do usuário que está responsável pela ação.

4.2 Armazenamento de dados em nuvem

Com os documentos armazenados no banco de dados do *Redmine*, a próxima etapa é a publicação dos dados em uma ambiente de armazenamento na nuvem. Nesse passo consiste em uma aplicação, na qual, é executada diariamente com a finalidade de buscar os documentos que estão salvos no *Redmine* e armazená-los no *Google Drive* para alimentação do portal Em Números. A escolha do *Google Drive* para base de dados se deu porque esta possibilita que os dados nela armazenados sejam visualizados de forma *online* pela plataforma utilizada para as visualizações do portal Em Números.

Figura 15 – Processo de publicação dos dados



Fonte: Elaborado pelo autor.

O processo de publicação, demonstrado na Figura 15, se inicia com uma aplicação que obtém os documentos necessários oriundos do *Redmine*. Em seguida, essa aplicação passa pelo procedimento de conexão com o serviço do *Google Cloud Plataform* (GCP) para efetuar a autenticação no *Google Drive*. Após ser autenticada, a

aplicação armazena os dados na nuvem. Posteriormente, a base de dados no *Google Drive* é utilizada para alimentar o portal Em Números, em um processo de atualização diária e sem intervenção humana.

O processo tem como início a execução de uma aplicação que, faz-se a publicação dos dados armazenados no *Redmine* para o *Google Drive*. Essa aplicação é configurada através de um agendador em um servidor Linux, no que lhe concerne, a execução dela diariamente. Em seguida, a aplicação obtém os arquivos necessários do *Redmine*, após isso, efetua a autenticação por meio do serviço do GCP e atualizada a base de dados no *Google Drive*. Com esses dados na nuvem, são consumidos pela plataforma que gera as visualizações resultando nos *dashboards* atualizados.

4.3 Visualização dos dados

O Portal da Transparência disponibiliza dados relacionados as entidades do Governo Federal, sendo as informações disponibilizadas por meio de tabelas e *dashboards* que simplificam o acesso às informações aos seus usuários.

Como o Portal da Transparência é feito para atender a todos os órgãos da Administração Pública Federal, as visualizações disponibilizadas acabam sendo genéricas demais em algumas situações e acabam por não atender a necessidades específicas das entidades. Por esta razão, a Enap (EV.G) dispõe de seu próprio mecanismo de transparência ativa, que permite o acesso ao mesmo conjunto de dados disponibilizados no Portal da Transparência, mas com visualizações específicas que permitam que os usuários analisem a utilização dos recursos públicos gerenciados pela Enap (EV.G), fomentando o controle social. Estes dados são publicados através de *dashboards* no Portal Em Números.

As visualizações dos dados supramencionadas foram possibilitadas por meio da criação de *dashboards* usando o *software Tableau*. O *Tableau* é uma plataforma de *business intelligence* para análise avançada de dados, que foi desenvolvida para atender às necessidades de cada usuário individual, sendo que sua implantação pode ser escalonada para toda a empresa[3]. Com este *software* é possível a criação de visualizações utilizando inúmeras bases de dados, que variam deste arquivos (CSV, xlsx, PDF etc) até banco de dados como (*mySQL*, *PostgreSQL*, *SQL Server*), dados na nuvem (*Google Drive*), entre outros. A base de dados utilizada pelo portal Em Números está armazenada na nuvem do *Google Drive*.

A criação das visualizações de *dashboards* no *Tableau* é feita a partir das bases de dados que contém as informações de interesse. Nesse trabalho, as visu-

[3] Disponível em $https://www.tableau.com/products/what-is-tableau$

alizações foram construídas com as informações da execução dos recursos entre as instituições parceira e a Enap (EV.G). A base de dados para essa aplicação está localizado no *Google Drive*, visto que o *Tableau* possui um serviço que permite obter os dados diretamente daquela plataforma.

Com a base de dados armazenada no *Google Drive*, é possível a sincronização automática das visualizações do *Tableau*. Em outras palavras, quando houver alterações na fonte de dados, automaticamente os novos dados serão refletidos nas visualizações. Neste contexto, as informações estão sendo atualizadas diariamente pelos *scripts* que buscam os dados do *Redmine* e os armazenam na nuvem, tornando o portal Em Números sempre atualizado com as informações mais recentes disponíveis no banco de dados do *Redmine*.

