UNIVERSIDADE FEDERAL DE SÃO CARLOS
CENTRO DE CIÊNCIAS EM GESTÃO E TECNOLOGIA
CAMPUS DE SOROCABA
PROGRAMA DE PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

JULIANA SABINO FERREIRA

UMA ABORDAGEM PARA CAPTURA AUTOMATIZADA DE DADOS ABERTOS GOVERNAMENTAIS

Sorocaba
2017

UNIVERSIDADE FEDERAL DE SÃO CARLOS
CENTRO DE CIÊNCIAS EM GESTÃO E TECNOLOGIA
CAMPUS DE SOROCABA
PROGRAMA DE PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

JULIANA SABINO FERREIRA

UMA ABORDAGEM PARA CAPTURA AUTOMATIZADA DE DADOS ABERTOS GOVERNAMENTAIS

Dissertação apresentada ao Programa de Pós-Graduação em Ciência da Computação (PPGCC-So), para obtenção do título de mestre em Ciência da Computação. Área de concentração: Engenharia de Software e Gestão do Conhecimento.

Orientação: Prof. Dr. Alexandre Alvaro

Sorocaba
2017

Sabino Ferreira, Juliana

UMA ABORDAGEM PARA CAPTURA AUTOMATIZADA DE DADOS ABERTOS GOVERNAMENTAIS / Juliana Sabino Ferreira. -- 2017.
122 f. : 30 cm.

Dissertação (mestrado)-Universidade Federal de São Carlos, campus Sorocaba, Sorocaba
Orientador: Alexandre Alvaro
Banca examinadora: Profa. Dra. Ellen Francine Barbosa, Profa. Dra. Luciana Aparecida Martinez Zaina
Bibliografia

1. Dados Abertos Governamentais. 2. Web Crawlers. 3. Cidades Inteligentes. I. Orientador. II. Universidade Federal de São Carlos. III. Título.

Ficha catalográfica elaborada pelo Programa de Geração Automática da Secretaria Geral de Informática (SIn).
DADOS FORNECIDOS PELO(A) AUTOR(A)

UNIVERSIDADE FEDERAL DE SÃO CARLOS

Centro de Ciências em Gestão e Tecnologia

Programa de Pós-Graduação em Ciência da Computação

Folha de Aprovação

Assinaturas dos membros da comissão examinadora que avaliou e aprovou a defesa de dissertação de mestrado da candidata Juliana Sabino Ferreira, realizada em 07/11/2017:

Prof. Dr. Alexandre Alvaro
UFSCar

Profa. Dra. Ellen Francine Barbosa
USP

Profa. Dra. Luciana Aparecida Martinez Zaina
UFSCar

Certifico que a sessão de defesa foi realizada com a participação à distância dos membros Profa. Dra. Ellen Francine Barbosa e Profa. Dra. Luciana Aparecida Martinez Zaina e, depois das arguições e deliberações realizadas, os participantes à distância estão de acordo com o conteúdo do parecer da comissão examinadora redigido no relatório de defesa da aluna Juliana Sabino Ferreira.

Prof. Dr. Alexandre Alvaro
Presidente da Comissão Examinadora
UFSCar

DEDICATÓRIA

[Para meus pais, que apesar das origens humildes me deixaram como herança o maior bem que alguém pode ter, a educação.]

AGRADECIMENTO

Agradeço

Aos meus pais, não só por me ter dado a vida, mas também por me educar e transformar na pessoa que sou hoje. Sou grata por toda educação que recebi, que permitiu a criação desse trabalho.

Ao meu orientador, Alexandre Alvaro, por acreditar na minha capacidade e no meu trabalho e por dar todo o apoio que precisei antes mesmo de me tornar sua orientada.

Aos professores da UFSCAR, que além de conhecimento, me proporcionaram toda a vivência na universidade, em especial das professoras Profs. Dr.as Luciana Zaina e Sahudy Montenegro, minha banca de qualificação e que permitiram a continuidade desse trabalho.

Ao Ms. Daniel Ianegitz Vieira, com trabalho com temática semelhante e que permitiu que eu desse continuidade na pesquisa.

A UFSCAR, com todos seus professores, funcionários e alunos, que me ofertaram uma estrutura adequada para que eu pudesse estudar e realizar este trabalho, além de me proporcionar bons momentos, e no compartilhamento de experiências, que foram muito ricas para a minha formação.

Ao IFSP, Câmpus Boituva, local onde trabalho, que me proporcionou tempo, compreensão e apoio para que eu pudesse estudar.

Aos alunos Ingrid Santos, Willian Adriano Alves, Caio Henrique Giacomelli, Lucas Yoshimura, Pedro Pires e Matheus Thomaz por me ajudarem na validação.

A Marcelo Polido e Alex Machado Sampaio Ferreira por também me ajudaram na validação do meu trabalho.

A João Eduardo Soares e Silva por não só me ajudar na validação como também com outras contribuições para esse trabalho.

E a todos que diretamente ou indiretamente me ajudaram a tornar esse trabalho uma realidade.

RESUMO

Atualmente os dados abertos governamentais exercem um papel fundamental na transparência pública na gestão dos governos, além de ser uma obrigação legal. Porém grande parte desses dados são publicados em formatos diversos, isolados e independentes, dificultado seu reaproveitamento por sistemas de terceiros que poderiam reusar informações disponibilizadas em tais portais. Este trabalho propõe a criação de uma abordagem para captura de dados abertos governamentais de forma automatizada, permitindo sua reutilização em outras aplicações.

Para isso foi construído um Web Crawler para captura e armazenamento de Dados Abertos Governamentais (DAG) e a API DAG Prefeituras para disponibilizar esses dados no formato JSON para que outros desenvolvedores possam utilizar esses dados em suas aplicações.

Também foi realizada uma avaliação do uso da API para desenvolvedores com diferentes níveis de experiência.

Palavras-chave: Dados Abertos Governamentais, Cidades Inteligentes, Web Crawler.

A**BSTRACT**

Currently open government data run an important job on regards to public transparency, besides being obligated by law. But most of this data are stored in non-standard ways, isolated and independent, making it very hard for its use by third party systems providers. This work proposes the creation of an approach for capturing this open government data in an automated way, allowing its use in various applications.
For that a Web Crawler was built for the capture and storing of this open government data, as well as an API for making this data available in JSON format, that way developers can easily use this data on their application.
We also performed an evaluation of the API for developers with different levels of experience.

Keywords: Open Government Data, Smart Cities, Web Crawler.

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE ABREVIATURAS, SIGLAS E SÍMBOLOS

CSV	Comma-separated values
DAG	Dados Abertos Governamentais
ICTs	Information and Comunication Technology
JSON	JavaScript Object Notation
ODS	Open Document Spreadsheet
RDF	Resource Description Framework
REST	Representation State Transfer
XML	eXtensible Markup Language

SUMÁRIO

1. INTRODUÇÃO

Nos últimos anos nota-se um grande crescimento das cidades, na maioria de forma desordenada. Tudo isso contribui para o aumento da demanda por serviços de utilidade à população. Em paralelo, o uso de Tecnologias de Informação e Comunicação (TICs) vem auxiliando no monitoramento das cidades como forma de acelerar o atendimento dessas demandas. A junção de demandas das cidades com o uso de tecnologias ganhou o conceito de "Smart City". (SUAKANTO, SUPANGKAT, *et al.*, 2013)

"Smart City", ou Cidades Inteligentes, é a utilização de redes, equipamentos, softwares, entre outras tecnologias e metodologias para auxiliar na coleta de dados e na tomada de decisões dos problemas das cidades. Uma cidade inteligente também pode ser definida como uma "cidade que prevê e acomoda as necessidades dos cidadãos" (EUROPEAN, 2013).

Para realizar o planejamento de uma cidade é necessário preocupar-se com muitas áreas, que varia desde mobilidade urbana, infraestrutura, política, planejamento, gestão, governança de dados abertos e compartilhamento de conhecimento (EUROPEAN, 2013).

Uma das áreas no contexto de cidades inteligentes que vem ganhado destaque é a Governança dos Dados Abertos Governamentais. Essa área está relacionada aos Dados Abertos Governamentais, e dentre outras coisas inclui formas de criação, publicação e gestão dos mesmos. Nos últimos anos os volumes de dados relativos aos governos aumentaram de forma muito significativa e com isso logo surgiram diversos desafios para gerenciá-los, além de disponibilizá-los em formatos abertos e reaproveitáveis (CORRÊA, CORRÊA e SILVA, 2014).

Esses dados, conhecidos como DAG (Dados Abertos Governamentais), ou OGD (Open Government Data), são publicados pelos governos com o intuito de aumentar a transparência das informações públicas. Também podem ser vistos como um recurso estratégico dos governos e também como uma forma de suporte à estratégia econômica (CORRÊA, CORRÊA e SILVA, 2014).

No Brasil, a Lei de Acesso à Informação regulamenta o "direito constitucional de acesso às informações públicas" (BRASIL, 2011). Hoje, diversos portais no Brasil disponibilizam dados abertos, porém os mesmos não são publicados de forma correta, pois a lei determina alguns tipos de formatos de arquivos que os dados precisam ser publicados. Além disso, não existe uma padronização na forma de dispor os dados, que por sua vez não seguem os princípios dos DAG.

Uma das alternativas para a utilização desses dados, que são publicados de forma incorreta é a obtenção e disponibilização dos mesmos em formatos corretos. Para isso uma solução é a utilização de Web Crawlers (DHENAKARAN e SAMBANTHAN, 2011).

Web Crawlers são softwares que fazem buscas automáticas na web, como uma espécie de "robô que varre a web". Ele captura uma ou mais URLs de uma página e conforme navega entre elas, vai capturando outras URLs, formando uma lista com elas. Durante a leitura também é possível capturar outros dados na página que não sejam URLs. (DHENAKARAN e SAMBANTHAN, 2011)

Neste contexto, este trabalho propõem o desenvolvimento de um Web Crawler que faz a captura de DAG de prefeituras e disponibiliza os dados através de uma API, que dispõe os mesmos em formato JSON, que é aceito segundos a Cartilha que norteia a publicação e DAG no Brasil (CARTILHA, 2016). Além disso, foi realizado um estudo de caso para avaliar o uso da API. Este estudo comparou o uso da API com desenvolvedores com pouca e muita experiência (alunos da graduação e profissionais do mercado).

1.1. MOTIVAÇÃO

Uma das principais motivações para esse trabalho é o contexto social que os Dados Abertos Governamentais podem representar para o país. Esses dados são considerados valiosos recursos estratégicos e uma forma de verificar se os gastos governamentais estão adequados.

Nota-se cada vez mais escândalos de corrupção na mídia ao mesmo tempo que a população enfrenta diversos problemas ao utilizar serviços do governo, como saúde e educação, que são cada vez mais deficientes e em quantidade insuficiente para atender a toda demanda.

Uma forma de combate a essa corrupção que tira recursos que poderiam ser utilizados para benefício da população é fiscalizar como o governo está gatando os recursos públicos. Por lei, o governo como um todo precisa publicar dados sobre si, incluindo os dos gastos, logo essas informações precisam estar disponíveis para a população em geral.

Os Dados Abertos Governamentais trazem os gastos e investimentos do governo. Esses dados podem ser analisados por qualquer pessoa e tornarem uma forma de fiscalização, onde a população em geral pode cobrar do governo que o dinheiro seja gasto da melhor forma possível, que seja benéfica para todos.

Porém, apesar de existir esses dados, o acesso e a utilização deles é algo difícil, principalmente devido a forma que esses dados são publicados. Uma das ideias de se publicar dados abertos é que eles fiquem disponíveis e possam ser facilmente utilizados em outras aplicações, de forma automatizada. No Brasil existem diretrizes para que os DAG sejam

publicados seguindo alguns padrões, porém, muitos desses dados não são publicados utilizando esses padrões (CARTILHA, 2016).

Como consequência, hoje tem-se um grande volume de dados que não são utilizados, logo esse trabalho teve como sua principal motivação ajudar a mudar esse cenário. Pois nele foi feita a captura de dados que inicialmente não poderiam ser aproveitados e posteriormente foi feita a publicação dos mesmos de forma que outras aplicações possam utilizá-los, de maneira automatizada.

A escolha por capturar dados de prefeituras dá devido alguns motivos: primeiro, é o setor que mais publica dados de forma precária ao analisar as diretrizes que deveriam ser seguidas para a publicação de DAG. Outro motivo é devido os dados se restringirem a cidades, o uso desses dados e a própria fiscalização ficaria mais facilitada, pois a população teria que se preocupar apenas com o que ocorre na cidade, que de certa forma seria mais fácil, pois a cobrança seria de um governo local, e devido a abrangência geográfica, o volume de dados seria menor. Uma das formas de se combater a corrupção seria preocupar-se primeiro com as cidades, fazer que a população em si começasse a cobrar uma melhor gestão do dinheiro público do local onde vivem. Por isso esse trabalho optou em utilizar DAG de prefeituras para fazer a captura.

O uso de DAG é uma grande ferramenta de gestão e combate a corrupção, capturar e disponibilizar esses dados é uma forma de facilitar a análise de DAG para que consequentemente a fiscalização de uso do dinheiro público seja mais eficiente.

1.2. OBJETIVOS

O Objetivo principal desse trabalho é a disponibilização de DAG em formatos adequados segundo os princípios dos DAG (OPEN, 2016) e as recomendações da Cartilha que norteia a publicação de DAG no Brasil (CARTILHA, 2016). Para esse fim foram selecionados DAG que foram publicados em formatos inadequados segundo esses princípios. Esses dados foram disponibilizados novamente, só que da forma correta.

Devido a isso, foi necessário dividir o trabalho em três etapas principais: pesquisa, desenvolvimento de software e avaliação. De forma geral, os objetivos foram os seguintes:

- Buscar informações detalhadas sobre DAG.
- Pesquisar sobre como os DAG são publicados pelo governo. Para isso foram selecionados dados de municípios brasileiros.
- Pesquisa sobre formas de captura de dados utilizando Web Crawlers.

- Construção de Web Crawler para reaproveitar DAG já existentes.
- Criação de API para disponibilizar os dados capturados pelo Web Crawler.
- Prover avaliação da API para verificar se ela atende desenvolvedores com diferentes níveis de experiência.

1.3. ESCOPO NEGATIVO

Para o desenvolvimento deste trabalho o escopo negativo foi:

- *Disponibilização de dados de mais cidades*: Foram analisados diversos portais da transparência de vários sites de prefeituras do estado de São Paulo e os mesmos foram divididos conforme sua complexidade para captura dos dados. Para o Web Crawler inicial foram escolhidas duas prefeituras em que a complexidade foi classificada como baixa. Como trabalhos futuros estão previstas ampliação do Web Crawler para que ele possa ler dados de outras prefeituras.
- *Desenvolvimento de aplicações utilizando os dados:* O foco deste trabalho foi o desenvolvimento de um Web Crawler para que seja feita a captura dos dados de forma automatizada. A disponibilização dos dados coletados ocorreu através de uma API, que fornece os dados de maneira facilitada para outros desenvolvedores criarem aplicações no contexto de DAG. Foram feitas avaliações do uso da API na construção de aplicações, só que estas desenvolvidas por terceiros.

1.4. METODOLOGIA

As etapas gerais relacionadas à metodologia foram as seguintes: estudo bibliográfico, pesquisa de portais, desenvolvimento do Web Crawler e do Web Service e avaliação (Figura 1).

Inicialmente em maior escala e durante todo o trabalho conforme a necessidade foi feito um estudo bibliográfico sobre os seguintes temas: Cidades Inteligentes, Dados Abertos Governamentais e Web Crawlers. Foi realizada a leitura e resumos de diversos artigos com essa temática.

Após o estudo bibliográfico, foi realizada uma pesquisa nos portais da transparência de 44 prefeituras do estado de São Paulo. Foi verificado informações como, o formato de disponibilização dos dados, estrutura dos sites, tipos de relatórios disponíveis, etc. Essa pesquisa foi a norteadora para a escolha dos portais que foram usados.

A partir de então foi iniciado o desenvolvimento do Web Crawler, mas para isso era preciso primeiro escolher que portais inicialmente o software faria a leitura. Com base na

pesquisa dos portais de transparência feitos anteriormente, os portais foram classificados em três níveis de complexidade: baixa, média e alta. Essa classificação levou em conta a sua dificuldade de coletar dados utilizado um Web Crawler. No final, portais de seis cidades foram classificados como baixa complexidade. Inicialmente foi escolhido o site da prefeitura de São Roque para a construção do Web Crawler e posteriormente o site da prefeitura de Capivari também foi utilizado.

Em seguida foram realizadas pesquisas sobre definições e funcionamentos de Web Crawlers. Foi estudado artigos sobre o tema e em paralelo diversas bibliotecas sobre Web Crawlers foram estudadas e testadas. Durante esses testes foram escolhidas as APIs JSOUP e HTMLUnit, ambas funcionam em linguagem Java.

Foi então construído um Web Crawler usando a linguagem Java e as APIs JSOUP e HTMLUnit para a captura dos dados relacionados a Despesa da prefeitura de São Roque. Posteriormente o software também fez a captura de dados de outra prefeitura, Capivari, que possui um site com estrutura parecida com de São Roque.

Após a construção do Web Crawler foram criadas funções para visualização dos dados capturados. Inicialmente essas funções funcionaram em ambiente local, mas posteriormente foram inseridas em uma API, que funciona através de um Web Service.

Posteriormente foi realizada uma avaliação da proposta usando os princípios de experimentação de software (TRAVASSOS, GUROV e AMARAL, 2002) (BASILI, SELBY e HUTCHENS, 1986) para verificar a utilidade e facilidade de uso da API para desenvolvedores com pouca e muita experiência. Para isso foi aplicado um estudo durante um dia como alunos de graduação do curso de Ciência da Computação da UFSCAR Sorocaba e para profissionais do mercado.

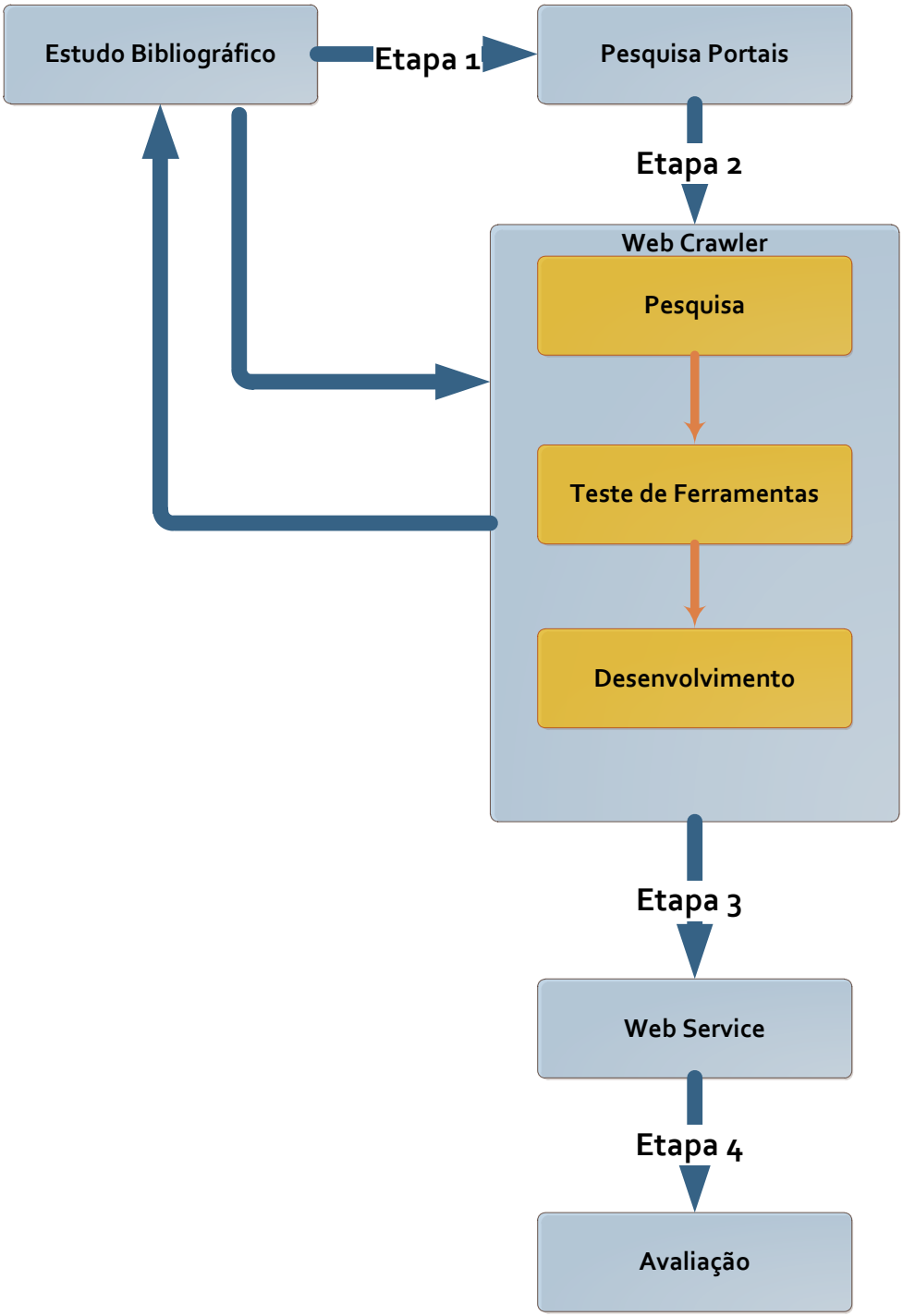


Figura 1: Metodologia da Pesquisa

1.5. RESULTADOS OBTIDOS

Neste trabalho foram obtidos diversos resultados, listados a seguir:

- Web Crawler: Aplicação feita na linguagem Java capaz de fazer a captura automatizada de dados relativos à despesa das prefeituras de São Roque e de Capivari. Após essa captura, os dados são analisados e armazenados em um Banco

de Dados para serem acessados por uma API.