5 RESULTADOS E DISCUSSÕES

A partir da utilização da solução proposta, a Enap (EV.G) não precisará mais buscar um a um, em um processo manual e lento, os documentos das despesas disponíveis no Portal da Transparência. Então, a partir da aplicação da solução proposta neste trabalho, os documentos necessários foram obtidos de forma simplificada e ágil, de forma que a base de dados armazenada no banco do *Redmine* esteja sempre consistente e atualizada em relação à execução dos recursos públicos envolvidos.

Além disso, o desenvolvimento de *scripts* que sincronizam diariamente os dados registrados no *Redmine* para a área de armazenamento do *Google Drive*, permite que rapidamente os dados mais recentes armazenados, na Enap (EV.G), possam estar disponíveis aos usuários finais, vez que os dados no *Google Drive* são utilizados como fonte de alimentação do portal Em Números, que por sua vez é construído com a ferramenta *Tableau*.

Com a proposta foi possível obter os dados de interesse do Portal da Transparência. Estes dados foram obtidos, estruturados, disponibilizados e armazenados na nuvem. Além disso, encontra-se uma seção de visualização, no qual, são gerados painéis com *dashboards* baseado nos dados extraídos e, em seguida, são publicados no portal Em Números. Com isso, os servidores da Enap (EV.G) e os cidadãos podem acompanhar, de forma sistematizada, a execução do recurso descentralizado para outros órgãos parceiros.

Com o desenvolvimento da aplicação web, que utiliza uma API também desenvolvida especificamente para este fim, foi possível obter-se dois resultados, diferenciado pelo tipo do documento. A busca das informações de um pagamento individual e os dados de vários pagamentos de um determinado empenho. Essa consulta pode ser reconhecia pela própria API, conforme o parâmetro do número do documento repassado pelo usuário. A consulta pelo número de uma OB, teve como resultado as informações apenas desse documento e a partir do número da NE que, teve como resultado os dados de todas as OBs vinculadas a ela.

A consulta realizada por meio do número da OB teve como resultado as informações desse documento em específico e os dados de todos os favorecidos finais desse pagamento. Como é ilustrado na Figura 16, as informações foram adquiridas pelo *bot*, passaram pelo procedimento de estruturação e foram exibidas em formato de tabela, na *interface web*. Com essa tabela o usuário analisa as informações, verifica e as valida. Com isso, os dados são disponibilizados para *download* em um arquivo CSV.

Figura 16 – Aplicação com tabela de informações da OB obtidos pelo *bot*

Download

Nº do documento	Data	Descrição	Fase	Tipo de documento	Valor do documento	Observação do documento	Favorecido Final	CPF / CNPJ	Valor do favorecido
2018OB802002	04/10/2018	Ordem Bancária (OB)	Pagamento	OBB PARA MESMO BANCO/AGENCIA	R$ 146.600,00	PAGAMENTO DE FOLHA 9/2018-172, AUXILIO PARA PESQUISA, SEI 23106.114365/2018-75, GEPRO_TED_ENAP_ESCOLAVIRTUAL_2016.	[illegible]	***.056.471-**	950,00
2018OB802002	04/10/2018	Ordem Bancária (OB)	Pagamento	OBB PARA MESMO BANCO/AGENCIA	R$ 146.600,00	PAGAMENTO DE FOLHA 9/2018-172, AUXILIO PARA PESQUISA, SEI 23106.114365/2018-75, GEPRO_TED_ENAP_ESCOLAVIRTUAL_2016.	[illegible]	***.550.311-**	950,00
2018OB802002	04/10/2018	Ordem Bancária (OB)	Pagamento	OBB PARA MESMO BANCO/AGENCIA	R$ 146.600,00	PAGAMENTO DE FOLHA 9/2018-172, AUXILIO PARA PESQUISA, SEI 23106.114365/2018-75, GEPRO_TED_ENAP_ESCOLAVIRTUAL_2016.	[illegible]	***.453.161-**	1.000,00
2018OB802002	04/10/2018	Ordem Bancária (OB)	Pagamento	OBB PARA MESMO BANCO/AGENCIA	R$ 146.600,00	PAGAMENTO DE FOLHA 9/2018-172, AUXILIO PARA PESQUISA, SEI 23106.114365/2018-75, GEPRO_TED_ENAP_ESCOLAVIRTUAL_2016.	[illegible]	***.607.661-**	1.000,00
2018OB802002	04/10/2018	Ordem Bancária (OB)	Pagamento	OBB PARA MESMO BANCO/AGENCIA	R$ 146.600,00	PAGAMENTO DE FOLHA 9/2018-172, AUXILIO PARA PESQUISA, SEI 23106.114365/2018-75, GEPRO_TED_ENAP_ESCOLAVIRTUAL_2016.	[illegible]	***.351.991-**	1.500,00
2018OB802002	04/10/2018	Ordem Bancária (OB)	Pagamento	OBB PARA MESMO BANCO/AGENCIA	R$ 146.600,00	PAGAMENTO DE FOLHA 9/2018-172, AUXILIO PARA PESQUISA, SEI 23106.114365/2018-75, GEPRO_TED_ENAP_ESCOLAVIRTUAL_2016.	[illegible]	***.924.451-**	1.500,00

Fonte: Elaborado pelo autor.

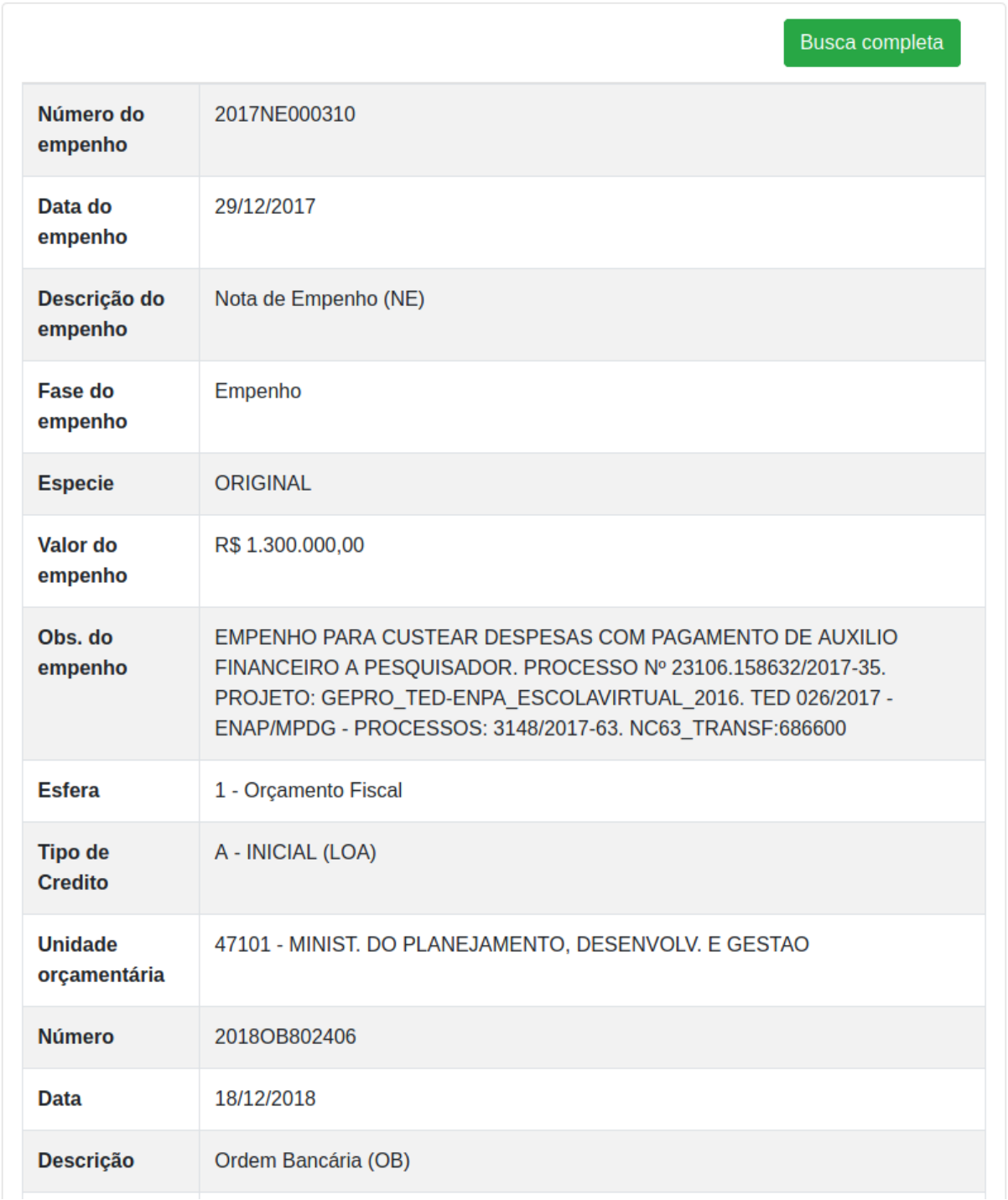
Ao se consultar o número de uma nota de empenho (NE), o resultado obtido é uma tabela contendo as informações do próprio documento e os dados da primeira ordem de pagamento bancário (OB) vinculada àquela NE.