- API DAG Prefeituras: API que através de um Web Service faz acesso à base de dados com os dados capturados pelo Web Crawler e retorna os mesmos no formato JSON. Esse formato é um dos formatos aceitos para a publicação de DAG, segundo os princípios dos dados abertos (OPEN, 2016). Esse acesso consiste de um conjunto de funções variadas que retornam os dados desejados através de filtros, passados como parâmetros pelo usuário.
- Avaliação da proposta: Avaliação utilizando princípios de experimentação de software (TAN e MITRA, 2010), (BASILI, SELBY e HUTCHENS, 1986), para verificação da utilidade e da facilidade de uso da API para programadores com muita e pouca experiência.

1.6. ORGANIZAÇÃO DO TRABALHO

Este trabalho está dividido nos seguintes capítulos, descritos a seguir:

- *Introdução:* Mostra um resumo geral sobre os itens tratados no trabalho, como Dados Abertos Governamentais e Web Crawlers. Além de mostrar as motivações, objetivos e metodologia utilizada.
- *Revisão Bibliográfica e Background:* Faz um estudo geral sobre Dados Abertos Governamentais e Web Crawlers. Também faz um panorama em ordem cronológica dos trabalhos relacionados ao tema, envolvendo Dados Abertos Governamentais, Web Crawlers e Cidades Inteligentes.
- *Estudo Dos Dados Abertos Governamentais De Municípios Brasileiros:* mostra a pesquisa inicial realizada para a escolha dos portais da transparência utilizados neste trabalho.
- *Abordagem para Captura Automatizada de Dados Abertos Governamentais:* Descreve detalhes do desenvolvimento do Web Crawler e da API DAG Prefeituras. No desenvolvimento explica de forma detalhada sobre Web Crawlers, mostrando sua definição e funcionamento. Detalha a arquitetura do software construído, que consiste de um Web Crawler para captura, análise e armazenamento de dados, e da API DAG Prefeituras, que disponibiliza os dados capturados em JSON, adequados para serem utilizados em outras aplicações.
- *Avaliação da Proposta:* mostra como foi a validação da API usando princípios de experimentação de software, para desenvolvedores com pouca e com muita experiência.

- *Conclusão:* Retoma os principais itens do trabalho e mostra a importância e relevância da pesquisa considerando DAG e Web Crawlers.

2. REVISÃO BIBLIOGRÁFICA E BACKGROUND

Este capítulo retoma alguns trabalhos relacionados a Dados Abertos Governamentais (DAG) e Web Crawlers. Além de definir e mostrar as principais características de DAG e Web Crawlers, também é mostrado em ordem cronológica alguns trabalhos relacionados ao tema. Por sua vez, esses trabalhos foram analisados para buscar a relevância deles ao tema e a pesquisa desse trabalho.

2.1. BACKGROUND

Nesta sessão são explicados dois assuntos de extrema relevância para essa pesquisa, Dados Abertos Governamentais (DAG) e Web Crawlers.

Em relação ao DAG, é mostrada sua definição, características, princípios e formatos aceitos. No caso dos Web Crawlers é descrita sua definição, características e tipos de Web Crawlers.

2.1.1. Dados Abertos Governamentais

Dados Abertos Governamentais (DAG) são dados relativos ao governo publicados de forma livre e aberta. Esses dados são utilizados para melhorar a transparência na gestão pública, para ajudar na contribuição da sociedade com serviços inovadores ao cidadão, para aprimorar a qualidade dos dados em si, para viabilizar novos negócios e pela obrigatoriedade por lei (lei nº 12.527/2011). Segundo a lei, "qualquer pessoa pode usar dados abertos livremente, reutilizá-los e redistribuí-los, estando sujeito apenas a creditar a sua autoria e compartilhar pela mesma licença" (BRITO, COSTA e GARCIA, 2014).

Desde de 1957 os governos tentam deixar os dados relativos a si transparentes, porém a tecnologia da época dificultava o processo, sendo que somente anos depois, com a evolução das TICs houve a publicação massiva de DAG (BRITO, COSTA e GARCIA, 2014).

2.1.1.1. Características dos Dados Abertos Governamentais

Segundo a W3C, para um dado ser considerado aberto, ele precisa atender os seguintes requisitos: (OPEN, 2016).

1. Se o dado não pode ser encontrado e indexado, ele não existe.
2. Se ele não estiver aberto e disponível em formato compreensível por máquina, ele não

pode ser reaproveitado.

3. Se algum dispositivo legal não permitir sua replicação, ele não é útil.

De maneira geral todo dado aberto precisa atender a oito princípios: (OPEN, 2016) (CARTILHA, 2016)

1. Completos: Todos os dados precisam ficar disponíveis, incluindo documentos e gravações audiovisuais. Eles não podem ter controle de acesso e privacidade.
2. Primários: São publicados da mesma forma que foram coletados na fonte.
3. Atuais: Os dados são públicos o mais rápido possível para preservar o seu valor.
4. Acessíveis: São publicados para o público mais amplo possível e para os propósitos mais variáveis.
5. Processáveis por máquina: os dados são estruturados para possibilitar o seu processamento automatizado.
6. Acesso não-discriminatório: os dados são disponíveis a todos, sem que seja necessário identificação ou registro.
7. Formatos não proprietários: são disponíveis em formatos que ninguém tenha controle exclusivo.
8. Livres de licença: os dados não estão sujeitos a direitos autorais, marcas ou patentes.

Um grande erro cometido por muitos portais do governo é a publicação no formato "PDF", pois o mesmo não é um formato aberto, além de não ser estruturado a ponto que facilite sua leitura automática. Outro problema enfrentado é em relação à padronização, já que os portais utilizam diferentes formatos, aceitos ou não segundo os princípios da DAG, o que dificulta a reutilização dos dados. (BRITO, COSTA e GARCIA, 2014). Alguns formatos abertos e não proprietários previstos na cartilha são: (CARTILHA, 2016)

- JSON4 (Java Script Object Notation): baseado em texto e legível pelo ser humano. A especificação é a RFC 4627. Possibilita a serialização de estrutura de objetos complexos como listas. Muito usados em frameworks e bancos de dados.
- XML5 (Extensible Markup Language): é um conjunto de regras para codificar documentos como estrutura hierárquica e em formato legível por máquinas. Baseado por simplicidade, extensibilidade e usabilidade. Muito usado em troca de dados entre servidores Web (Web Services).
- CSV6 (Comma-Separated Values): valores separados por vírgulas, é um formato de

armazenamento de dados tabuladores em texto. Usado em estruturas de dados mais simples. Visualizável por programas livres e/ou proprietários.

- ODS7 (Open Document Spreadsheet): formato não proprietário de arquivo baseado em XML, padronizado pela ABNT8. Chamado de planilha, similar ao XLS da Microsoft, porém o formato é aberto.
- RDF9 (Resource Description Framework): Modelo estruturado em grafos e possui diversos formatos de serialização, como RDF/XML, Notation 3 e Turtle. Ele usa vocabulários disponíveis na Web, mais é possível criar novos vocabulários. Ele está no último nível dos dados abertos em relação a qualidade/complexidade.

Todos os formatos descritos são bastante utilizados na atualidade em diversas aplicações. O formato JSON ganhou destaque nos últimos anos, competindo com o XML. O formato CSV é o mais simples dos descritos, porém possui estrutura fácil de ser compreendida pelo ser humano, além de ser facilmente convertido para planilhas eletrônicas. O formato RDF por sua vez é o mais complexo dos mostrados, mas apresenta a melhor relação entre qualidade e complexidade dos dados.

2.1.2. Web Crawlers

Web Crawler é um programa que faz buscas automáticas na web. Geralmente um Web Crawler captura uma ou várias URLs de uma página e conforme for navegando entre as mesmas vai capturando outras URLs que respeitem os parâmetros passados, formando uma lista com elas (BAI, XIONG, *et al.*, 2014).

Para Ravi Bhushan (BAI, XIONG, *et al.*, 2014) um Web Crawler realiza as seguintes atividades: rastrear, indexar, buscar, filtrar, ordenar e classificar. Um Web Crawler pode ser dividido em alguns tipos:

- *Web Crawler de Propósito Geral:* realiza a coleta e processamento dos dados em um local específico.
- *Web Crawler Focado,* (CHEN, 2013)*:* direciona a busca através da utilização de algoritmos de análise, melhorando o desempenho durante a captura dos dados.
- *Web Crawler Incremental,* (TAN e MITRA, 2010)*:* possui a capacidade de incrementação. Com ele é possível fazer sucessivas leituras, aproveitando os conteúdos lidos anteriormente que se mantiveram e adicionando novos conteúdos.

Esse tipo é ideal para captura de dados em páginas que sofrem constantes mudanças.

- *Web Crawler Profundo* (SHARMA e SHARMA, 2011)*:* realiza buscas na Deep Web, ou seja, páginas que não são indexadas por buscadores comuns.

Foi utilizado um Web Crawler de propósito Geral para captura e análise de dados governamentais para este trabalho, pois o mesmo realiza buscas em um local centralizado (site de prefeituras).

2.2. REVISÃO BIBLIOGRÁFICA

As pesquisas realizadas neste trabalho estão relacionadas a temática de "Dados Abertos Governamentais" – DAG.

Para isso foram realizadas buscas nas bases de dados IEEE Explorer1, ACM2, Google Scholar3, SCOPUS4 e Elsevier5, utilizando as seguintes palavras chave: "Open Government Data", "Open Data", "Dados Abertos Governamentais", "Web Crawler", "Crawler".

A busca realizada retornou 62 artigos de 2010 a 2016, dos quais 34 foram selecionados para esse trabalho por estarem relacionados com a pesquisa. Os outros 29 foram excluídos baseados na leitura e resumo dos artigos.

Como esse trabalho tem como foco a captura automática, análise e visualização de dados abertos governamentais relacionados à educação, foi decidido relacionar a revisão bibliográfica, em ordem cronológica, de projetos relacionados executados no Brasil e no mundo.

2.2.1. Trabalhos Relacionados

Em 2009 o governo brasileiro começou uma iniciativa para a criação de base de dados com DAG. Em 2010 iniciou-se a conversão das bases de dados para formato RDF. O RDF (Resource Description Framework) é um dos formatos aceitos conforme a Cartilha (CARTILHA, 2016) .

[1] http://ieeexplore.ieee.org/
[2] http://dl.acm.org
[3] http://scholar.google.com
[4] http://www.scopus.com
[5] http://www.elsevier.com.br/site/Default.aspx

Neste mesmo ano, o escritório do W3C do Brasil realiza um importante processo educacional para uso correto das Tecnologias de Informação e Comunicação (TICs) voltado aos funcionários do setor público, para que os dados abertos sejam publicados de forma correta. Foram desenvolvidas metodologias e ferramentas (BREITMAN, SALAS, *et al.*, 2012). Porém iniciativas como essa não estão sendo tomadas no âmbito dos governos municipais.

Mundialmente, o movimento de dados abertos, chamado de "Open Government Partnership (OGP)", iniciou-se em 2011, tendo como fundadores oito países (Brasil, Indonésia, México, Noruega, Filipinas, África do Sul, Reino Unido e Estados Unidos), sendo que posteriormente, mais 63 países aderiram ao movimento (BRITO, COSTA, *et al.*, 2014).

Foram lançadas diferentes iniciativas para montagem de bases de dados abertos. Em fevereiro de 2014, os EUA continham cerca de 89.000 bases de dados, O Reino Unido, 13.000. Já o Brasil no mesmo período continha apenas 200 (BRITO, COSTA, *et al.*, 2014).

Segundo Brito (BRITO, COSTA, *et al.*, 2014), no Brasil, "a lei nº 12.527/2011, conhecida como "Lei de Acesso à Informação", torna obrigatória a publicação de DAG pelo governo, seja ele municipal, estadual ou federal. Também estão incluídos os Tribunais de Conta e Ministério Público. Além disso, a lei também estabelece padrões que os governos precisam adotar para fazerem a publicação desses dados.

Hoje existem diversos portais no Brasil disponibilizando os dados abertos, porém a maioria não segue os princípios legais. Um dos principais motivos para o não seguimento da lei é em relação a imaturidade dos mesmos em seguir os requerimentos.

Em 2012 foi publicado um artigo relacionado com "Linked Open Government Data" (DING, PERISTERAS e HAUSENBLAS, 2012). Este artigo mostrou que um dos grandes problemas na publicação de dados abertos é devido à descentralização nessas publicações, além do uso de diversos formatos. Uma das formas de se resolver esses problemas é o uso do "Linked Government Data", que prevê a interligação desses dados para facilitar o reuso e a reutilização. A Inglaterra é a pioneira nessa iniciativa. Mas para isso é necessário obedecer a três estágios: 1º, o governo precisa publicar os dados em formatos reusáveis e de fácil localização, 2º, a comunidade (indústria e academia) gera as ligações entre os dados e 3º, desenvolvedores publicam esses dados através do desenvolvimento de aplicações. Porém no Brasil enfrenta-se problemas já no primeiro estágio, já que os dados não são publicados em formatos adequados e reutilizáveis.

Em julho de 2013, a cidade de Recife, capital do Estado de Pernambuco lançou a aplicação Cidadão Inteligente (BRITO, COSTA, *et al.*, 2014). Ela foi construída devido a um concurso com o objetivo de incentivar a produção de aplicativos e projetos para ajudar os

cidadãos recifenses. Nesse concurso concorrem projetos conceituais e APPs. O APP Cidadão Inteligente ganhou o concurso, dente 23 aplicações submetidas (BRITO, COSTA e GARCIA, 2014).

Em agosto de 2013 a aplicação Rio Inteligente foi lançada na cidade do Rio de Janeiro com o objetivo de criar uma aplicação para ajudar cidadãos e turistas. Ela possui diversas funcionalidades, desde lista de hospitais, carteira vacinação on-line e recursos de educação (BRITO, COSTA, *et al.*, 2014).

Entre setembro e outubro de 2013 foi realizada uma pesquisa na Universidade de São Paulo em relação a portais da internet de dados abertos de 20 prefeituras do estado de São Paulo. Foram avaliados na pesquisa formato de arquivos, possibilidade de leitura de dados por máquina, estrutura, autenticidade e integridade, atualização dos dados, informações de contato, e acessibilidade. Dos requisitos apresentados, 44% das prefeituras os atenderam, 32% atenderam de forma parcial e 24% atenderam. Ao olhar esses dados, a maioria das prefeituras não atende os requisitos relacionados a forma de publicação dos dados, além de grande parte dos que atende, não os atende de forma completa. Apenas 24% das prefeituras pesquisadas fazem a publicação de dados respeitados as diretrizes adequadas para a publicação de DAG. Nesta pesquisa, mas analisando o formato dos arquivos que eram publicados os dados, 68% dos dados estavam no formato HTML, 22% em PDF e 10% em formatos abertos. Nota-se que umas porcentagens muito pequenas (10%) dos locais pesquisados publicam os dados usando formatos adequados segundo os princípios DAG. Conclui-se que a maioria das prefeituras ao analisar a amostra dessa pesquisa, não cumpre os princípios para a publicação de DAG (CORRÊA, CORRÊA e SILVA, 2014).

Em maio de 2013, o presidente dos Estados Unidos, Barack Obama assinou uma lei que os dados do governo dos EUA seriam publicados de forma aberta e "interpretada por máquinas". A expectativa era que esses dados trouxessem diversos benefícios nas áreas de saúde, energia, educação, segurança pública, finanças e desenvolvimento global (JETZEK, AVITAL e BJORN-ANDERSEN, 2014).

Em outubro 2013 a Câmera dos Deputados do Brasil promoveu o primeiro "Hackathon", com o objetivo de criar aplicações para interligar o parlamento com a sociedade brasileira. Foram submetidos 99 projetos e 27 foram selecionados para a etapa final da seleção. A aplicação Meu Congresso Nacional foi a vencedora. A aplicação é adaptada para web e dispositivos móveis, e tem como foco a transparência dos dados dos parlamentares. É exibido dados de gastos com telefone, transporte, viagens, serviços, etc. de cada parlamentar (BRITO, COSTA e GARCIA, 2014).

Em 2014, Kaschesky e Selmi desenvolveram uma pesquisa onde foi usado o "7R Data Value Framework", um framework para dados abertos que identifica problemas com abordagens existentes para fazer dados abertos e conectar aplicações, empresas entre outros, além de sugerir fazer dados abertos mais eficientes e utilizáveis (KASCHESKY e SELMI, 2014).

Em 2015 foi publicado um estudo sobre dados abertos de Cuiabá, capital do Estado do Mato Grosso com foco em casos de Dengue no município, que inicialmente estavam em formatos não condizentes com os princípios dos Dados Abertos (MENDONÇA, MACIEL e VITERBO, 2015). Esses dados foram tratados e posteriormente visualizados através de mapa, onde é possível verificar os principais focos de Dengue do local.

2.2.2. Análise dos Trabalhos

Ao analisar os trabalhos, foi verificado do ponto de vista temporal uma grande publicação de trabalhos relacionados ao tema no ano de 2012, como mostra no gráfico (**Erro! F onte de referência não encontrada.**). Nota-se uma diminuição de trabalhos nos anos antes e depois do ano em questão.

Figura 2: Número de Artigos ao longo do tempo

Dos trabalhos analisados, foram estabelecidos os seguintes requisitos como forma de classificação:

R1: Disponibilizar solução compatível com cidades inteligentes

R2: Disponibilizar solução relacionada com DAG

R3: Dados em formatos conforme os princípios de DAG

R4: Coleta dos dados de forma automatizada

R5: Reaproveitamento de Infraestruturas já existentes

R6: Prover solução que possa ser utilizada por outros desenvolvedores.

Uma matriz, disposta na Tabela 1 mostra a disposição desses requisitos nos trabalhos citados anteriormente juntamente com a API desenvolvida nesta dissertação:

Tabela 1: Tabela de Requisitos

Trabalhos	**Requisitos**					
	R1	**R2**	**R3**	**R4**	**R5**	**R6**
Linked Government Data	X	X	X			X
Cidadão Inteligente	X	X	X			
Rio Inteligente	X	X	X			
Meu Congresso Nacional	X	X	X		X	X
7R Data Value Framework	X	X	X		X	X
Mapeamento Casos de Dengue	X	X	X			X

Desta maneira, não foram encontrados, até o momento, trabalhos que realizasse a coleta automatizada de Dados Abertos Governamentais e os disponibilizasse. Todavia, existem trabalhos que consideram dados de outros domínios, como uma plataforma de captura de dados abertos (VIEIRA e ALVARO), aplicativos como Cidadão Inteligente (BRITO, COSTA, *et al.*, 2014) e Rio Inteligente (BRITO, COSTA, *et al.*, 2014), além do desenvolvimento de frameworks, como o "7R Data Value Framework" (BRITO, COSTA, *et al.*, 2014).

2.3. CONSIDERAÇÕES FINAIS

A publicação de DAG além de uma obrigação legal é uma ferramenta estratégica dos governos. Porém essa publicação precisa seguir alguns princípios e diretrizes relacionadas a estrutura, a formatos, etc. Isso tudo permite que os DAG publicados possam ser aproveitados de forma automatizada em outras aplicações.

Hoje no Brasil, é feita a publicação de DAG, porém em muitos casos essas publicações não seguem os princípios e formatos aceitos, o que dificulta a utilização desses dados.

Por causa disso, a proposta deste trabalho é a captura desses dados e a disponibilização dos mesmos seguindo os princípios DAG. A captura é feita de forma automatizada através da

utilização de Web Crawlers, que são softwares que fazem busca de dados em páginas da internet.

Foi feito um estudo bibliográfico sobre DAG e Web Crawlers e percebeu-se que até o momento não existem trabalhos que utilizam Web Crawlers para a captura de DAG no contexto de municípios Brasileiros.

3. ESTUDO DOS DADOS ABERTOS GOVERNAMENTAIS DE MUNICÍPIOS BRASILEIROS

Foi realizada uma pesquisa com sites de 44 prefeituras do Estado de São Paulo. As pesquisas concentram-se nos sites da transparência de todas elas. Foi detectado informações como:

- Estrutura geral dos sites
- Formatos em que os dados eram disponibilizados (Páginas HTML, Documentos PDF editáveis ou não, planilhas em formatos proprietários, planilhas em formatos abertos, CSV, XML, JSON, RDF, etc.)
- Tipos de Dados disponibilizados (dados de receitas, despesas, recursos humanos, etc.)
- Relatórios disponíveis com DAG (Balanços, balancetes, relatórios financeiros, etc.)
- Quantidade de dados disponibilizados

Todos esses dados foram levantados para auxiliar na escolha dos portais em que teriam os dados capturados. Na Figura 3 são mostradas as cidades pesquisadas com suas respectivas populações:

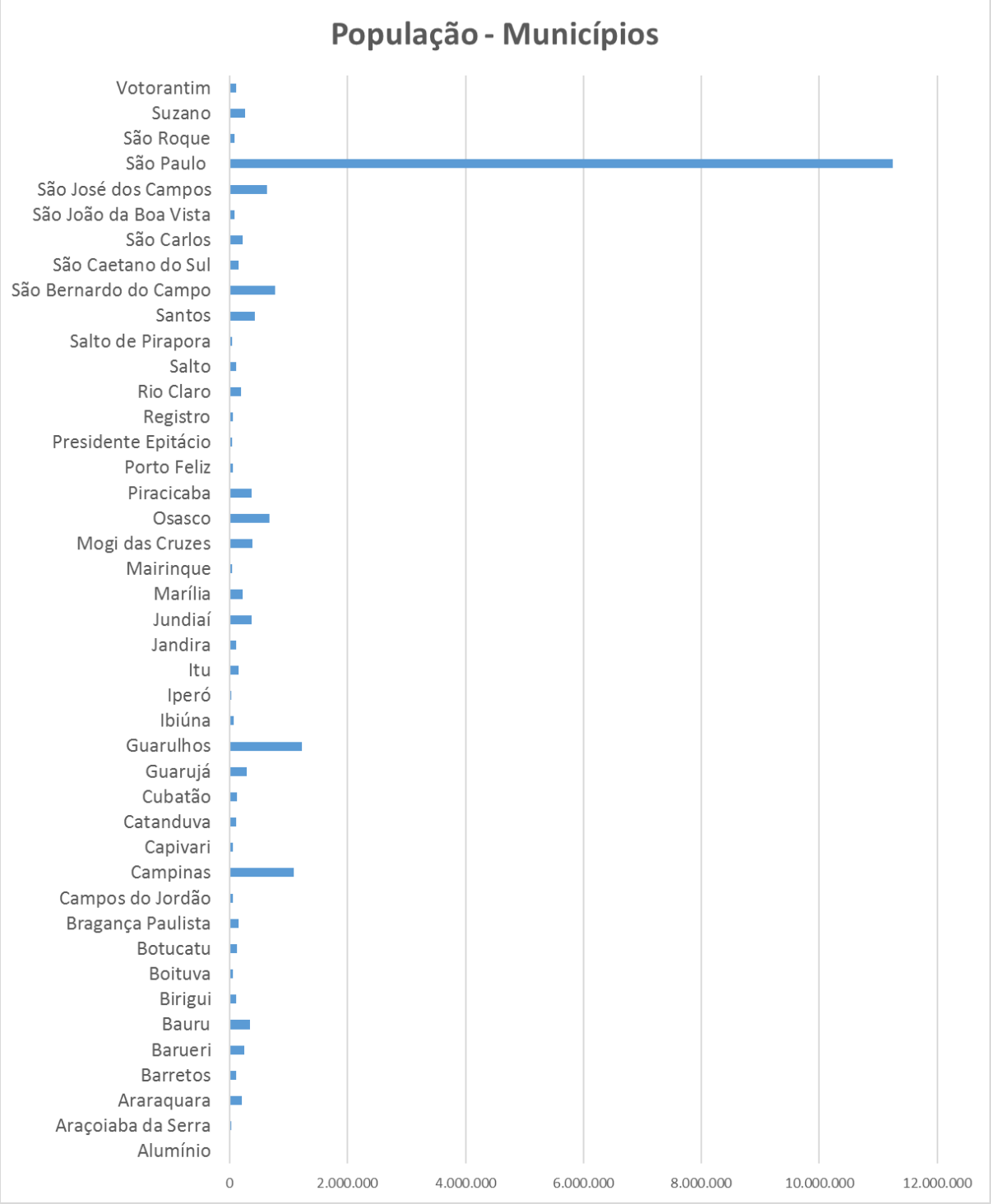


Figura 3: Lista de Cidades Pesquisas com suas respectivas populações

Após a pesquisa os dados foram selecionados e analisados e conseguiu-se concluir sobre os formatos mais usados e quais prefeituras utilizam em si portais da transparência. A partir desses resultados que foi norteado a escolha de quais portais seriam usados para a captura dos dados.

Como mostrado no gráfico da Figura 4, 46% das cidades pesquisadas utilizam o formato PDF (editável) para exibir seus dados abertos. 12% utilizam PDF, porém gerado com imagens, o que dificulta seu processamento automático. 21% das cidades mostram os dados no formato HTML, 12% em CSV, 9% em XLS e nenhuma cidade pesquisada exibiu os dados no formato RDF.

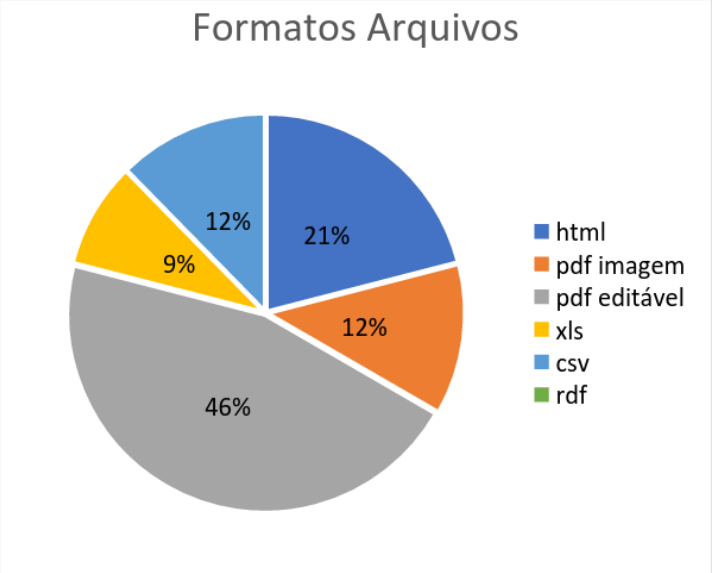


Figura 4: Gráfico de Formato de Arquivos

Na Figura 5 é mostrado um gráfico onde 77% das prefeituras pesquisadas possuem um site específico da transparência da cidade.

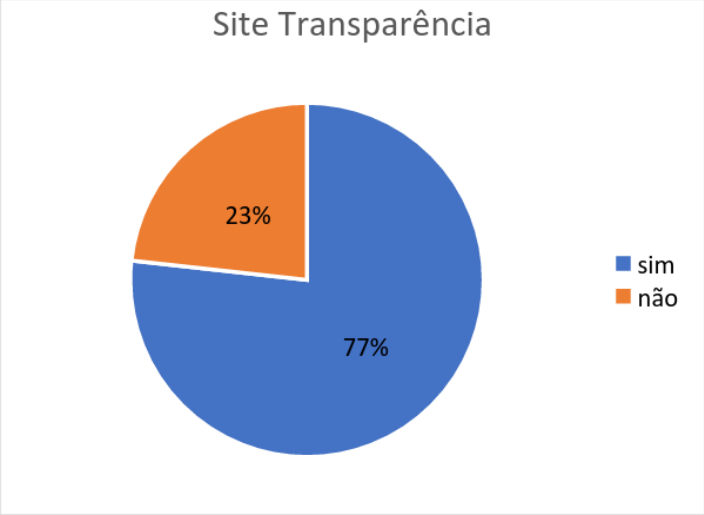


Figura 5: Gráfico Site da Transparência

Como conclusão, os formatos PDF e HTML são os formatos mais utilizados pelas prefeituras, porém os mesmos são inadequados, pois dificulta a reutilização e o processamento dos mesmos por máquinas.

Isso mostra que a preocupação com a publicação de dados abertos é real para as cidades, porém os mesmos são publicados em formatos inadequados.

Outro ponto é em relação à padronização dos relatórios e formatos, pois hoje, as prefeituras usam diferentes relatórios entre si. Não há uma padronização e exigência em relação a esses documentos. Algumas prefeituras, por exemplo, publicam dados relacionados à Educação em um relatório específico, já outras não. Durante a pesquisa foi verificado também que existem alguns sistemas específicos sobre transparência, que são usados por mais de uma prefeitura. Conclui-se que apesar de um sistema ser utilizado por mais de uma prefeitura, não existe um sistema padrão usado por todas.

A falta de maturidade das prefeituras poderia ser um motivo que leva a essa publicação errada, principalmente devido à natureza recente da lei e a disponibilização de recursos tecnológicos e profissionais qualificados nos órgãos específicos dessas prefeituras.

Logo hoje tem-se um ambiente com muitos dados úteis sobre o governo, mas muitos com dificuldades de uso. Uma das formas de uso desses dados seria com a utilização de Web Crawlers, que permitiria uma posterior análise desses dados e até mesmo a republicação dos mesmos usando formatos corretos.

Um Web Crawler é capaz de capturar os dados em páginas, mas para que essa captura seja feita, é preciso levar em consideração primeiramente como a página é construída (DHENAKARAN e SAMBANTHAN, 2011). A leitura realizada é feita de forma metódica, onde a partir de uma URL inicial o software irá ler os elementos HTML na página e assim que encontrar outra URL irá para a página correspondente a essa URL e irá ler os elementos destas, e assim por diante. Logo, de forma resumida, um Web Crawler lê elementos HTML, sejam eles links, tabelas, parágrafos, etc.

Diversos fatores dificultam a captura dos dados em uma página, entre eles:

- *Imagens:* em muitos portais dados textuais foram publicados em forma de imagem, dificultando a leitura, já que o Web Crawler consegue identificar a imagem, mas não o texto que está nela.
- *URLs não amigáveis:* o uso de URLs não amigáveis prejudica capturas onde é necessário manipulá-las, como para passagem de parâmetros, já que a mesma possui uma estrutura com dificuldades de interpretação.
- *URLs ocultas:* o uso de URLs ocultas dificulta a leitura, já que não se consegue capturar

uma página através link, já que o mesmo fica oculto.

- *Páginas em flash:* assim como imagens, dados mostrados no formato flash sofrem problemas ao serem interpretados por Web Crawlers.
- *Quantidade excessiva de elementos HTML:* a leitura de uma página pelo um Web Crawler consiste em uma leitura sequencial de cada elemento HTML. O excesso de elementos de forma desorganizada dificulta a leitura e classificação dos dados lidos.
- *Sessões e Cookies:* sites que utilizam sessões ou cookies para a navegação entre páginas enfrentam problemas em sua captura, já que sem os dados da sessão, ao navegar entre as páginas, as mesmas apresentaram problemas de carregamento.

Esses fatores prejudicam a captura dos dados de uma página. Alguns são de fácil resolução, outros não. Por causa disso, a escolha dos portais levou em consideração esses fatores que estão contidos na estrutura dos sites em si.

Para isso todos os portais pesquisados passaram por uma classificação (baixa, média e alta) considerando os critérios descritos acima, como estrutura da página, possibilidade de download dos dados em formatos abertos, URLs amigáveis, etc. Com isso, os portais pesquisados foram classificados e no final foram contabilizados da seguinte forma (Tabela 2):

Tabela 2: Complexidade dos portais

Complexidade	**Nº Portais**	**Portais**
Baixa	6	Capivari, Jundiaí, Piracicaba, Salto, São Bernardo do Campo, São Roque.
Média	16	Alumínio, Barretos, Bauru, Botucatu, Bragança Paulista, Guarulhos, Ibiúna, Iperó, Jandira, Mairinque, Marília, Osasco, Porto Feliz, São Caetano do Sul, São Carlos, Votorantim.
Alta	22	Araçoiaba da Serra, Araraquara, Barueri, Birigui, Boituva, Campinas, Campos do Jordão, Catanduva, Cubatão, Guarujá, Itu, Mogi das Cruzes, Presidente Epitácio, Registro, Rio Claro, Salto de Pirapora, Santos, São João da Boa Vista, São José dos Campos, São Paulo, Sorocaba, Suzano.

Através dessa pesquisa foram escolhidos os portais das cidades de Capivari e São Roque como *case* inicial para o desenvolvimento do Web Crawler. Ambos portais possuem os dados organizados em tabelas, em formato HTML e com URL parcialmente amigáveis. Nos portais não há a possibilidade de download dos dados em formatos abertos. Porém ambos portais utilizam cookies para permitir a navegação. Os cookies não permitiam a leitura individualizada de cada página apenas por uma URL, logo foi necessário fazer a captura inicial desses cookies para posteriormente fazer a leitura das páginas.

Ambas prefeituras utilizam uma forma semelhante de organização dos dados. Na página inicial, elas possuem um menu principal de acesso às principais páginas do portal, uma caixa de seleção para a escolha do ano que se deseja obter os dados e uma caixa de seleção da instituição desejada. Em ambas cidades, foram capturados os dados de Despesa relacionados ao Orçamento e à Execução.

3.1. CAPIVARI

Situada no interior de São Paulo, com população de 53.731 habitantes. Economia baseada em serviços e na indústria. Na educação, no ano de 2013 teve o IDEB (Índice de Desenvolvimento da Educação Básica) de 4,9, acima da meta estadual (4,4) e igual a meta nacional.

A prefeitura disponibiliza um portal da transparência separado do site principal no seguinte endereço: http://capivari.giap.com.br/apex/capivari/f?p=297:1 (Figura 6).

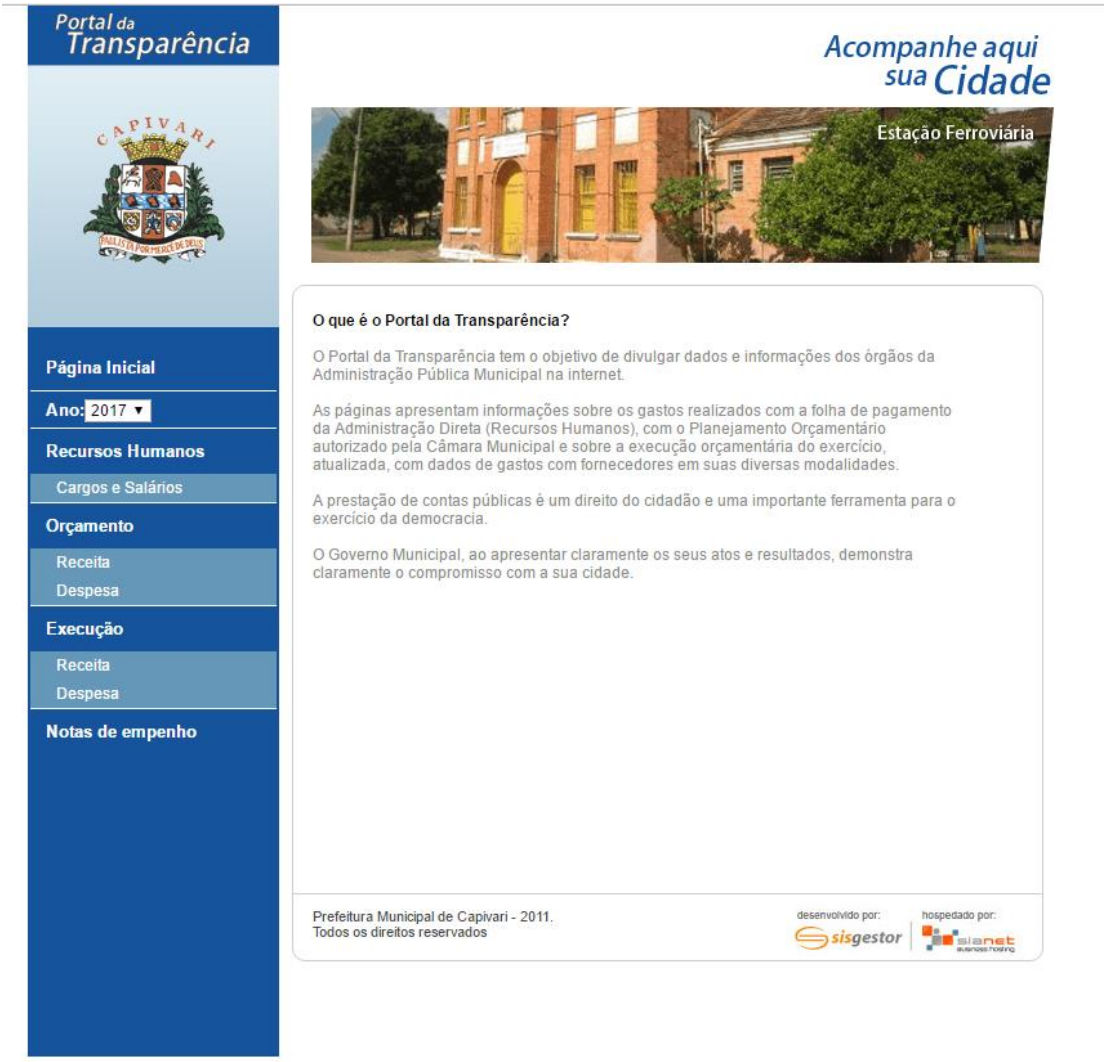


Figura 6: Página Inicial da Transparência de Capivari

O site conta com um menu vertical de acesso as principais páginas do site (Figura 6). Os dados capturados foram os disponíveis nos menus Orçamento → Despesa e Execução → Despesa.

Foram capturados os dados de todos os anos disponibilizados pelo site no momento da captura, no caso do período de 2010 a 2016. Por padrão, ao carregar o site, são mostrados os dados do ano atual, para visualizar dados de outros anos, é necessário escolher o ano desejado na caixa de seleção “Ano”, disponível juntamente com os menus na página. Automaticamente a página é recarregada e os dados são mostrados seguindo a mesma estrutura.

Os dados são dispostos em tabelas e são agrupados, seguindo departamentos, unidades, fornecedores, etc. Cada dado é um link que leva para outra página que mostra de forma detalhada os dados do link em questão. Por exemplo, o link “Coordenadoria Municipal de Defesa Civil”, ele encaminha para uma página onde os dados relacionados a esses departamentos são depostos em outras categorias, e assim por diante (Figura 7).