A busca pela NE retorna uma quantidade maior de dados do que a busca pela OB. Ao se buscar por uma OB, somente são recuperados os dados de um único objeto, enquanto que ao se buscar uma NE, podem existir diversas ordens bancárias vinculadas a ela. Para os casos analisados neste trabalho, o maior número de OBs associados a uma mesma NE foi 35 (trinta e cinco). Devido ao tempo necessário para a recuperação dos dados, este processo de busca precisou ser dividido em duas partes, uma denominada **busca simples** e outra denominada **busca completa**.

A Figura 17 a seguir ilustra os resultados obtidos por meio da **busca simples** que resulta nas informações da NE, com os dados da primeira OB a ela vinculada. A busca simples retorna as informações de uma NE, agrupadas com a primeira OB associada a ela, resultando em uma tabela demonstrativa para o usuário. Dessa forma, ele poderá validar os dados antes de efetuar uma consulta completa que possui um maior tempo de resposta.

Na Figura 18 é possível se visualizar as informações que são recuperadas por meio de uma **busca completa** que retorna as informações da NE com os dados de todas as OBs a ela vinculadas. A busca completa mantém a tabela ilustrativa com os dados da busca simples, mas requisita os dados com as informações de todas as OBs vinculadas a NE. Desta forma, o usuário poderá realizar o *download* de todas essas informações em um arquivo CSV.

Figura 17 – Resultados de buscas simples usando NEs



Fonte: Elaborado pelo autor.

Figura 18 – Resultados de buscas completas usando NEs

Download

Número do empenho	2017NE000310
Data do empenho	29/12/2017
Descrição do empenho	Nota de Empenho (NE)
Fase do empenho	Empenho
Especie	ORIGINAL
Valor do empenho	R$ 1.300.000,00
Obs. do empenho	EMPENHO PARA CUSTEAR DESPESAS COM PAGAMENTO DE AUXILIO FINANCEIRO A PESQUISADOR. PROCESSO Nº 23106.158632/2017-35. PROJETO: GEPRO_TED-ENPA_ESCOLAVIRTUAL_2016. TED 026/2017 - ENAP/MPDG - PROCESSOS: 3148/2017-63. NC63_TRANSF:686600
Esfera	1 - Orçamento Fiscal
Tipo de Credito	A - INICIAL (LOA)
Unidade orçamentária	47101 - MINIST. DO PLANEJAMENTO, DESENVOLV. E GESTAO
Número	2018OB802406
Data	18/12/2018
Descrição	Ordem Bancária (OB)

Fonte: Elaborado pelo autor.

6 CONSIDERAÇÕES FINAIS E TRABALHOS FUTUROS

Este trabalho tem como objetivo principal promover uma solução para automatizar a atualização dos dados de execução de despesa pública, necessários ao monitoramento realizado pela Enap (EV.G), usando para isso informações do Portal da Transparência. Os dados necessários ao monitoramento das ações da Enap (EV.G) são extraídos automaticamente do referido portal, inseridos nos sistemas de registros existentes na Enap (EV.G), e publicados por meio de *dashboards* disponíveis no *site* da entidade. Isso viabiliza a transparência ativa incentivada pelos órgãos de controle, promove um controle social e permite uma visão global da aplicação dos recursos que circulam na Enap (EV.G).

Com aplicação da proposta, a Enap (EV.G) não precisa mais buscar manualmente os dados com informações sobre a aplicação dos recursos públicos executados pelas suas entidades parceiras, minimizando o tempo gasto nas execução manual deste processo naturalmente lento. A aplicação, objeto deste trabalho, foi construída para realizar a requisição desses dados, utilizando *scripts* que sincronizam diariamente os dados contidos no sistema de armazenamento da Enap (EV.G) para o *Google Drive*. Os dados disponibilizados no armazenamento em nuvem do *Google Drive*, por sua vez, são utilizados como fonte para a geração dos *dashboards* que são publicados no portal Em Números.

Com essa proposta de solução, foi possível obter-se os dados de interesse da Enap (EV.G) diretamente do Portal da Transparência, sem a necessidade de uma intermediação manual. Por meio de uma *interface web*, o usuário consegue solicitar os documentos de interesse em um dado momento. Por exemplo, o usuário pode solicitar todas as ordens de pagamentos associadas a uma nota de empenho inserindo na aplicação o número do documento do empenho e questão.

A *interface* utilizada pelo usuário final está conectada a uma API, que é a responsável pela ativação dos *bots*, que, por sua vez, são encarregados da obtenção dos dados no portal. Os *bots* demonstraram um bom desempenho, em termos de tempo gastos para a realização das consultas, conforme análise destacada no Capítulo 5. Com base nesses resultados, como era natural de se esperar, os *bots* obtiveram as informações de forma muito mais rápida se comparado a um processo de busca manual, realizado pelo servidor da Enap (EV.G).

Indubitavelmente, a aplicação desenvolvida apoiará de forma contundente a consolidação dos dados de execução das despesas da Enap (EV.G) no âmbito das parceiras firmadas com outros órgãos da Administração Pública Federal. Como men-

cionado na Seção 2.4.2, a parceria entre uma instituição e a Enap (EV.G) é formalizada por meio de um TED, mecanismo pelo qual se autoriza o repasse de créditos orçamentários de uma instituição para outra. A formalização do repasse se dá por meio da emissão de um documento de nota de movimentação de crédito (NC).Uma NC pode conter várias notas de empenho (NEs) vinculadas e as NEs, por sua vez, cada uma, pode obter várias ordens de pagamentos (OBs) associadas.

Como trabalho futuro vislumbra-se três linhas de ações: (i) a busca das OBs por meio de NCs e não apenas por meio de NEs; (ii) uma melhoria do desempenho no tempo de resposta para a obtenção dos dados utilizando uma API de serviços disponibilizada pelo Portal da Transparência, que só foi identificada ao final do desenvolvimento deste trabalho; e (iii) sugere-se ainda, a implantação deste serviço no âmbito da Enap para coletar *feedback* a partir da utilização do sistema em produção, pelos próprios beneficiários da solução.

REFERÊNCIAS

Algiryage, N.; Dias, G.; Jayasena, S. Distinguishing real web crawlers from fakes: Googlebot example. In: *2018 Moratuwa Engineering Research Conference (MERCon)*. [S.l.: s.n.], 2018. p. 13–18. Citado na página 19.

ARRUDA, D. G.; ARAUJO, I. D. P. S. *Contabilidade pública*. São Paulo, SP: Editora Saraiva, 2017. Citado na página 29.

BRASIL. Senado Federal. *O que é transparência ativa?* 2017. Disponível em: <https://www12.senado.leg.br/perguntas-frequentes/perguntas-frequentes/canais-de-atendimento/transparencia-1/o-que-e-transparencia-ativa>. Acesso em: 06 de julho de 2019. Citado na página 14.

CHALLITA, S. et al. A precise model for google cloud platform. In: IEEE. *IEEE International Conference on Cloud Engineering (IC2E)*. Orlando, FL, USA, 2018. p. 177–183. Citado na página 28.

CHERKESOV, V. et al. Parsing of data on real estate objects from network resource. In: *IV International research conference "Information technologies in Science, Management, Social sphere and Medicine"(ITSMSSM 2017)*. [S.l.]: Atlantis Press, 2017. Citado na página 23.

D'HAEN, J. et al. Integrating expert knowledge and multilingual Web crawling data in a lead qualification system. *Decision Support Systems*, Elsevier, v. 82, p. 69–78, 2016. Citado na página 19.