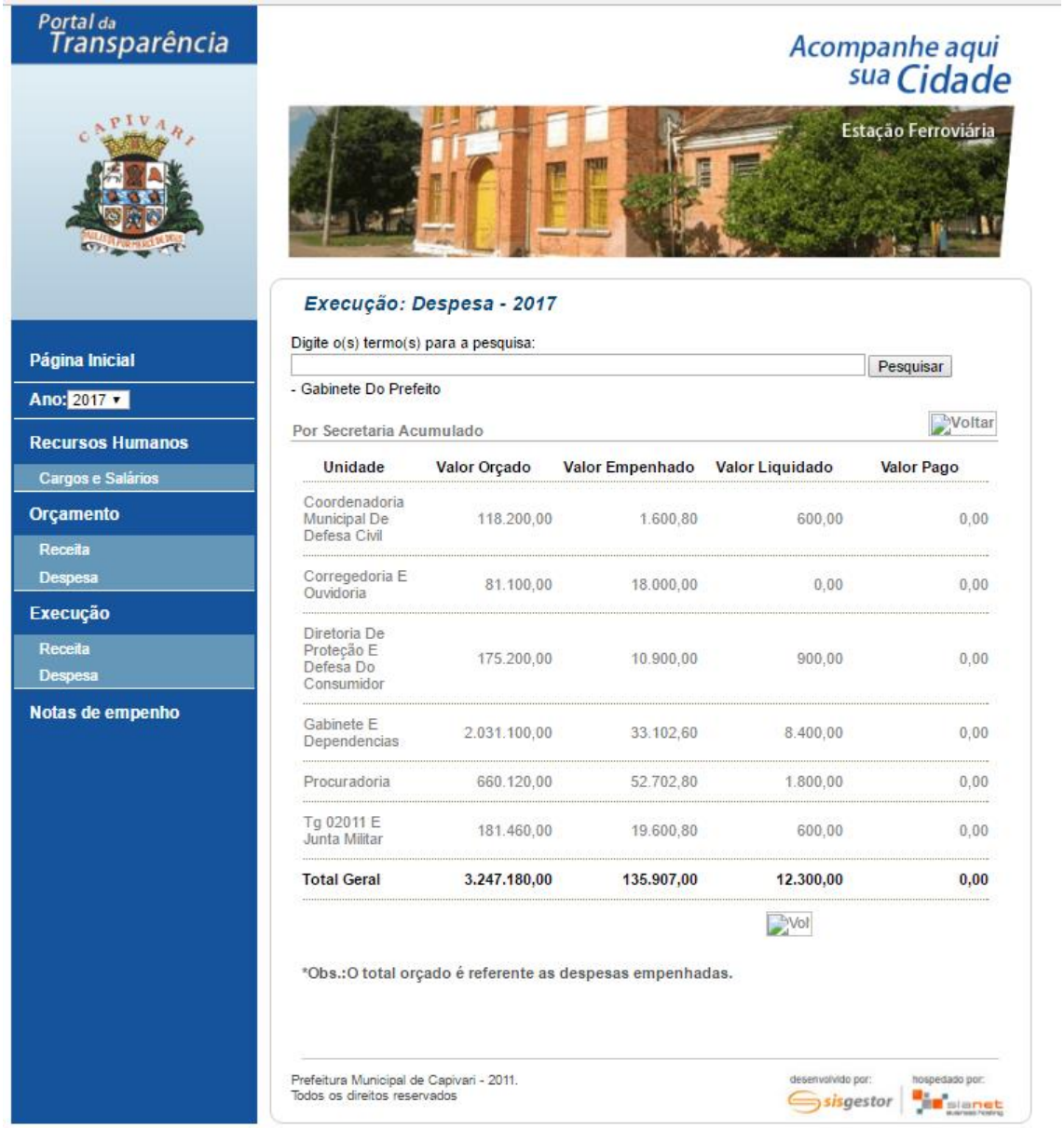


Figura 7: Dados do site de Capivari

No caso dos dados Relacionados à Execução, foram capturados os dados do primeiro nível até o nome dos fornecedores por motivos de desempenho e para equiparar os dados capturados com o da prefeitura de São Roque.

3.2. SÃO ROQUE

Cidade situada no interior de São Paulo, com população de 87.821 habitantes. Sua economia é focada em serviços, seguindo da indústria e da agropecuária. Na área da educação,

no ano de 2013, ela obteve o IDEB (Índice de Desenvolvimento da Educação Básica) de 4,1, um pouco abaixo do índice do estado (4,4) e do Brasil (4,9). {fonte IBGE cidades – 12/01/2017}

A prefeitura disponibiliza um portal da transparência separado do site principal, disponível no endereço: http://sroque.giap.com.br/apex/sroque/f?p=292:1 (Figura 8).:

Figura 8: Página Inicial do Portal da Transparência da cidade de São Roque

O site conta com um menu horizontal de acesso as principais páginas do site. Os dados capturados foram os disponíveis nos menus orçamento → Despesa e Execução → Despesa.

Foram capturados os dados de todos os anos disponibilizados pelo site no momento da captura, no caso do período de 2011 a 2016. Por padrão, ao carregar o site, são mostrados os dados do ano atual, para visualizar dados de outros anos, é necessário escolher o ano desejado na caixa de seleção “Ano”, disponível juntamente com os menus na página. Automaticamente a página é recarregada e os dados são mostrados seguindo a mesma estrutura.

Os dados são dispostos em tabelas e são agrupados, seguindo departamentos, unidades, fornecedores, etc. (Figura 9). Cada dado é um link que leva para outra página que mostra de forma detalhada os dados do link em questão. Por exemplo, o link “Departamento de Administração”, ele encaminha para uma página onde os dados relacionados a esses departamentos são depostos em outras categorias, e assim por diante.

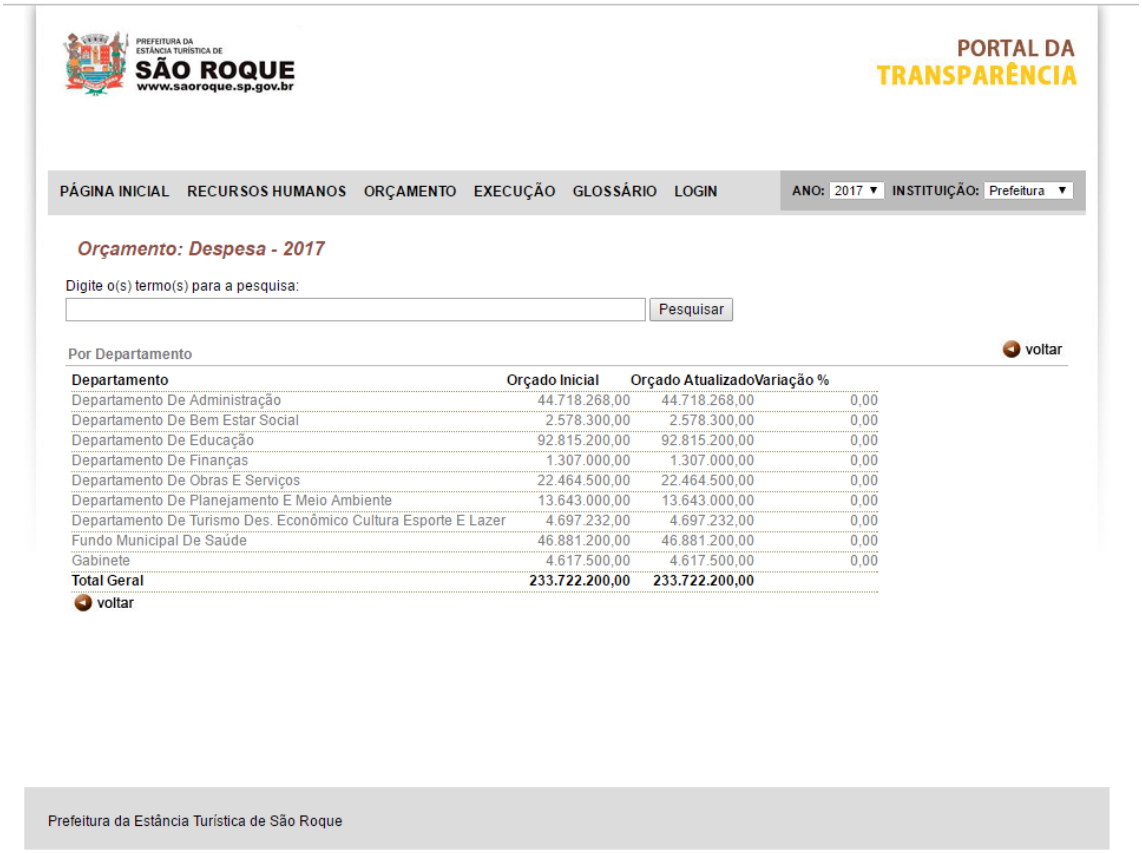


Figura 9: Exemplo da disposição dos dados no site da transparência da prefeitura de São Roque

3.3. CONSIDERACOES FINAIS

Ao analisar sites de diferentes prefeituras do estado de São Paulo pode-se comprovar que atualmente a maioria preocupa-se em publicar DAG, porém não seguem os princípios e formatos para a disponibilização desses dados. Para resolver esse problema foi proposta a captura desses dados de forma automatizada através de um Web Crawler.

Devido isso foi realizada uma classificação de todos os portais pesquisados em níveis de complexidade para a captura dos dados utilizando um Web Crawlers. Após isso foram escolhidos os sites das prefeituras de Capivari e São Roque como case inicial para a captura dos dados.

4. ABORDAGEM PARA CAPTURA AUTOMATIZADA DE DADOS ABERTOS GOVERNAMENTAIS

Como proposta para a resolução do problema com DAG em prefeituras nos municípios brasileiros, foi construído um software para realizar a captura desses dados e disponibilização em formatos adequados.

O software em si é dividido em dois principais: Web Crawler e API DAG Prefeituras. O Web Crawler é responsável pela captura, análise e armazenamento dos dados. Já a API DAG Prefeituras é responsável pela disponibilização dos dados.

Foi escolhido, como case inicial prefeituras que disponibilizavam os dados através de páginas HTML. Para a escolha das prefeituras foi inicialmente realizada uma pesquisa em diversos sites, onde foram escolhidos os sites das seguintes prefeituras: Capivari e São Roque.

4.1. DESENVOLVIMENTO

Foi construído um Web Crawler com a função de realizar a captura dos dados dos portais escolhidos de forma automática, analisá-los e armazená-los em um banco de dados e disponibilizá-los em formatos abertos através da API DAG Prefeituras. A leitura é feita link por link e recursivamente são lidos todos os links de todas as páginas. Na Figura 10 é apresentada a arquitetura do software.

A arquitetura do software é baseada em três fases principais, descritas na Figura 10:

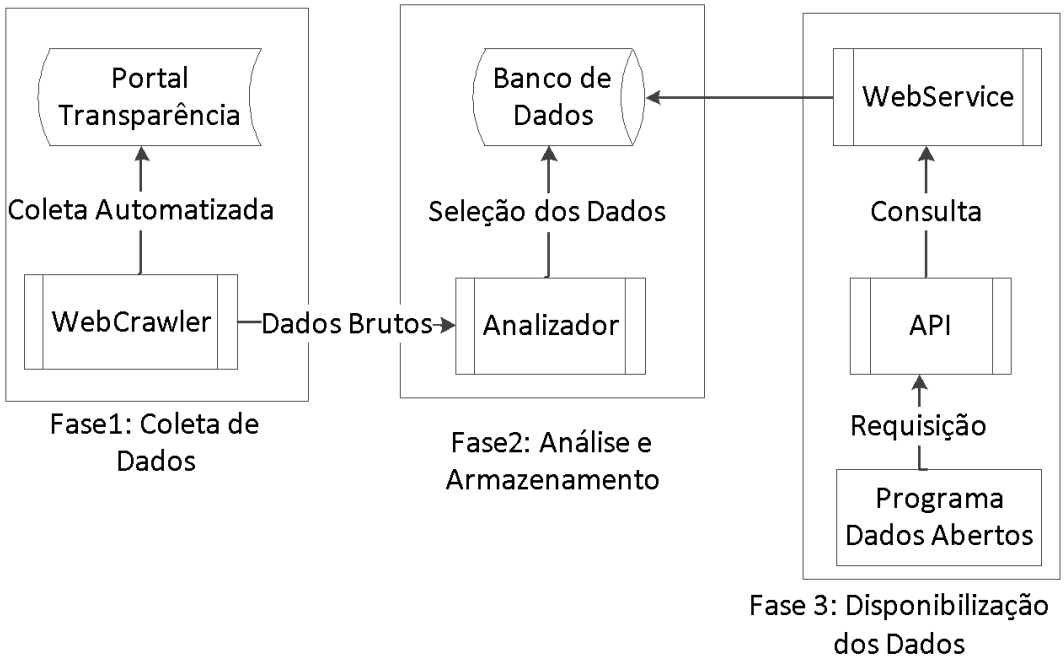


Figura 10: Arquitetura do Web Crawler da API DAG Prefeituras

4.1.1. Coleta De Dados

Nessa etapa é feita a busca e captura dados de portais da transparência dos municípios através de Web Crawler que automatiza o processo.

A partir de um link inicial, o programa faz a leitura dos links de uma página, cria objetos "Página" para cada link lido e continua de forma recursiva a leitura de links até encerrar-se os links ou for solicitada a parada da leitura pelo programa. No final terá uma estrutura de objetos página ligados entre si.

O código na Figura 11 é um trecho do programa onde é feita a leitura inicial dos elementos da página a partir de um link. Esse código faz parte de uma função que recebe como um dos parâmetros um objeto do tipo "Página" (psar), onde o atributo "getURL" contém a URL inicial para o início da leitura da página. Inicialmente é necessário criar-se um objeto do tipo "WebClient" proveniente da API HtmlUnit, capturar os cookies da página como o método "setCookieManager", e capturar a página em si com o método "getPage", todos provenientes na API HTMLUnit. Em seguida deve-se selecionar o ano na página capturada (pois na página original é possível carregar os dados de diversos anos ao clicar no combobox "ano"). Para isso deve-se usar a função "selecionaAno". Já esta função foi criada neste trabalho. O retorno é salvo em um objeto do tipo "HtmlPage", que por sua vez é passado para uma função de outra API, Jsoup. Essa API, por sua vez, cria um objeto do tipo "Document", onde contém os dados da página através do método parse. E através do "Document" é possível navegar pelos elementos através do tipo. No caso, a navegação foi através do tipo "a[href]", que correspondem a links em HTML.

```
WebClient webClient = criaWebClient();
HtmlPage page;
String idTabela = "report_R449736245754299619";
       if(status == 1){ //pagina inicial- criar cookies novos
           psar.setCookies(webClient.getCookieManager());
           webClient.getOptions().setJavaScriptEnabled(false);
           page = webClient.getPage(psar.getUrl());
       }
       else{ //paginas restantes 2- chamada recursiva, 3 - sem chamada
recursiva, usar cookies existentes
           webClient.setCookieManager(psar.getCookies());
           page = webClient.getPage(psar.getUrl());
           page = this.selecionaAno(page, ano,"p_t01");
       }
 Document doc = Jsoup.parse(page.asXml(),page.getBaseURL().toString());
 Elements links = doc.select("a[href]");
```

Figura 11: Captura dos Cookies e dos Links de uma página usando o Web Crawler

A partir desse ponto consegue-se uma lista de links lidos na página. As páginas que esses links redirecionam por sua vez passarão pelo mesmo processo descrito acima, e de forma recursiva e hierárquica todos os links da página serão lidos até se chegue na última página do site.

São feitos outros testes e conforme o programa é executado são criados diversos objetos do tipo “Página” que são ligados entre si da mesma forma que os links são ligados.

Posteriormente esses objetos são salvos em um banco de dados, onde cada objeto é um registro em uma tabela.

4.1.1.1. Utilização Do Jsoup E Do HTMLUnit

Para que a captura dos dados pudesse ser realizada de forma satisfatória, ambas APIs foram combinadas. A JSOUP ficou responsável por fazer a leitura de link por link das páginas e também da leitura dos dados disponíveis em tabelas que não eram links. Já a HTMLUnit foi utilizada para a seleção dos anos de captura dos dados, que utiliza um evento JavaScript cada vez que um ano é selecionado na caixa de seleção e para a captura dos Cookies (Figura 12). Em ambos portais há utilização de cookies que não permite que uma URL seja carregada corretamente sem os mesmos. Para que pudesse ocorrer a captura, na leitura da primeira página esses cookies foram capturados e repassados durante a leitura das páginas seguintes.

A utilização de ambas APIs em conjunto foi satisfatória para obter os dados com desempenho satisfatório e de maneira correta. Ambas conseguem trabalhar de forma integrada.

```
WebClient webClient = criaWebClient();
HtmlPage page;
webClient.setCookieManager(psar.getCookies());
page = webClient.getPage(psar.getUrl());
```

Figura 12: Atribuição dos Cookies para outra página

No código da Figura 12 é utilizado objetos provenientes da API HTMLUnit, onde é criado um objeto do tipo “WebClient” para que posteriormente seja capturado os cookies da página em questão (representada por um objeto do tipo “Página” na variável “psar”).

```
Document doc = Jsoup.parse(page.asXml(),page.getBaseURL().toString());
```

Figura 13: Criação do objeto do tipo "Document" que irá armazenar os dados lidos pelo Web Crawler

Na linha de código da Figura 13 é criado um objeto “doc”, que posteriormente possibilita a navegação dos elementos capturados pela tag HTML (a – link, table – tabela, etc.). Esse trecho utiliza recursos provenientes da API Jsoup, mas note-se que como parâmetro é passado o objeto “page”, que é do tipo “HTMLPage”, criado no trecho anterior utilizando a API HTMLUnit. Conclui-se que ambas APIs trabalham de forma integrada.

4.1.1.2. Criação de Objetos

Devido à forma hierárquica da disposição dos links de ambas as páginas, foi criado um tipo de objeto específico para a construção da API. Esse objeto foi chamado de “Página” e representa um link, pois um link é o endereço para uma página. Nessa estrutura, cada link capturado é transformado em um objeto do tipo “Página”, que por sua vez pode ser ligado com outros objetos do mesmo tipo de forma hierárquica assim como ocorre no site original Figura 8. Logo, conclui-se que um objeto do tipo “Página” pode ter zero ou mais objetos do mesmo tipo. Na Figura 14 é mostrada a estrutura do objeto “Página” (Figura 14), que além de conter o link em si, possui algumas informações importantes do mesmo, como o texto mostrando no link, o ano, os cookies e listas de Orçamento e Execução.

```
public Pagina(String u, String tu, int a, CookieManager c,
ArrayList<OrcamentoUnidade> lo, ArrayList<ExecucaoUnidade> le){
        this.url = u;
        this.textoUrl = tu;
        this.ano = a;
        this.cookies = c;
        this.listaOrcamento = lo;
        this.listaExecucao = le;
    }
```

Figura 14: Estrutura do objeto "Página"

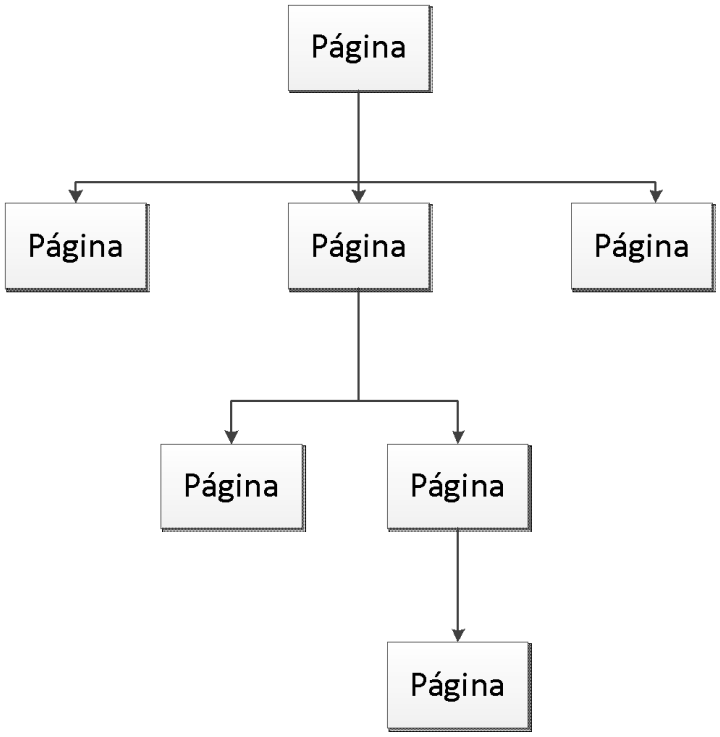


Figura 15: Estrutura de Objetos "Página"

Essa estrutura de "páginas" (Figura 15) é criada para todos os links encontrados durante a captura. A cada leitura os links são lidos e organizados numa estrutura de objetos, onde cada um representa um link. Como um link é um endereço de uma página, esses objetos foram chamados de "Página".