ESCAVADOR. *Quem Somos*. 2019. Disponível em: <https://www.escavador-.com/quem-somos>. Acesso em: 27 de agosto de 2019. Citado na página 39.

FAROOQ, B.; HUSAIN, M. S.; SUAIB, M. Crawling of japanese real-estate websites using scrapy. *International Journal of Advanced Research in Computer Science*, v. 9, n. 2, p. 64–67, 2018. Citado na página 25.

GOOGLE. *Como a Pesquisa Google funciona*. 2019. Disponível em: <https://support-.google.com/webmasters/answer/70897>. Acesso em: 09 de agosto de 2019. Citado na página 19.

GOOGLE. *Introdução ao robots.txt*. 2019. Disponível em: <https://support.google-.com/webmasters/answer/6062608>. Acesso em: 15 de agosto de 2019. Citado na página 22.

GRINBERG, M. *Flask Web development: developing Web applications with python*. Sebastopol, CA, USA: O'Reilly Media, 2018. Citado na página 28.

JARAMILLO, D.; NGUYEN, D. V.; SMART, R. Leveraging microservices architecture by using docker technology. In: *SoutheastCon 2016*. Norfolk, VA, USA: IEEE, 2016. p. 1–5. Citado na página 27.

KAUR, T. Trivago business model: A platform to think before. *Our Heritage*, v. 68, n. 1, p. 9504–9509, 2020. Citado na página 41.

KHALIL, S.; FAKIR, M. Rcrawler: An R package for parallel Web crawling and scraping. *SoftwareX*, Elsevier, v. 6, p. 98–106, 2017. Citado na página 19.

KOUZIS-LOUKAS, D. *Learning scrapy*. Birmingham, United Kingdom: Packt Publishing, 2016. Citado na página 23.

MAHTO, D. K.; SINGH, L. A dive into Web scraper world. In: *3rd International Conference on Computing for Sustainable Global Development (INDIACom)*. New Delhi, India: IEEE, 2016. p. 689–693. Citado na página 22.

MITCHELL, R. *Web scraping with Python: Collecting more data from the modern Web*. [S.l.]: "O'Reilly Media, Inc.", 2018. Citado na página 23.

MOTOMURA, M. *Quanto tempo o Google demoraria para catalogar todas as páginas da internet?* 2016. Disponível em: <https://super.abril.com.br/mundo-estranho-/quanto-tempo-o-google-demoraria-para-catalogar-todas-as-paginas-da-internet/>. Acesso em: 18 de agosto de 2019. Citado na página 19.

NASCIMENTO, E. R. *Gestão pública*. São Paulo, SP: Editora Saraiva, 2017. Citado na página 32.

NIÑO-MORA, J. A dynamic page-refresh index policy for Web crawlers. In: SERICOLA, B.; TELEK, M.; HORVÁTH, G. (Ed.). *International Conference on Analytical and Stochastic Modeling Techniques and Applications*. Cham: Springer, 2014. p. 46–60. Citado na página 20.

NISAFANI, A. S.; HENDRAWAN, R. A.; WIBISONO, A. Eliciting data from website using scrapy: An example. *SEMNASTEKNOMEDIA ONLINE*, v. 5, n. 1, p. 2–1, 2017. Citado na página 24.

OMARI, A.; SHOHAM, S.; YAHAV, E. Cross-supervised synthesis of web-crawlers. In: *Proceedings of the 38th International Conference on Software Engineering*. New York, NY, USA: ACM, 2016. p. 368–379. Citado na página 18.

RAMÍREZ-CORONEL, R. L. et al. Semantic architecture for the extraction, storage, processing and visualization of internet sources through the use of scrapy and crawler techniques. In: BOTTO-TOBAR, M. et al. (Ed.). *Information and Communication Technologies of Ecuador*. Cham: Springer, 2019. p. 301–313. Citado na página 24.

SANTOS, M. G. Portal da transparência da cidade de bananeiras: uma análise segundo parâmetros da lei de acesso à informação e requisitos de usabilidade. 2018. Citado na página 14.

UDAPURE, T. V.; KALE, R. D.; DHARMIK, R. C. Study of Web crawler and its different types. *IOSR Journal of Computer Engineering*, v. 16, n. 1, p. 01–05, 2014. Citado na página 20.