Porém, quando o programa encontra a última página, os dados também estão organizados em uma tabela, mas os textos que estão nela não são links. Nesse caso, devido à falta de links, não é possível gerar um objeto página com esses dados. Devido a isso, a leitura desses dados precisa ser feita analisando uma "Tabela HTML" e não "Links". O Web Crawler passa então a procurar o elemento "tabela" na página, e não o elemento "link".

Por causa disso, foram criados outros dois objetos, relacionados respectivamente ao Orçamento e a Execução: "OrçamentoUnidade", "ExecucaçãoUnidade". Eles irão armazenar os dados relativos as tabelas encontradas na última página dos sites, que no caso não possui links. Esses objetos possuem uma estrutura diferente do objeto "Página", já que ele não utiliza uma estrutura hierárquica. Nesse caso, durante a leitura dos dados, é criado um objeto para cada linha da tabela, sendo os atributos as colunas da tabela.

Foi criado um objeto separado para orçamento e execução devido às diferenças dos dados (colunas nas tabelas) em ambos os casos.

Esses objetos são armazenados dentro do seu objeto "Página" de origem (Figura 16), ou seja, se tenho uma página X que direciona para uma página sem links (do tipo Orçamento

ou Execução), este objeto “Página” terá armazenado dentro uma lista de objetos do tipo “OrçamentoUnidade” ou “ExecucaoUnidade”. É utilizada uma lista porque cada linha da tabela representa um objeto, logo como o link direciona para uma página contendo uma tabela, conclui-se que essa tabela gerará diversos objetos.

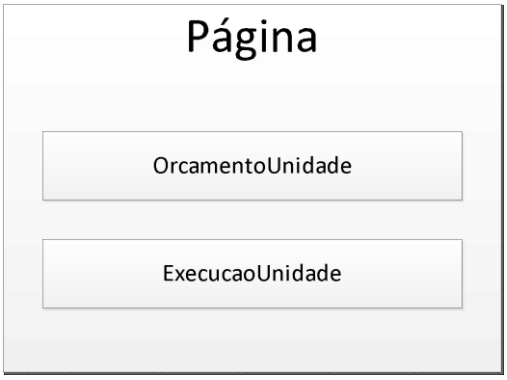


Figura 16: Estrutura de um objeto "Página" com os objetos "OrçamentoUnidade" e "ExecuçãoUnidade" embutidos

Esses objetos do tipo “OrçamentoUnidade” ou “ExecucaoUnidade” são gerados apenas quando a página em questão possui uma tabela com os dados e os mesmos não são links. No caso apesar do objeto “Página” suportar duas listas, ele terá apenas preenchida uma lista (Orçamento ou execução), pois um determinado link corresponde ao Orçamento ou a Execução e não aos dois ao mesmo tempo. Isso deve-se devido a organização dos portais. Na Figura 17 são mostrados trechos de código exemplificando a criação do objeto dos tipos citados acima.

```
public OrcamentoUnidade(String u, String i, String a, String v){
        this.unidade = u;
        this.inicial = i;
        this.atualizado = a;
        this.variacao = v;
    }

public ExecucaoUnidade(String u, String cc, String mo, String vo, String
eo, String ea, String ve, String vl, String vp){
        this.uni_fornecedor = u;
        this.cnpj_cpf = cc;
        this.modalidade = mo;
        this.valor_orcado = vo;
        this.emp_original = eo;
        this.emp_anulado = ea;
        this.valor_empenhado = ve;
        this.valor_liquidado = vl;
        this.valor_pago = vp;
    }
```

Figura 17: Estrutura dos objetos "OrcamentoUnidade" e "ExecucaoUnidade"

Durante a leitura, todos os links são lidos sequencialmente. Após ler um link, o software navega para a página desse link lido, se nessa página houver links também, os mesmos também serão lidos. Logo todos os links são lidos, de forma recursiva. Durante a leitura são lidos todos os links da página, inclusive os que não vão ser utilizados. Logo, o software durante a leitura seleciona os links usados dos não usados. Na leitura da página principal o software detecta os menus relacionados ao Orçamento e à Execução, ambos do menu principal Despesa. Em seguida é feita a leitura somente dos dados relacionados a essas páginas, ignorando os outros dados no site. Em uma página, os links que serão lidos são dispostos em forma de tabela. Os links pertencentes às mesmas linhas redirecionam para a mesma página, logo essa página destino seria lida mais de uma vez. Para evitar a leitura repetida, antes de realizar a leitura de uma página é feita uma verificação se a página em questão já foi lida por outro link da mesma linha. Caso tenha sido, a página não será lida novamente e o objeto "Página" gerado na leitura anterior será referenciado também nesse link. Logo, conclui-se que uma página pode ter como origem um ou mais links, uma relação de 1-N. Essa seleção dos dados a serem lidos torna o software mais eficiente, diminuindo o tempo de execução.
Após a leitura dos links e a consequente criação dos objetos do tipo "Página", "OrcamentoUnidade" e "ExecucaoUnidade", os dados são salvos em um banco de dados, onde cada objeto representa uma linha nas suas respectivas tabelas.

4.1.1.2. Tempo de Leitura

As páginas relacionadas ao orçamento tiveram um tempo de leitura dos dados médio de 20 segundos por ano. Já os dados relacionados à execução tiveram um tempo médio de leitura de 5 minutos por ano. Esse tempo é o de coleta de dados, análise e armazenamento no banco de dados. Esses dados gravados no banco de dados serão disponibilizados para o usuário final através da API. Já o acesso aos dados diretamente da base de dados tem um tempo de leitura inferior a 1 segundo em ambiente local, podendo variar dependendo do local onde a base de dados ficará hospedada.

A diferença de leitura das páginas teve uma variação conforme a quantidade de links presentes em cada página. As páginas relacionadas à execução como consequência da quantidade de dados tiveram um tempo bem superior de leitura comparada ao do orçamento, como mostrado acima. Porém como o usuário final irá utilizar somente os dados que estão gravados no banco de dados, o desempenho da API é satisfatório.

4.2. ANÁLISE E ARMAZENAMENTO

Os dados coletados passam por uma análise manual pelo software, onde, sendo selecionados apenas os dados necessários, que no caso serão os dados de Despesa, tanto do Orçamento como da Execução. Os outros dados não serão capturados. Estes também são organizados para posteriormente serem armazenados em um banco de dados.

Porém, durante essa leitura não são exatamente todos os links que são lidos, muitos são ignorados pelo Web Crawlers porque não vão retornar dados relevantes conforme os dados escolhidos para a leitura inicialmente (apenas dados de Despesa de todos os anos). Para isso é feito um filtro para que alguns links sejam ignorados, na Figura 18 tem um trecho de código do programa que ignora alguns tipos de links:

```
if(ps.getTextoUrl().equals("") || //ignora links com as palavras abaixo
                        "PÁGINA INICIAL".equals(ps.getTextoUrl()) ||
                          "RECURSOS HUMANOS".equals(ps.getTextoUrl()) ||
                          "Cargos e Salários".equals(ps.getTextoUrl()) ||
                          "ORÇAMENTO".equals(ps.getTextoUrl()) ||
                          "Receita".equals(ps.getTextoUrl()) ||
                          "Despesa".equals(ps.getTextoUrl()) ||
                          "EXECUÇÃO".equals(ps.getTextoUrl()) ||
                          "Fornecedores".equals(ps.getTextoUrl()) ||
                          "GLOSSÁRIO".equals(ps.getTextoUrl()) ||
                          "LOGIN".equals(ps.getTextoUrl())   ||
                          "3º Setor".equals(ps.getTextoUrl())
                       ){
```

Figura 18: Exemplo de código para filtragem de páginas

O link “Despesa” é ignorado pois esse é o link inicial do Crawler, se ele não fosse ignorado, o software iria fazer leituras repetidas nas mesmas páginas. Esses links correspondem ao menu principal da página, como pode ser visto na Figura 6.

4.2.1. Banco De Dados

Foi criado um banco de dados para armazenamento dos portais lidos. A estrutura criada é adaptada para dados de qualquer cidade, como pode ser visualizado na Figura 19.

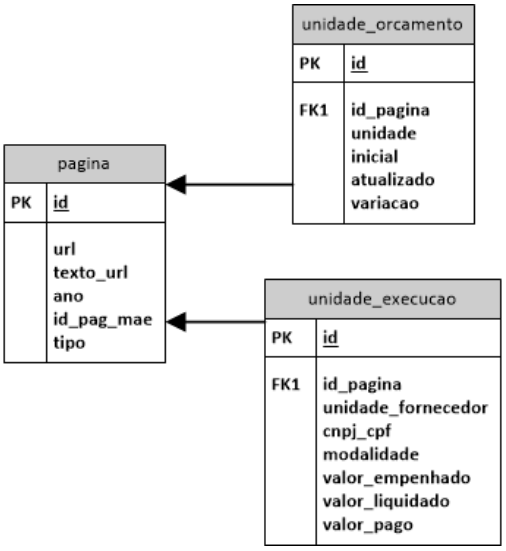


Figura 19: MER da estrutura do Banco de Dados

Foram criadas três tabelas, como descrito a seguir:

- Página: armazena todos os links lidos pelo Web Crawler. Ela possui os seguintes campos:
  - id: id do registro do link.
  - url: endereço do link lido, esse é o dado lido realmente pelo Crawler. Cada link lido e aceito pela análise é gerado um registro na tabela página.
  - texto_url: texto escrito no link, é o que aparece para o usuário quando ele visualiza o link.
  - ano: ano pertencente ao link, pois os dados nos portais são separados por ano.
  - id_pag_mae: id do seu link de origem, da página anterior ao link em questão. Esse id é um registro na mesma tabela. Para links da página inicial é atribuído zero, pois nesse caso nenhuma página o originou no momento da leitura.
  - tipo: caractere que especifica se o link pertence ao orçamento (O) ou à execução (E), ambos dados relacionados às despesas.
- Unidade_orcamento: armazena dados relativos ao orçamento que não estão no formato de link. Cada linha da tabela gera um registro na tabela. Ela possui os seguintes campos:
  - id: id do registro.
  - id_pagina: id do link (página anterior) que originou a página com os dados relativos ao orçamento que não estão em forma de link.
  - unidade: texto da primeira coluna da tabela, relacionado ao nome da unidade dos dados (nome de departamento, de fornecedor, etc.).
  - inicial: valor inicial gasto.

- atualizado: valor atualizado gasto.
- variacao: porcentagem de variação de gastos.

- Unidade_execucao: armazena dados relativos à execução que não são links. Assim como no orçamento, cada linha da tabela gera um registro no banco. Ela possui os seguintes campos:
  - id: id do registro
  - id_pagina: id do link (página anterior) que originou a página com os dados relativos à execução que não estão em forma de link.
  - unidade_fornecedor: texto da primeira coluna da tabela, nome da unidade ou do fornecedor relativo ao registro.
  - cnpj_cpf: CNPJ ou CPF para os dados que envolvem pessoa física ou jurídica.
  - modalidade: tipo de despesa.
  - valor_empenhado: valor empenhado da despesa.
  - valor_liquidado: valor liquidado da despesa.
  - valor_pago: valor pago da despesa.

As tabelas unidade_orcamento e unidade_execução são necessárias porque quando o software lê a última página, os dados na mesma não são links, porém os mesmos também são lidos e salvos pelo software.

4.3. DISPONBILIZAÇÃO

Através de uma API foram disponibilizados serviços que possibilitam a realização de diversas consultas à base de dados. Os dados são retornados em um formato aberto, para que possam ser reaproveitados em outras aplicações.

Os dados podem ser acessados através de diversas funções, listadas a seguir:

4.3.1. gastoTodosDepartamentos

Lista os dados financeiros de todos os departamentos da prefeitura de um determinado ano, seja referente ao orçamento ou a execução. O resultado da função é mostrado na Figura 20.

Parâmetros:
Ano: ano referente aos gastos desejados.

Tipo: caractere para indicar se os dados são do orçamento (o) ou da execução (e)


```
[
  {
    "orcado_inicial": "41.056.000,00",
    "variacao": "7,02",
    "departamento": "Departamento De Administração",
    "orcado_atualizado": "43.940.350,00"
  },
  {
    "orcado_inicial": "2.573.900,00",
    "variacao": "-2,34",
    "departamento": "Departamento De Bem Estar Social",
    "orcado_atualizado": "2.513.924,96"
  },
  {
    "orcado_inicial": "94.479.800,00",
    "variacao": "1,50",
    "departamento": "Departamento De Educação",
    "orcado_atualizado": "95.906.378,01"
  },
  {
    "orcado_inicial": "5.296.000,00",
    "variacao": "2,00",
    "departamento": "Departamento De Finanças",
    "orcado_atualizado": "5.402.000,00"
  },
  {
    "orcado_inicial": "20.936.000,00",
    "variacao": "1,31",
    "departamento": "Departamento De Obras E Serviços",
    "orcado_atualizado": "21.211.837,46"
  },
  {
    "orcado_inicial": "10.410.000,00",
    "variacao": "5,59",
    "departamento": "Departamento De Planejamento E Meio Ambiente",
    "orcado_atualizado": "10.992.800,00"
  },
  {
    "orcado_inicial": "3.085.000,00",
    "variacao": "-40,16",
    "departamento": "Departamento De Turismo Des. Econômico Cultura Esporte E Lazer",
    "orcado_atualizado": "1.846.150,00"
  },
  {
```


Figura 20: Exemplo de resultado da função "gastoTodosDepartamentos"

4.3.2. gastoDepartamento

Tem funcionalidade parecida da função “gastoTodosDepartamentos”, só que ela permite escolher qual o departamento que se deseja obter os dados, seja referente ao orçamento ou à execução.

Parâmetros:

Ano: ano referente aos gastos desejados.

Tipo: caractere para indicar se os dados são do orçamento (o) ou da execução (e) Departamento: texto do departamento desejado (ex: ‘Educação’). A função aceita que seja digitado parte do texto do departamento que o sistema irá procurar um departamento com o nome mais próximo do indicado. O resultado da função é mostrado na Figura 21.

```
[
  {
    "orcado_inicial": "94.479.800,00",
    "variacao": "1,50",
    "departamento": "Departamento De Educação",
    "orcado_atualizado": "95.906.378,01"
  }
]
```


Figura 21: Exemplo de resultado da função "gastoDepartamento”

4.3.3. gastoDepartamentoDetalhe

Lista os detalhes de um departamento de um determinado ano, seja do orçamento ou da execução. Esse detalhe são os dados referentes a página que direciona o link correspondente ao departamento. Por exemplo, o departamento de Educação, no site ele é um link que redireciona para outra página onde são mostrados os departamentos que são dependentes do anterior (ex: Educação Básica, Educação Superior, etc.). Na Figura 22 é mostrado um exemplo de retorno da função.

Parâmetros:

Ano: ano referente aos gastos desejados.

Tipo: caractere para indicar se os dados são do orçamento (o) ou da execução (e)

Departamento: texto do departamento desejado (ex: ‘Educação’). A função aceita que seja digitado parte do texto do departamento que o sistema irá procurar um departamento com o nome mais próximo do indicado.


```
[
  {
    "orcado_inicial": "13.113.500,00",
    "unidade": "Ensino Básico - Fundamental",
    "orcado_atualizado": "15.738.500,00",
    "orcado_variacao": "20,01"
  },
  {
    "orcado_inicial": "10.977.000,00",
    "unidade": "Ensino Básico - Infantil",
    "orcado_atualizado": "10.212.000,00",
    "orcado_variacao": "-6,97"
  },
  {
    "orcado_inicial": "1.605.000,00",
    "unidade": "Ensino Médio",
    "orcado_atualizado": "1.364.000,00",
    "orcado_variacao": "-15,02"
  },
```


Figura 22: Exemplo de resultado da função "gastoDepartamentoDetalhe"

4.3.4. gastoTodasUnidadeOrcamento

Lista os dados resumidos de todas unidades referentes ao orçamento, já que a página inicial do orçamento os dados são agrupados de duas formas: por unidade, por departamento. A Figura 23 mostra um exemplo de saída da função.

Parâmetros:

Ano: ano referente aos gastos desejados.


```
[
  {
    "orcado_inicial": "5.629,50",
    "unidade": "Controladoria E Contabilidade Geral",
    "orcado_atualizado": "466.413,46",
    "orcado_variacao": "8.185,16"
  },
  {
    "orcado_inicial": "1.246,00",
    "unidade": "Coordenadoria Municipal De Defesa Civil",
    "orcado_atualizado": "109.907,14",
    "orcado_variacao": "8.720,79"
  },
  {
    "orcado_inicial": "1.810,00",
    "unidade": "Corpo De Bombeiros Municipal",
    "orcado_atualizado": "154.065,98",
    "orcado_variacao": "8.411,93"
  },
```


Figura 23: Exemplo de resultado da função "gastoTodasUnidadeOrcamento"

4.3.5. gastoUnidadeOrcamento:

Lista os dados resumidos de uma unidade referente ao orçamento, já que a página inicial do orçamento os dados são agrupados de duas formas: por unidade, por departamento.

Parâmetros:

Ano: ano referente aos gastos desejados.

Unidade: Texto da unidade (ex: “Fundo Municipal de Saúde”). Caso o usuário não digite todo o nome da unidade, o sistema buscará uma com o nome mais próximo possível do que foi informado.

4.3.6. gastoFornecedorExecucao:

Lista os fornecedores relacionados à execução de despesas de um determinado departamento. A Figura 24 mostra um exemplo de saída da função.

Parâmetros:
Ano: ano referente aos gastos desejados.
Departamento: Texto do departamento. Caso o usuário não digite todo o nome do departamento, o sistema buscará uma com o nome mais próximo possível do que foi informado.


```
[
  {
    "fornecedor": "A. M. Cerezer Ltda - Epp",
    "cnpj_cpf": "01.917.621/0001-05"
  },
  {
    "fornecedor": "Antonio Tebom Cia Ltda - Epp",
    "cnpj_cpf": "49.388.655/0001-78"
  },
  {
    "fornecedor": "Associacao Com Ind De Capivari",
    "cnpj_cpf": "46.927.638/0001-73"
  },
  {
```


Figura 24: Exemplo de resultado da função "gastoFornecedorExecucao"

4.3.7. departamentoGastoFornecedor

Mostra os departamentos pertencentes a um fornecedor. Dados relacionados à execução. Um exemplo de retorno da função é exibido na Figura 25.

Parâmetros:
Nome do fornecedor: nome do fornecedor que se deseja ver os departamentos.

```
[
  {
    "departamento": "Por Fornecedor Mensal"
  },
  {
    "departamento": "Cultura Turismo E Dependências"
  },
  {
    "departamento": "Educação Infantil"
  },
```


Figura 25: Exemplo de resultado da função "departamentoGastoFornecedor"

4.3.8. gastoExecucaoTodosMeses

Mostra os dados de gastos relativos à execução separados por mês de um determinado ano. Um exemplo de resultado da função pode ser visto na Figura 26.

Parâmetros:

Ano: ano que se deseja saber os dados separados por mês.


```
[
  {
    "valor_pago": "10.421.549,73",
    "mes": "01 - Janeiro",
    "valor_empenhado": "57.272.593,57",
    "valor_liquidado": "11.499.615,57"
  },
  {
    "valor_pago": "16.976.257,71",
    "mes": "02 - Fevereiro",
    "valor_empenhado": "20.500.052,31",
    "valor_liquidado": "17.083.762,39"
  },
  {
    "valor_pago": "19.588.562,66",
    "mes": "03 - Março",
    "valor_empenhado": "18.675.429,52",
    "valor_liquidado": "21.190.325,20"
  },
```


Figura 26: Exemplo de resultado da função "gastoExecucaoTodosMeses"

4.3.9. gastoExecucaoMes:

Mostra os dados gastos de um determinado mês de um ano. Dados relacionados à execução.

Parâmetros:

Ano: ano que se deseja saber os dados.

Mês: mês que se deseja buscar os dados.

4.3.10. gastoDepExecuçãoMes

Mostra os gastos relativos à execução separados por departamento, porém esses gastos são de apenas um determinado mês de um ano.

Parâmetros:
Ano: ano que se deseja saber os dados.
Mês: mês que se deseja buscar os dados.

4.3.11. gastoDepDetalheExecucaoMes:

Mostra os detalhes de um determinado departamento de um mês. Dados relacionados à execução. A Figura 27 mostra um exemplo de resultado da função.

Parâmetros:
Ano: ano que se deseja saber os dados.
Mês: mês que se deseja buscar os dados.
Nome do departamento: nome do departamento que se deseja buscar os dados (ex: execução).


```
[
  {
    "orcado_inicial": "1.302.944,48",
    "variacao": "1.044.012,83",
    "departamento": "Ensino Básico - Fundamental",
    "orcado_atualizado": "852.419,97"
  },
  {
    "orcado_inicial": "678.273,22",
    "variacao": "837.855,17",
    "departamento": "Ensino Básico - Infantil",
    "orcado_atualizado": "834.449,73"
  },
  {
    "orcado_inicial": "4.067,21",
    "variacao": "26.041,14",
    "departamento": "Ensino Médio",
    "orcado_atualizado": "26.041,14"
  },
```


Figura 27: Exemplo de resultado da função "gastoDepDetalheExecucaoMes"

4.3.12. Como usar a API DAG Prefeituras

Ao utilizar-se da API DAG Prefeituras (Figura 28), todas as funções acima estarão disponíveis para o usuário, basta fazer a chamada e passar os parâmetros corretos que a API irá acessar o banco e retornar os dados desejados. Esses dados são retornados em formato JSON, que é um formato aceito conforme os princípios dos dados abertos governamentais (OPEN, 2016) e de possível utilização em outras aplicações. A ideia principal da API é que seu usuário não se preocupe com a captura dos dados e sim com sua posterior utilização.

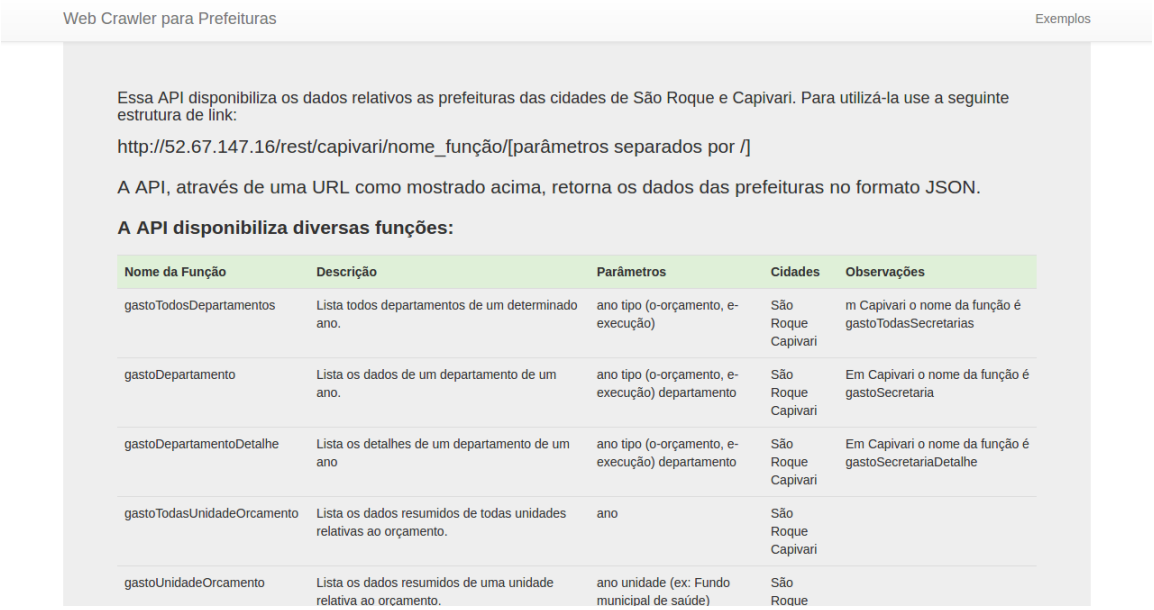


Figura 28: Página inicial da API DAG Prefeituras

Na Figura 29 é mostrado um exemplo de como se utilizar as funções da API de forma local. Como pode-se notar, basta criar o objeto e em seguida chamar a função que a API faz a busca e retorna os dados no formato JSON.

```
PaginaSaoRoque ps = new PaginaSaoRoque();
JSONArray ja6 = ps.gastoTodosDepartamentos(2016,"e");
ps.imprimeJson(ja6);
```

Figura 29: Como usar a API DAG Prefeituras localmente

4.3.13. Uso de Web Services

O acesso aos dados é feito com uma API que utiliza um Web Service. Foi utilizado um Web Service RESTFUL utilizando a linguagem JAVA.

O Web Service retorna dados do tipo JSON, que podem ser consumidos em diversas linguagens de programação, não necessariamente JAVA, já que o formato de dados JSON e as requisições HTTP são de certa forma, mais “universais”.

Na Figura 30 é mostrado um exemplo de acesso ao Web Service utilizando apenas Java Script através do Framework JQuery. A vantagem dessa abordagem é facilidade no desenvolvimento, já que o desenvolvedor precisa apenas do Framework JQuery, que pode ser chamado através de um link online.

```
$(document).ready(function() {
   $.ajax({
         type: "GET",
         url: "http://localhost:8085/WebServiceRest/rest/capivari/
gastoTodasSecretarias/2014/o",
         contentType: "application/json; charset=utf-8",
         dataType: "json",
      crossDomain: true,
         cache:false,
         async:false,
         success: function (data) {
          $.each(data, function(i, item) {
             $.each(item, function(j, objeto){
                 $(document.body).append(objeto+'<br>');
             });
             $(document.body).append('<br>');
          });
          },
          error: function (xhr, status, error) {
        $(document.body).append(xhr.responseText+ " - " + status );
          }
   });
});
```

Figura 30: Exemplo de chamada da API DAG Prefeituras com Jquery via ajax

Outro exemplo de acesso ao Web Service é através da linguagem PHP (Figura 31). Neste caso o desenvolvedor precisa ter um servidor apache e o PHP instalado em sua máquina para conseguir utilizar o código. Mas isso são requisitos da linguagem PHP e não do Web Service.

```php
<?php
        $response=utf8_encode(file_get_contents('http://localhost:8085/
WebServiceRest/rest/capivari/gastoTodasSecretarias/2014/o'));
        $dados = json_decode($response);
        echo "<b>Secretarias</b><br><br>";

        foreach($dados as $d){
        echo $d->orcado_inicial."<br>";
        echo $d->variacao."<br>";
        echo $d->orcado_atualizado."<br>";
        echo $d->secretaria."<br>";
        echo "<br>";
        }
?>
```

Figura 31: Exemplo de chamada da API DAG Prefeituras via PHP

Independentemente da linguagem utilizada para acesso aos dados da API utilizando o Web Service, o retorno dos dados é no formado JSON, que pode ser facilmente aproveitado em diversas linguagens de programação de forma facilitada. O código na Figura 32 mostra o retorno de uma função da API, que retorna os dados no formato JSON.


```json
[
 {
   "valor_pago": "10.421.549,73",
   "mes": "01 - Janeiro",
   "valor_empenhado": "57.272.593,57",
   "valor_liquidado": "11.499.615,57"
 },
 {
   "valor_pago": "16.976.257,71",
   "mes": "02 - Fevereiro",
   "valor_empenhado": "20.500.052,31",
   "valor_liquidado": "17.083.762,39"
 },
 {
   "valor_pago": "19.588.562,66",
   "mes": "03 - Março",
   "valor_empenhado": "18.675.429,52",
   "valor_liquidado": "21.190.325,20"
 },
 {
   "valor_pago": "18.402.842,91",
   "mes": "04 - Abril",
   "valor_empenhado": "18.263.011,55",
   "valor_liquidado": "18.220.325,38"
 }
```


Figura 32: Exemplo de retorno de dados da API DAG Prefeituras

4.4. CONSIDERACOES FINAIS

Para a captura dos dados das prefeituras escolhidas foi criado um Web Crawler de propósito geral utilizando a linguagem Java. Esse software foi responsável pela captura dos dados, seleção e armazenamento dos mesmos em um banco de dados.

A disponibilização dos dados foi feita através de um API, também desenvolvida na linguagem JAVA, denominada de API DAG Prefeituras. Essa API utilizar um Web Service para acessar o banco de dados. O acesso é através de diversas funções, que já disponibiliza os dados utilizando diversos filtros. Os dados são retornados no formato JSON, um formato aceito conforme os princípios dos DAG.

Apesar da API ser feita em JAVA, é possível chamá-la em diversas linguagens de programação, através de uma URL, ampliando as possibilidades de uso da API DAG Prefeituras em aplicações desenvolvidas utilizando diferentes linguagens.

A API possui capacidade de expansão, porém é preciso que o Web Crawler seja adaptado para ler dados provenientes de outro site. Para isso é preciso ser feita uma análise da estrutura do site para então o Web Crawler ser atualizado, onde é preciso criar outra classe, correspondente a nova cidade. Porém essa classe herda diversas características gerais do software do Web Crawler. Lembrando que o software captura dados em HTML, logo o site precisa disponibilizar os dados neste formato.

Essa necessidade de alteração do Web Crawler para adicionar mais cidades deve-se a estrutura dos portais da transparência das prefeituras brasileiras, que seguem diferentes estruturas, sem um padrão.

5. AVALIAÇÃO DA PROPOSTA

Neste capítulo é mostrada a avaliação da proposta realizada neste trabalho, onde são utilizados princípios de experimentação de software de Travassos (TRAVASSOS, GUROV e AMARAL, 2002), Brasili (BASILI, SELBY e HUTCHENS, 1986) e alguns conceitos de Davis (DAVIS, 1989). O foco dessa avaliação, é verificar a combinação desses princípios em dois grupos, com níveis diferentes de experiência em programação e desenvolvimento.

5.1. Experimentação de Software segundo Travassos

Em uma experimentação é preciso atentar alguns elementos, descritos a seguir (TRAVASSOS, GUROV e AMARAL, 2002):

- ***variáveis:*** podem ser classificadas em dependentes e independentes, relacionadas as entradas e saídas do experimento, respectivamente.
- ***objetos:*** são eles que recebem os tratamentos durante o experimento para, no final, gerar o resultado que será avaliado.
- ***participantes:*** indivíduos selecionados para participar do experimento. Eles são uma amostra a partir de uma população que tenha as características adequadas para o experimento em questão.
- ***contexto:*** condições onde o experimento vai ser executado. Essas condições podem ser classificadas em:
    - in-vitro vs. In-vivo: o primeiro em laboratório e o segundo no mundo real.
    - alunos vs. profissionais
    - problemas de sala de aula vs. problemas reais
    - específico vs. geral: resultados voltados para um contexto particular ou para todo o contexto da Engenharia de Software.
- **hipóteses:** faz parte do experimento, é ela que será provada, para verificar se a mesma é verdadeira ou falsa.

Um experimento precisa ser organizado e para isso é preciso seguir alguns princípios, como aleatoriedade, agrupamento (blocking) e balanceamento. Outro princípio utilizado também é a repetição, que busca através da repetição do experimento verificar se não houve resultados equivocados no experimento.

Outra característica importante em um experimento é a **medição**, que consiste em "mapeamento do mundo experimental para o mundo formal". Consiste na manipulação dos dados de uma forma mais “formal” e precisa. Para isso são utilizados números ou símbolos, desde que seja possível medir, aplicar uma métrica.

Essa medição é vista em duas dimensões, a objetivo e subjetiva e direta e indireta. Para a medição ser objetiva, ela precisa depender somente do objeto. Caso ela dependa do objeto e da perspectiva do mesmo, ela é subjetiva. Neste caso, ao medir o objeto diversas vezes, o valor pode ser diferente. Já no caso de uma medida direta é “aquela que não envolva a medição de outros atributos”, e no caso da medida indireta, é “derivada de outros atributos”.

Em engenharia de software o próprio programa é usado as próprias métricas do programa como equipe, hardware, ferramentas, entre outros recursos necessários para seu desenvolvimento. Um software passa por uma medição devido a vários motivos que visa compreender melhor seu funcionamento, além de identificar possíveis correções e melhorias que ele poderia ter.

Outro item importante em experimentação é a validade. É preciso comprovar o “quanto é válido os resultados de um experimento na população onde ele foi aplicado”. Existem quatro tipos de validade, descritos a seguir:

- Validade de conclusão: relacionada a conclusão correta do experimento a partir dos resultados. Ela leva em consideração testes estatísticos, participantes, medidas e confiabilidade dos tratamentos.
- Validade interna: verifica se o resultado não sofreu influência de outro fator que não foi controlado ou medido. Esse tipo de validade deve-se observar com atenção os participantes.
- Validade de construção: “considera os relacionamentos entre a teoria e a observação”. Neste caso, deve-se observar fatores humanos.
- Validade externa: busca limitações para generalizar os resultados do experimento para a prática industrial. Deve ser considerado os relacionamentos entre os participantes, local e o tempo.

5.1.1. Tipos de Experimentos

Na experimentação de software é possível aplicar diversos tipos de experimentos. Os mais comuns são: Survey, Estudo de caso e Experimento.

O Survey “é uma investigação executada em retrospectiva”, ele é feito quando “algumas técnicas ou ferramentas já foram utilizadas”. Seus objetivos são descritivos, explanatório e explorativo. Uma característica dele á a capacidade de levantar um grande número de variáveis.

Já o estudo de caso é utilizado para “monitorar projetos, atividades e atribuições”. Ele também busca a comparação de resultados, dentro do projeto e também através do uso de projetos semelhantes. Ele possui um controle de medição das variáveis, porém ainda sim o nível de controle geral é baixo devido a fatores de confusão, que podem modificar os resultados.

O experimento em si por sua vez é aquele “realizado em laboratório”, que devido a isso, oferece um maior nível de controle, pois é feito em um ambiente isolado e controlado.

Travassos (TRAVASSOS, GUROV e AMARAL, 2002) criou uma tabela onde é possível comprar esses três tipos de experimentação levando em relação itens como grau de controle e repetição. Essa tabela é mostrada na Figura 33.

Fator	*Survey*	Estudo de caso	Experimento
O controle da execução	Nenhum	Nenhum	Tem
O controle da medição	Nenhum	Tem	Tem
O controle da investigação	Baixo	Médio	Alto
Facilidade da repetição	Alta	Baixa	Alta
Custo	Baixo	Médio	Alto

Figura 33: Comparação dos tipos de experimentação por Travassos

5.1.2. Coleta de Dados

Em um processo de experimentação é preciso atentar-se as formas de coleta de dados. Existem três métodos principais para este fim:

- ***Histórico:*** coleta dados de projetos já finalizados. Utiliza pesquisa bibliográfica, lições aprendidas e análise estatística.
- ***Observação:*** a coleta dos dados é feita durante a execução do projeto. Utiliza monitoramento do projeto, estudo de caso, afirmação, estudo de campo.
- ***Controlado:*** provem validade estatística dos resultados coletados. Utiliza repetição, sintético, análise dinâmica, simulação.

Já para a classificação desses dados coletados, são utilizadas as seguintes categorias: Estudo qualitativo, Estudo quantitativo e Benchmarking (utilizado para medição de produtos de software).

5.1.3. Processo do Experimento

Um experimento é realizado, dentre outras coisas, para comprovar uma teoria. Para que isso fosse possível foram desenvolvidas diversas metodologias para esse fim.

Um exemplo de metodologia é conhecido como Paradigma da Melhoria da Qualidade (Quality Improvement Paradigm – QIP), baseada na melhoria contínua e focada em sete fases principais: Caracterizar, estabelecer objetivos, escolher o processo, Executar, Empacotar e Analisar.

O GQM (Goal/QuestionqMetric), é outra abordagem, ligada a QIP, em que a medição é baseada em camadas. Seus objetivos principais são: compreender, controlar e melhorar, focados em quatro fatores: custo, risco, tempo, qualidade. Essa abordagem é composta por quatro fases:

- ***Planejamento:*** o projeto é selecionado, definido, medido, caracterizado e planejado.
- ***Definição:*** são estabelecidas as questões, métricas e hipóteses.
- ***Coleta de dados:*** é feia a coleta durante os experimentos.
- ***Interpretação:*** feita uma análise, utilizando as métricas, questões e objetivos definidos inicialmente.

5.2. Experimentação de Software segundo Brasili

Brasili (BASILI, SELBY e HUTCHENS, 1986) estabeleceu um framework para auxílio em experimentação em software. O mesmo consiste em quatro etapas, definição, planejamento, operação e interpretação.

Na primeira etapa, definição são estabelecidas diretrizes de como será feito esse estudo. Primeiramente deve-se entender as motivações do mesmo, que envolve questões de avaliação, gerenciamento, melhorias, aprendizado, validações, entre outras. Em seguida, deve-se ser definido qual o objeto do estudo, que pode ser um produto, um processo, uma teoria, entre outros, como também os propósitos deste estudo, como caracterização do objeto de estudo, evolução do mesmo, etc. Deve-se também durante a definição do estudo considerar as perspectivas envolvidas no estudo, sejam elas relacionadas ao objeto de estudo ou as pessoas

envolvidas. E não mais importante, deve-se definir de forma bem precisa o escopo do estudo, considerando o projeto em si e outros projetos envolvidos.

Para o planejamento é necessário considerar três fatores: design, critérios e medições. No design pode ser utilizado design experimental, bloqueio de randomização, análise multivariável, modelos estáticos, amostragem, e também sem a utilização de parametrização. Para os critérios do estudo são considerados diversos fatores como: custo, qualidade, erros, mudanças, reabilitação, correção, compreensão do programador, cobertura de execução, tamanho e complexidade. E como métricas é preciso primeiramente definir quais métricas serão utilizadas, como métricas de validação, coleção de dados, formulários de design e teste, níveis de medição (nominal, classificatório), intervalo, proporção etc.

Na operação é preciso inicialmente preparar o estudo piloto e em seguida executar o mesmo, através da coleta e validação de dados. Posteriormente, os dados coletados precisam passar por uma análise quantitativa e qualitativa.

E por fim, na fase de interpretação, é estabelecido o contexto de interpretação, verificando propostas de estudo e campos para pesquisa e em seguida vendo seus impactos e aplicações.

5.3. Utilidade vs. Facilidade de Uso

Segundo Davis (DAVIS, 1989) vários motivos podem levar pessoas a rejeitarem ou aceitarem uma tecnologia. Algumas pessoas preferem uma aplicação que melhore sua performance no seu trabalho, ou seja, uma aplicação com **utilidade**. Outras preferem aplicações que tenham utilidade também, mas ao mesmo tempo acreditam que sistemas precisam ser **fáceis de se utilizar**.

Logo, tem-se duas perspectivas, “Utilidade” e “facilidade de uso”. A primeira é definida como “um degrau que a pessoa acredita estar usando um sistema que aumente sua performance no trabalho”. Já a segunda é definida “como o degrau que a pessoa acredita utilizar um sistema onde ela esteja livre de esforço”.

5.4. Considerações

Na avaliação desse trabalho será feita uma combinação dos conteúdos apresentados, utilizando parte dos princípios de experimentação de software apresentados por Travassos (TRAVASSOS, GUROV e AMARAL, 2002), juntamente com algumas ideias de Davis

(DAVIS, 1989), verificando a utilidade e facilidade de uso e também alguns conceitos de experimentação de software de Brasilli (BASILI, SELBY e HUTCHENS, 1986).

De forma geral, a avaliação da proposta foi feita através de um estudo de caso, onde foi dividido em três etapas principais: planejamento, execução e análise, além de uma validação preliminar inicial.

5.5. VALIDAÇÃO PRELIMINAR

Visando realizar uma análise preliminar dos serviços disponibilizados pela API um aluno do IFSP de Boituva realizou o desenvolvimento de uma aplicação Web que consumia informações disponibilizadas pela função “gastoTodosDepartamentos” (Figura 34). No caso foi utilizada a função para exibir os dados dos anos de 2012 a 2016. Com esses dados foram gerados a gráficos histórico do uso do recurso público separado por departamento, logo tem-se uma ideia do gasto de cada departamento ao longo dos anos. Nessa implementação, feita pelo aluno, os dados foram mostrados na legenda do gráfico. Uma forma de melhorar a exibição seria a conversão dos dados brutos em porcentagens.

Em contato com o aluno, a análise preliminar mostrou que o uso da API auxiliou na velocidade de desenvolvimento e no número de linhas de código do software final. O resultado mostrou também que a API facilita as análises estatísticas de dados abertos governamentais, pois a aplicação já recebe os dados prontos, tirando a necessidade de consultar o portal para capturar os mesmos. A API disponibiliza os dados em funções que facilitem inclusive sua reutilização. No caso da análise feita, a API fornece os dados prontos dos departamentos e gastos respectivos, tirando a necessidade que o usuário precise analisar os dados para obter as informações necessárias.