VIEIRA, F. S.; PIOLA, S. F. *Restos a pagar de despesas com ações e serviços públicos de saúde da União: impactos para o financiamento federal do Sistema Único de Saúde e para a elaboração das contas de saúde*. Brasília, 2016. Citado 2 vezes nas páginas 31 e 32.

ZHAO, B. Web scraping. *Encyclopedia of Big Data*, Springer, Basel, p. 1–3, 2017. Citado na página 21.

ZHOU, J. et al. Integration and analysis of agricultural market information based on Web mining. In: *6th IFAC Conference on Bio-Robotics*. Beijing, China: Elsevier, 2018. v. 51, n. 17, p. 778–783. Citado na página 26.

APÊNDICES

APÊNDICE A – Código do *bot* que busca as ordens bancárias vinculada ao Empenho

Código 1 – Busca das ordens bancárias vinculada ao empenho.

```
import os, scrapy
from scrapy_splash import SplashRequest
from portal.items import PortalItem

class PagamentosEmpenhoSpider(scrapy.Spider):
    name = 'pagamentoEmpenho'
    init = False
    URL = 'http://www.portaltransparencia.gov.br/despesas/pagamento/{}?ordenarPor=fase&direcao=asc'
    SRC_FAVORECIDOS = 'wget http://www.portaltransparencia.gov.br/despesas/documento/pagamento/favorecido/baixar\?direcaoOrdenacao\=asc\&codigo\={0} -O ./files/favorecidosFinais-{0}.csv'

    def __init__(self, code='', *args, **kwargs):
        self.codigo = code
        # Obtem informações do empenho
        self.getDataEmpenho(self.codigo)
        super(PagamentosEmpenhoSpider, self).__init__(*args, **kwargs)

    def start_requests(self):
        codes = self.getCodesPay(self.codigo)  #Pega uma lista de códigos de OB relacionada ao empenho
        for code in codes:
            yield SplashRequest(
                url = self.URL.format(code),
                callback = self.parse, args={'wait': 5}
            )

    def parse(self, response):
        code = response.url.split('?')[0].split('/')[-1:][0]
        # Lista de favorecidos da OB
        self.codigoOB = code
        os.system(self.SRC_FAVORECIDOS.format(code))
        if not self.init:
            self.init = True
            header = response.css('section.dados-tabelados strong::text').extract()
            yield PortalItem(row = self.H_EMPENHO + header)

```

```
35          row = response.css('section.dados-tabelados span::text').extract()
36          yield PortalItem(row = self.R_EMPENHO + row)
37
38      #Pega o número das OB do arquivo gerado pelo empenho
39      def getCodesPay(self, code):
40          codes = []
41          with open('./files/documentosRelacionados-'+ code +'.csv', 'r') as
                file:
42              for line in file:
43                  if 'OB' in line:
44                      codes.append(line.split(';')[2])
45          return codes
46
47      # Pega dados do empenho
48      def getDataEmpenho(self, code):
49          with open('./files/portal-empenho-{}.csv'.format(code), 'r') as
                file:
50              # Adicionando empenho no header para diferenciar
51              self.H_EMPENHO = file.readline().replace('|', ' empenho|').
                    replace('\n', '').split('|')
52              # Pegando dados do empenho
53              self.R_EMPENHO = file.readline().replace('\n', '').split('|')
```