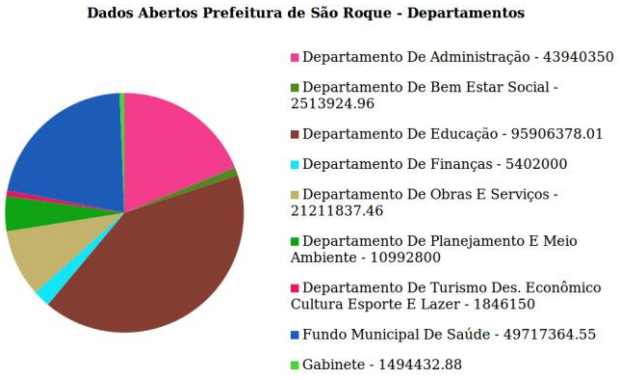


Figura 34: Exemplo de Gráfico com os dados da API DAG Prefeituras

Os dados são disponibilizados em diversas formas através das funções da API, agrupando os mesmos por departamento, fornecedor, ano, mês, tipo, etc., facilitando sua reutilização.

Entretanto existe a necessidade de realizar uma validação formal do uso da abordagem proposta. Neste sentido, foi realizado um estudo de caso durante o primeiro semestre de 2017, visando comparar a utilização da API com programadores com mais experiência, seja da pós-graduação ou do mercado e programadores com pouca experiência, ou seja, alunos da graduação. Nesse estudo foi avaliado a aceitação da API pelos desenvolvedores utilizando o TAM.

5.6. ESTUDO DE CASO

Durante o primeiro semestre de 2017 foi realizado um estudo experimental do tipo estudo de caso com a API, onde durante um dia os participantes foram separados em dois grupos, um composto por programadores com pouca experiência e outro com profissionais experientes do mercado. Esse estudo verificou o comportamento da plataforma nos dois grupos validados, considerando os princípios de experimentação de software de Travassos (TRAVASSOS, GUROV e AMARAL, 2002), Brasili (BASILI, SELBY e HUTCHENS, 1986) e os conceitos de Davis (DAVIS, 1989).

5.7. PLANEJAMENTO

O estudo foi planejado e executado no primeiro semestre de 2017, após o início do semestre letivo, para que pudessem ser selecionados alunos da graduação como parte dos participantes do estudo. A avaliação consistiu de um questionário inicial, da proposta de desenvolvimento e de um questionário final, aplicados para dois grupos de usuários.

5.7.1. Participantes

Para a realização do estudo foram selecionados alunos da UFSCAR do curso de Graduação em Ciência da Computação e profissionais do mercado. Como o estudo envolveu programadores com diferentes níveis de experiência, foi selecionado alunos do curso de graduação como programadores com pouca experiência e profissionais do mercado como programadores com muita experiência.

Os participantes do grupo de alunos da USCAR foram divididos em três grupos, denominados GA1, GA2 e GA3, cada um com dois alunos. Já os profissionais do mercado

também foram divididos em grupos, denominados GP1, GP2 e GP3, porém cada um trabalhou sozinho.

5.7.2. Projeto a ser validado

Para os participantes foi proposto um desafio de programação onde foi desenvolvido um software utilizando os dados providos pela API. O software utilizou as funções da API e exibiu os resultados em forma de gráficos. Para o desafio pode-se utilizar qualquer linguagem de programação compatível com o tipo JSON, mas durante o estudo foi mostrado como se utiliza a API em três linguagens: JAVA, Java Script e PHP.

A Proposta do Desafio foi criar uma aplicação que mostrasse em forma de gráfico os gastos das duas prefeituras da API, Capivari e São Roque. O gráfico teria que mostrar os gastos no período de 2011 até 2016 de dois setores de cada prefeitura (ex: Saúde, Educação, Administração, etc.).

5.7.3. Objetivo do estudo

Analisar as competências da API

Com propósito de avaliação

Com respeito a utilidade e facilidade de uso

Do ponto de vista de desenvolvedores de software com diferentes níveis de experiência

No contexto de estudantes do curso de Ciência da Computação e profissionais e profissionais do mercado.

5.7.4. Questões e métricas

As questões e métricas que se investiga são:

Q1: A utilização da API facilita a execução das tarefas propostas

Métrica: Grau de utilidade da API DAG Prefeituras.

Q2: Qual o nível de dificuldade da utilização da API levando em consideração aos conhecimentos prévios em programação.

Métricas: Grau de Facilidade de Uso da API no desenvolvimento da atividade proposta. Nível de conhecimento em programação e tecnologias dos participantes.

5.7.5. Objetivos da medição

Os objetos da medição da API são os seguintes:

- Quais os ganhos da utilização da API no desenvolvimento proposto utilizando DAG.
- Quais são as dificuldades enfrentadas durante a execução do desenvolvimento da aplicação pelos participantes dos dois grupos.
- Que outros recursos a API poderiam disponibilizar.

5.7.6. Variáveis

Variáveis dependentes: facilidade de uso e utilidade do projeto

Variáveis independentes: metodologia, tecnologias, linguagens de programação, ambiente.

Projeto desenvolvido: integração e visualização de DAG.

Tecnologias utilizadas: JAVA, Java Script, JQuery, HTML, CSS, JSP, Web Services.

Ambientes: Laboratórios de informática da UFSCAR, computadores pessoais dos participantes.

5.8. EXECUÇÃO

Com os participantes foi proposto um desafio de programação utilizando a API. Esse desafio consistiu basicamente de quatro etapas: aplicação de um questionário inicial, aula sobre DAG, desafio de programação usando a API DAG Prefeituras e aplicação de um questionário final. (APENDICE – B)

Todos os participantes do estudo, independente do grupo realizaram todas essas etapas, só que em momentos diferentes. Os participantes dos grupos GA1, GA2 e GA3 (alunos da graduação) realizaram o experimento durante um dia, nas dependências da UFSCAR. Os participantes dos grupos GP1, GP2 e GP3 (profissionais do mercado) fizeram o estudo também em um dia, mas em momentos distintos.

A primeira etapa foi a aplicação de um questionário inicial com o objetivo de traçar o perfil dos participantes e verificar a experiência e conhecimento dos mesmos em itens como: habilidade em algoritmos e programação, conhecimento em linguagens de programação, conhecimentos em DAG, etc.

Após foi dada uma aula sobre DAG. Durante a aula foi explicado conceitos como Dados abertos Governamentais, formatos aceitos para DAG, Web Crawler, Web Services,

JSON, além de explicar como funciona a API proposta no estudo. Posteriormente foi proposto um desafio de programação utilizando as funções da API. Esse desafio teve duração de um dia.

A API DAG Prefeituras foi disponibilizada para os participantes do estudo, onde eles tiveram que criar um programa para fazer a captura dos dados utilizando essa API. A API por sua vez fará a comunicação com a base de dados.

O desafio consistia, usando a API DAG Prefeituras criar um gráfico que mostrasse os gastos de dois departamentos do período de 2011 a 2016, de ambas prefeituras, de São Roque e Capivari. O objetivo do desafio é mostrar de forma visual a comparação de gastos entre departamentos de uma prefeitura e entre elas. Na Figura 35 é mostrada uma forma de implementação do desafio.

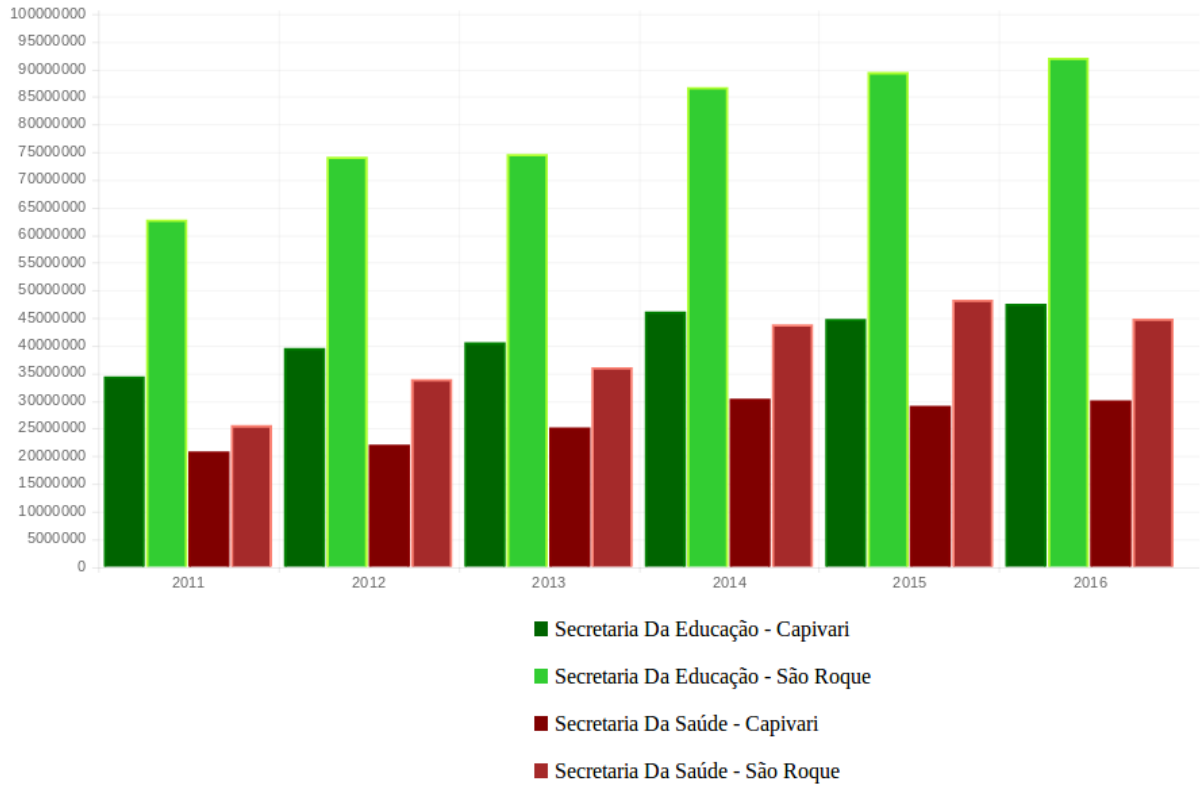


Figura 35: Exemplo de aplicação usando a API DAG Prefeituras

No desafio, além da utilização da API DAG Prefeituras para a captura dos dados, os participantes usaram outras APIs para a criação dos gráficos, variando conforme a solução de cada grupo.

Durante o estudo realizado com os grupos GA, (alunos da graduação) foi feito um acompanhamento presencial e as possíveis dúvidas dos participantes foram retiradas. Já os participantes dos grupos GP, o acompanhamento foi feito de forma presencial e a distância, conforme a necessidade.

Ao final do desafio, todos os participantes responderam outro questionário. Este por sua vez detecta aspectos importantes durante o desenvolvimento, como tempo de programação, número de linhas de código, dificuldades durante o desenvolvimento, tecnologias utilizadas, sugestões e também a utilidade e facilidade de uso da API DAG Prefeituras no contexto de DAG (BRITO, COSTA, *et al.*, 2014) e TAM (BASILI, SELBY e HUTCHENS, 1986). A Figura 36 mostra um exemplo de aplicação criada pelos participantes, no caso utilizando PHP, assim como na Figura 37, só que usando JQuery (Framework da linguagem Java Script) para construir a solução. A Figura 38 mostra uma aplicação feita em Android, mostrando a solução proposta por outro participante.

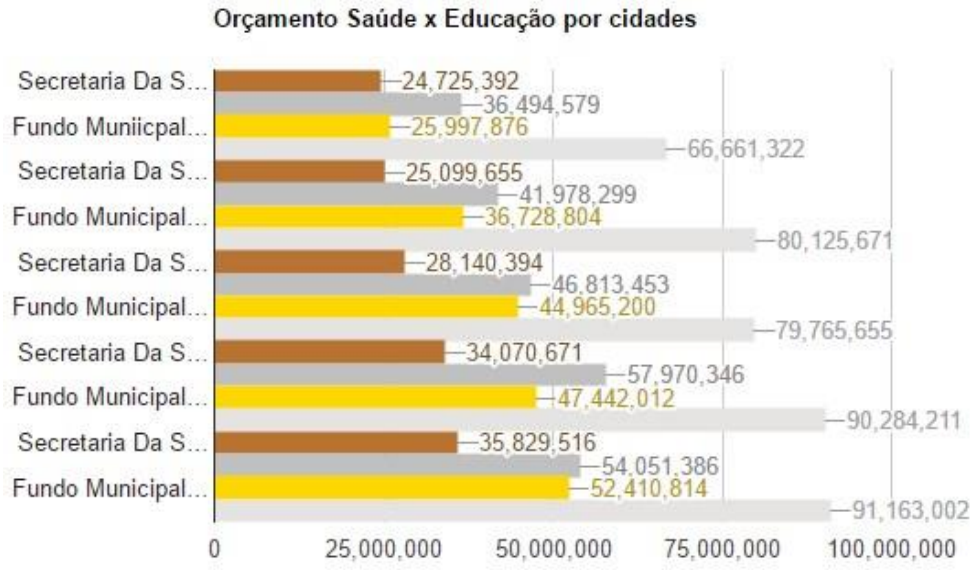


Figura 36: Exemplo de Solução utilizando PHP

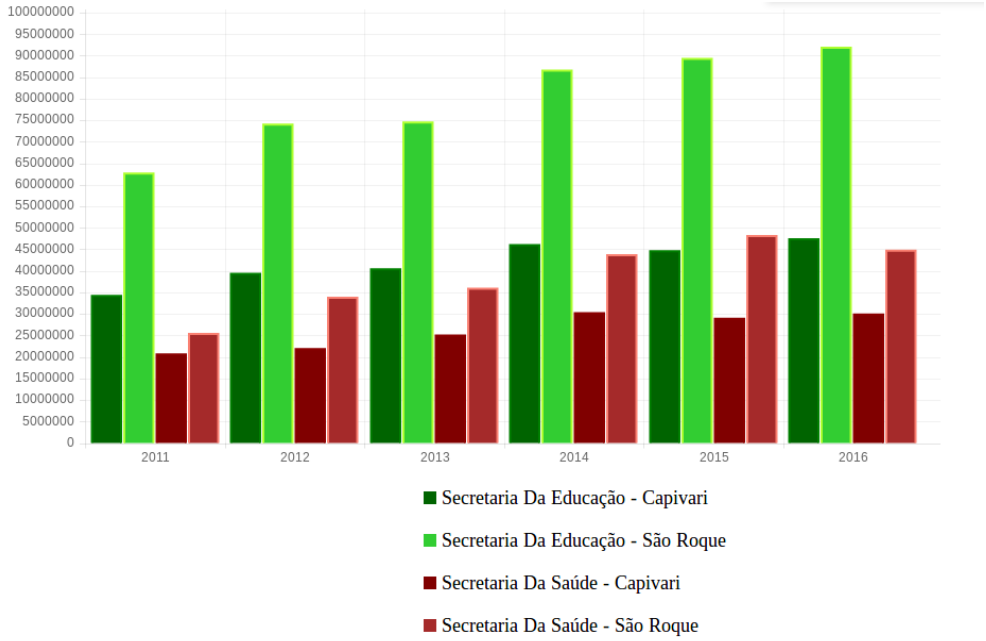


Figura 37: Exemplo de Solução utilizando JQuery

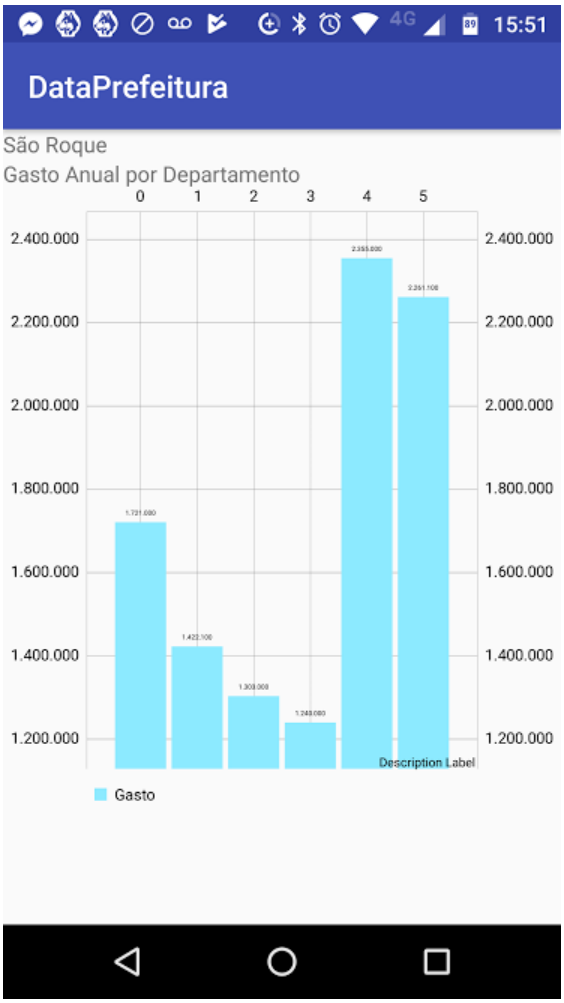


Figura 38: Exemplo de solução utilizando Android

5.9. ANÁLISE

Após a aplicação das validações os dados coletados passarão por uma análise quantitativa e qualitativa considerando as ferramentas em si e o contexto de dados abertos governamentais.

Na análise foi considerado a questão da utilidade da aplicação para coleta de dados de dados governamentais automatizada na realização das atividades propostas durante o estudo. Foi verificado se as funções disponíveis na aplicação conseguiram atender as necessidades dos envolvidos no estudo e que outras funcionalidades poderiam ser implementadas para sua melhoria.

Além da verificação da utilidade também foi analisado a facilidade de uso da ferramenta, tanto para os indivíduos com mais ou menos experiência em programação. Com essa análise será possível detectar que outras funcionalidades poderiam ser implementadas para facilitar o uso da ferramenta para os diversos tipos de usuários, como também que outros formatos os dados poderiam ser disponibilizados.

Foi feita uma verificação quantitativa com todos os dados coletados para que se possa traçar com precisão questões como: utilidade das funções disponibilizadas pela API para a criação de aplicações no contexto de dados abertos governamentais, facilidade da coleta automatizada de Web Crawlers perante coletas manuais nos portais da transparência, dificuldades encontradas na utilização da API, entre outros.

5.9.1. Análise Inicial dos Grupos

Antes da proposta do desafio de programação foi aplicado um Questionário Inicial para todos os participantes de ambos grupos. Esse questionário teve como objetivo traçar o perfil dos participantes no sentido de formação, experiência profissional, conhecimentos em linguagens de programação, tecnologias e DAG, que são itens importantes na resolução do desafio proposto na validação.

Um dos questionamentos foram em relação a idade dos participantes, onde no GA, 67% tinham entre 18 a 20 anos e 33% de 21 a 25 anos. Já no GP, 34% possuíam de 21 a 25 anos, 33% de 26 a 30 anos e 33% acima de 30 anos. Percebe-se que, a idade dos participantes do grupo GA é ligeiramente mais baixa que a do grupo GP, o que era esperado, pois o GA são alunos que ainda estão em formação e o GP são profissionais do mercado. Mas de maneira geral os participantes são jovens.

Foi verificada a experiência profissional dos grupos. Do grupo GA, 50% dos participantes não possuíam experiência profissional, 17% por menos de 1 ano e 33% entre 1 e 2 anos. Já do grupo GP, 33% possuem experiência entre 2 e 5 anos e 67% por mais de 5 anos. Ao analisar o perfil dos candidatos, apesar da maioria ser jovem, a maioria possui alguma experiência com TI, mesmo os alunos do grupo de alunos da graduação. Figura 39

Figura 39: Experiência Profissional de ambos grupos

Além da experiência profissional, também foi questionada a atual situação dos participantes em relação a estudo e carreira. Do Grupo GA, todos os participantes, no momento da validação eram estudantes de graduação. Do grupo GP, 33% dos participantes são profissionais de TI e estudantes e 67% somente profissionais de TI.

Grande parte do grupo de controle mostrado são estudantes de graduação. Nota-se que apesar da pergunta anterior todos tenham mostrado algum grau de experiência com TI, boa parte dos participantes no momento da aplicação do questionário eram somente estudantes de graduação.