APÊNDICE B – Código para sincronização dos dados na nuvem

Código 2 – *Script* para atualizar os dados na nuvem

```
import sys
import math
import gspread
import numpy as np
import pandas as pd
from oauth2client.service_account import ServiceAccountCredentials

scope = ['https://spreadsheets.google.com/feeds', 'https://www.googleapis.com/auth/drive']

NAME_FILE = sys.argv[1]          # Nome do arquivo com os dados
NAME_WORKSHEET = sys.argv[2]     # Nome da planilha a ser criada

# Faz a leitura dos dados do arquivo
def getData(file):
    data = pd.read_csv(file, sep=';', encoding="UTF-8") # Selecionando os dados a serem inseridos
    return data # Return (Dataframe)

# Autenticação através do OAuth 2.0
def getCredenciais(scope, file_secret_json):
    creds = ServiceAccountCredentials.from_json_keyfile_name(file_secret_json, scope)
    client = gspread.authorize(creds)
    return client

# Busca o arquivo com as planilhas ou cria se não existir
def initFile(client):
    try:
        file = client.open(NAME_FILE)
    except:
        file = client.create(NAME_FILE)
        #Compartilha arquivo para o drive
        file.share('igormgaldino@gmail.com', perm_type='user', role='writer')
    return file
```

```
37 # Busca a planilha ou cria se não existir
38 def initWorksheet(file, name, rows, cols):
39     try:
40         planilha = file.worksheet(name)
41     except:
42         planilha = file.add_worksheet(title=name, rows=rows, cols=cols)
43     return planilha
44 
45 def main():
46     # Efetua a autenticação
47     client = getCredenciais(scope, 'client.json')
48     # Busca o arquivo com as planilhas
49     file = initFile(client)
50     # Dataframe do arquivo local
51     data = getData(NAME_WORKSHEET)
52     #Quantidade de colunas e linhas
53     qntCol, qntRow = len(data.columns), len(data)+1
54     # Busca planilha no drive pelo nome
55     planilha = initWorksheet(file, NAME_WORKSHEET, qntCol, qntRow)
56     # Seleciona celulas que receberão os dados em um List
57     listCell = planilha.range(1, 1, qntRow, qntCol)
58     # Lista com as linhas da planilha
59     rows = np.array(data).flatten().tolist()
60     # Lista com cabeçalho da planilha
61     head = np.array(data.columns).tolist()
62     # União do cabeçalho com as linhas
63     dados = head + rows
64     # Atribuicao dos valores dos dados nas celulas
65     for cell, item in zip(listCell, dados):
66         # Transforma os valores (numéricos) nulos em branco
67         if(type(item) == float and math.isnan(item)):
68             cell.value = ''
69         else:
70             cell.value = item
71     # Limpa todas as celulas da pagina da planilha
72     planilha.clear()
73     # Atualização das celulas com os dados
74     planilha.update_cells(listCell)
75 
76 if __name__ == "__main__":
77     main()
```

ANEXO

Anexo A – Termo de autorização

Enap Escola Nacional de Administração Pública

Despacho nº 13018/2019

De: CGEAD/DEC

Processo: 04600.006017/2019-81

Assunto: **Termo de autorização**

1. Autorizo o aluno **IGOR MARTINS GALDINO**, sob a orientação da professora **ÉRICA DE LIMA GALLINDO**, do Instituto Federal do Ceará, a utilizar informações acerca do processo de prestação de contas da Coordenação-Geral de Educação a Distância da Diretoria de Educação Continuada (CGEAD/DEC) da Fundação Escola Nacional de Administração Pública (Enap), no contexto do seu projeto de pesquisa conforme indicado a seguir:

- **Título** (provisório): Extração automática de dados do Portal da Transparência
- **Problema**: Manutenção atualizada das informações acerca da execução financeira das instituições parceiras da EV.G.
- **Objeto de Pesquisa**: Prover uma solução para automatizar a atualização dos dados a partir das informações disponibilizadas no Portal da Transparência, que atualiza diariamente os dados associados à execução da despesa pública, oriundos do SIAFI.
- **Objetivos Específicos:**
    - Criar um extrator automáticos dos dados do Portal da Transparência;
    - Criar um mecanismo para inserção automática dos dados extraídos do Portal da Transparência no sistema de registros dos documentos da despesa utilizado pela EV.G.

2. É também de responsabilidade dos pesquisadores garantir que as informações coletadas sejam utilizadas exclusivamente para fins acadêmicos e que não sejam utilizadas em prejuízo desta instituição.
3. Por fim, para fins de socialização, os pesquisadores assumem o compromisso de compartilhar com a Enap o resultado da pesquisa e o trabalho final.

Documento assinado eletronicamente por **Natália Teles da Mota Teixeira**, **Coordenador(a)-Geral de Educação a Distância**, em 24/10/2019, às 19:35, conforme horário oficial de Brasília e Resolução nº 9, de 04 de agosto de 2015.

A autenticidade deste documento pode ser conferida no site http://sei.enap.gov.br/autenticidade, informando o código verificador **0335168** e o código CRC **D5CEB13F**.

Referência: Processo nº 04600.006017/2019-81 SEI nº 0335168