Também foi mapeado o nível de conhecimento em diversas tecnologias, como Java Web Service, Banco de Dados, Orientação a Objetos, JSON, Java Script, JQuery, PHP, JSP/Servlet e Web Crawlers. Os participantes tiveram que se auto avaliar nas seguintes categorias: sem conhecimento, conhecimento acadêmico apenas, conhecimento acadêmico e prático, conhecimento acadêmico, teórico e prático e aplicação do conhecimento no mercado de trabalho.

No Grupo GA Os participantes mostraram resultados variados em conhecimentos em diversas tecnologias, mas o que prevaleceu de forma geral em todas elas foram conhecimento

acadêmico, podendo ser acompanhado ou não por prática ou teoria sobre a tecnologia. Alguns participantes mostraram aplicação de alguns itens no mercado de trabalho. Em relação a cada tecnologia separadamente, as conclusões foram: Figura 40

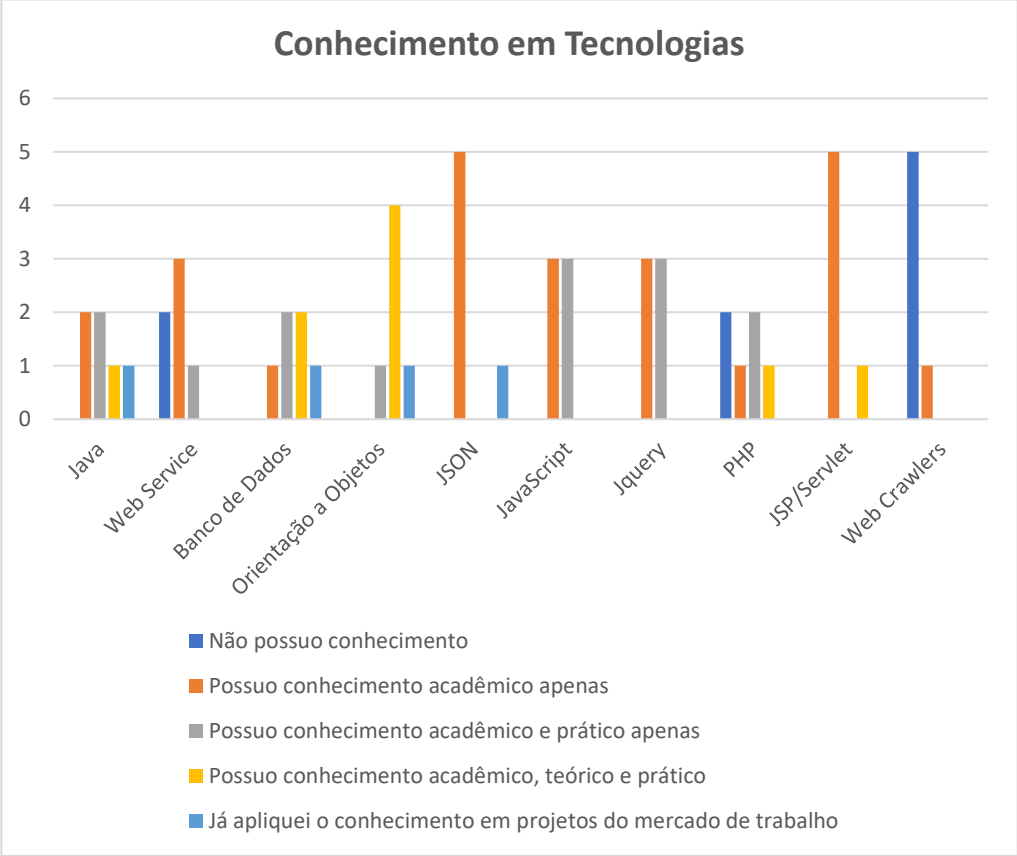


Figura 40: Conhecimentos em tecnologias dos participantes do grupo GA

*Já o*s participantes do grupo GP mostraram conhecimento e aplicação no mercado de trabalho na maioria das tecnologias propostas no questionário Figura 41

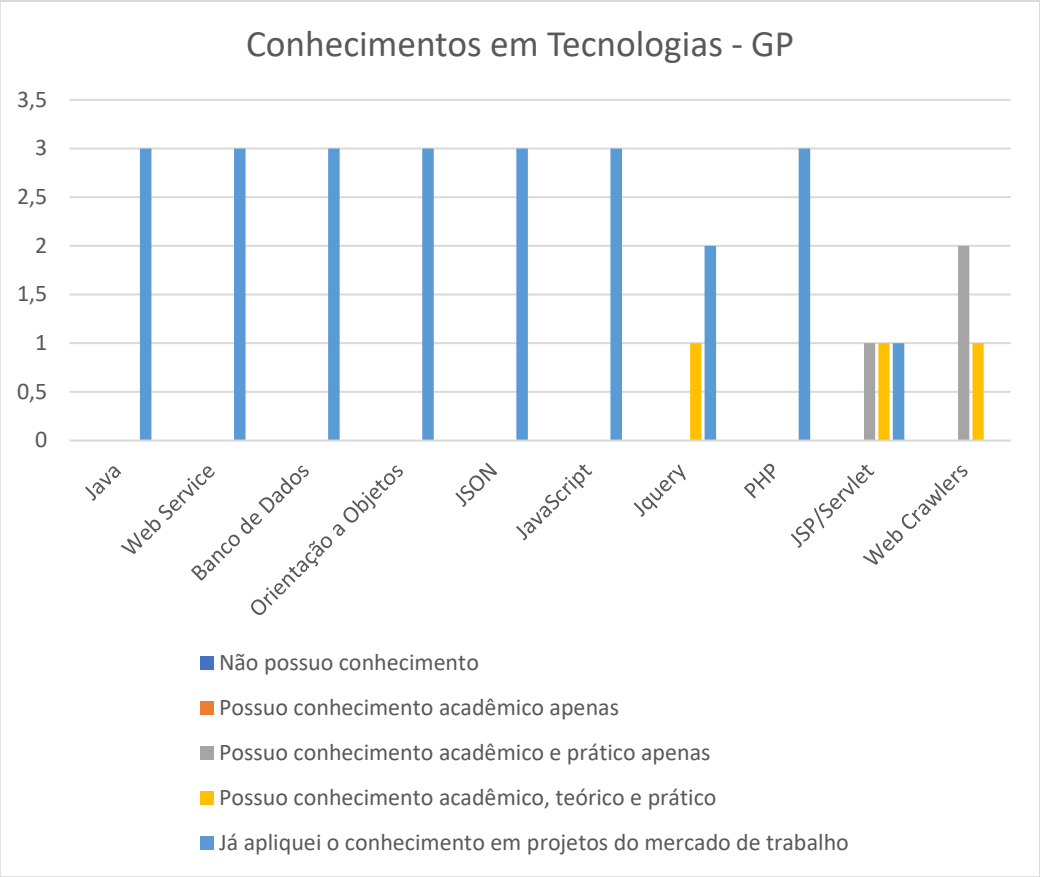


Figura 41: Conhecimentos em tecnologias dos participantes do grupo GP

Nota-se, ao analisar os dois grupos juntos, que em relação á linguagens de programação e orientação à objetos a maioria tem conhecimento seja acadêmico, teórico, prático e até mesmo aplicação no mercado de trabalho. Em relação a elas o grupo mostra mais conhecimentos em Java, Java Script e menos em PHP e em Java utilizando JSP e Servlets. Em relação a banco de dados a todos mostram algum grau de conhecimento, sendo que parte já aplicou esses conhecimentos no mercado de trabalho. Já em relação a trabalhar com JSON, os participantes se dividem em dois casos, uns com conhecimento acadêmico e outro com aplicação no mercado de trabalho. Em Web Services, todos os participantes mostraram algum grau de conhecimento, desde de apenas acadêmico até aplicação no mercado de trabalho. E por fim, em relação à Web Crawlers, boa parte do grupo mostrou sem conhecimento, mas alguns mostram conhecimento acadêmico e prático, mas ninguém chegou a aplicar no mercado de trabalho.

Como a proposta de desafio está relacionada a DAG, foram feitos alguns questionamentos em relação à área para os participantes. A primeira pergunta foi em relação aos conhecimentos em relação á DAG. Os participantes mostraram um conhecimento que varia de mediano a baixo em ambos grupos. Porém o grupo GP mostrou um conhecimento maior se

comparado com o grupo GA. Ao analisar os resultados separadamente por grupo temos os resultados a seguir: Figura 42

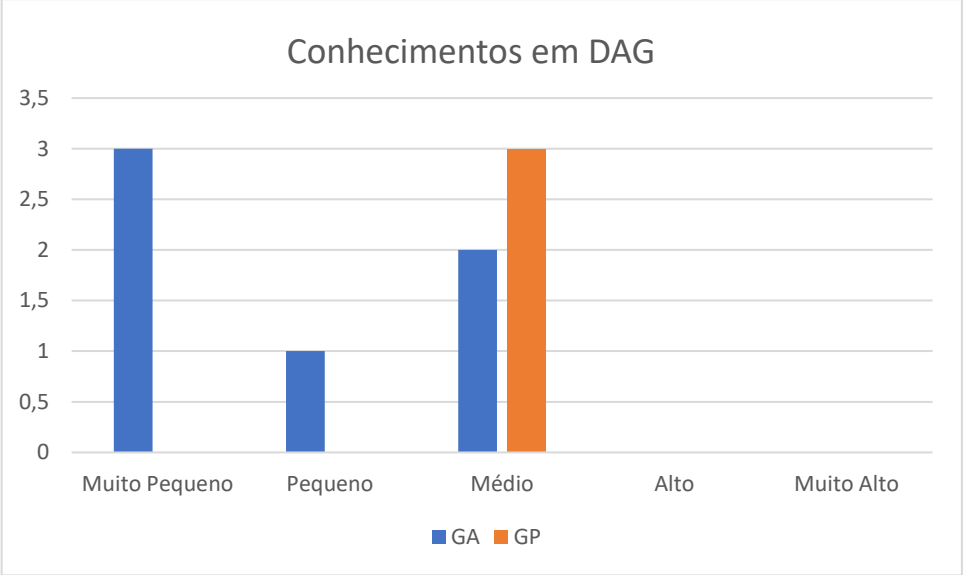


Figura 42: Conhecimentos em DAG de ambos grupos

Outro questionamento foi em relação ao hábito de acessar sites que disponibilizam DAG. Do grupo GA, a maioria (87%) não possuem esse hábito, comparado com apenas 17% que possui. Já no grupo GP, todos os participantes não possuem o hábito de acessar DAG. Grande maioria, independentemente do grupo não possuem o hábito de pesquisarem DAG. Logo, espera-se que muitos não conheçam totalmente a estrutura e a forma de publicação desses dados.

Apesar do conhecimento mediano para baixo e do pouco hábito do acesso de DAG pelos participantes, quando questionados sobre a importância da publicação desses dados, os resultados foram um pouco diferentes.

No Grupo GA, de forma geral os participantes concordam da importância da publicação de DAG pelos governos em diversos aspectos. A relação de cada item estão na Figura 43.

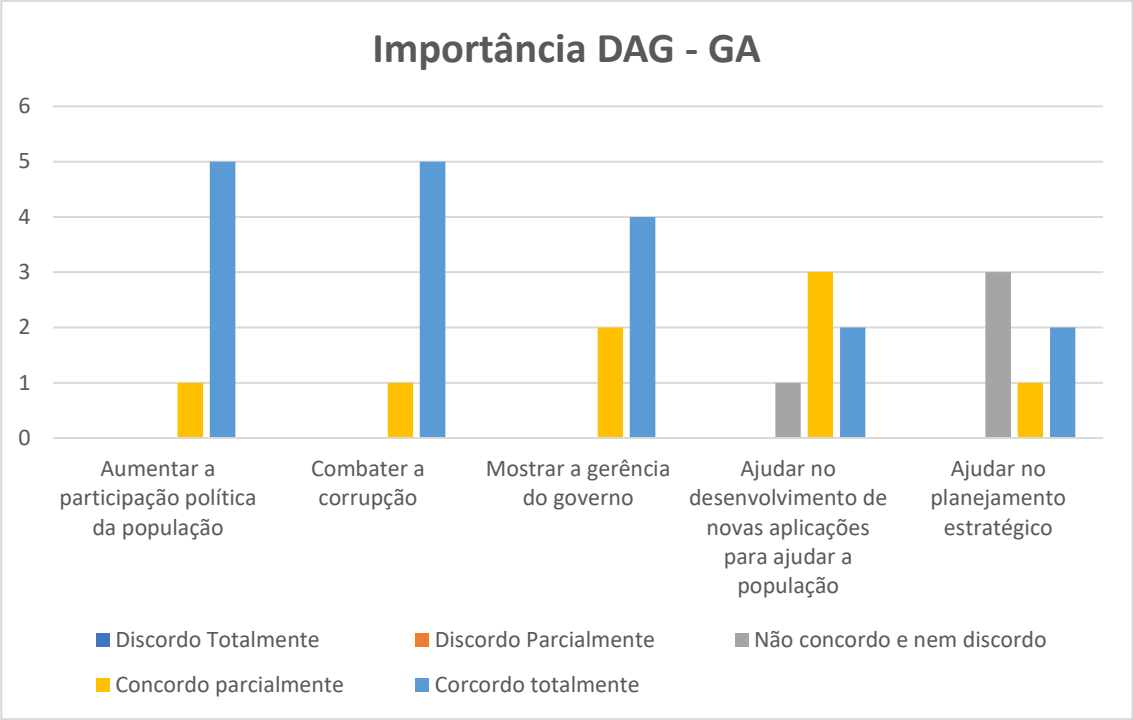


Figura 43: Importância de DAG dos participantes do grupo GA

No grupo GP, grande parte também concorda com a maioria dos itens, como mostra na Figura 44.

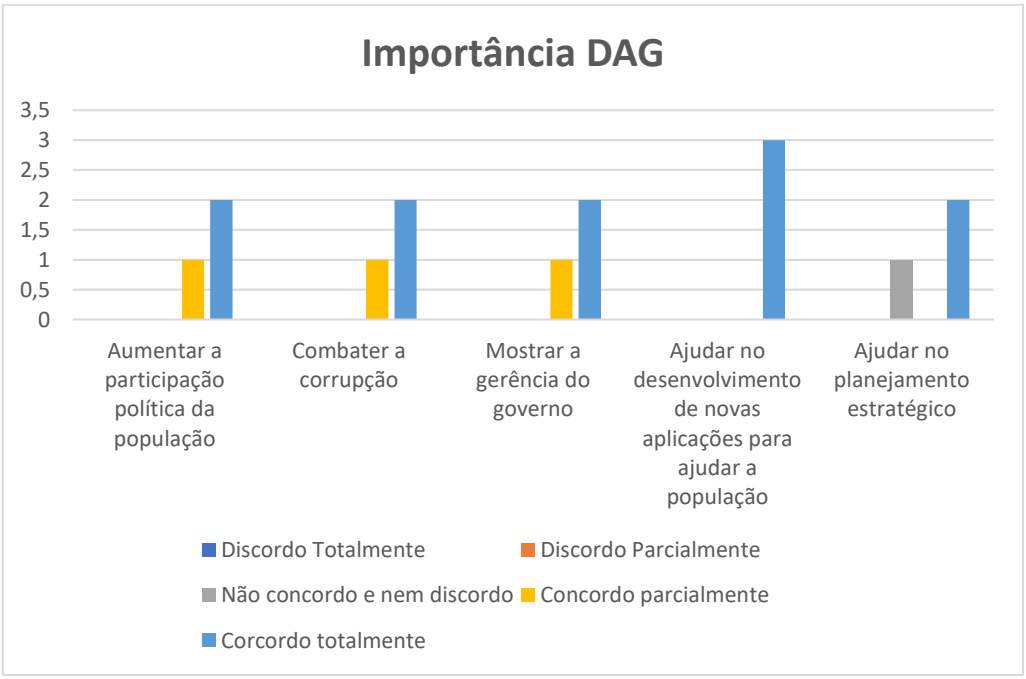


Figura 44: Importância da publicação de DAG dos participantes do grupo GP

Nota-se que em todos os itens, os participantes de forma geral concordam parcialmente ou totalmente que DAG são importantes. Porém, no GP, os participantes concordam mais com a importância de DAG para o desenvolvimento de aplicações para ajudar a população geral, que é o que a validação propõe.

5.9.2. Análise pós desafio

Após a conclusão do desafio de programação, foi aplicado um outro questionário, titulado de questionário final a todos os participantes, independentemente do perfil do mesmo. Este foi aplicado após a conclusão do desafio de programação utilizando a API DAG Prefeituras. Esse questionário tem como objetivos principais mapear como foi a utilização da API e a usabilidade e facilidade de uso da mesma.

Uma das avaliações foi na utilização de linhas de código no desenvolvimento das soluções. No grupo GA, 67% dos participantes utilizaram de 101 a 200 linhas e 33% entre 201 a 300 linhas. Já no grupo GP, 34% utilizaram de 10 a 50 linhas, 33% de 101 a 200 linhas e 33% de 201 a 300 linhas Figura 45.

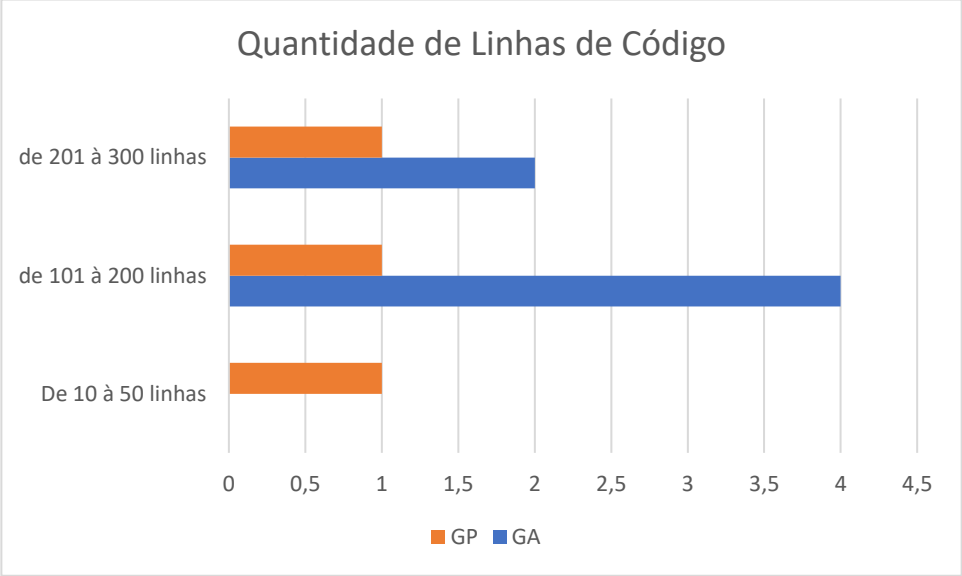


Figura 45: Quantidade de Linhas de Código

Ao analisar os dados, nota-se que grande maioria dos participantes tiveram uma variação de 100 à 300 linhas utilizadas para o desenvolvimento da solução. Isso se deve a vários fatores, como organização do código, forma de chamada da API DAG Prefeituras e das APIs relacionadas à criação de gráficos, etc. Realmente, em Java Script, por exemplo são várias linhas utilizadas para fazer a chamada e retornar os dados, porém na página inicial da API DAG

Prefeituras é ilustrado como fazer essa chamada, facilitando a vida do desenvolvedor, que copiaria esse código e alteraria o mesmo conforme suas necessidades.

Outra análise foi em relação ao tempo demorado para cumprir o desafio proposto. No grupo GA, todos os participantes utilizaram mais de 3 horas para finalizar a solução, mas todos terminaram em um dia. No grupo GP, 67% demoraram mais de 3 horas 3 33% de 1 a 2 horas.

A maior parte dos participantes demoraram mais de 3 horas para finalizarem a solução, mas todos conseguiram concluir a mesma em um mesmo dia. Na Q3 são mostradas as dificuldades que influenciaram esse tempo, que varia desde utilização da linguagem de programação, criação de gráficos, entre outros. Isso se deve porque foi proposto um desafio utilizando a API DAG Prefeituras, onde os participantes tinham que ir além de chamar os dados da API e mostrar na tela. Além de capturar os dados eles precisavam selecionar os desejados e ainda os exibir em um gráfico, o que demandou pesquisa para seleção e utilização de outra API para gerar os gráficos.

Foi mapeada também as possíveis dificuldades que os participantes de ambos grupos encontraram no desafio. As dificuldades são mostradas na Figura 46

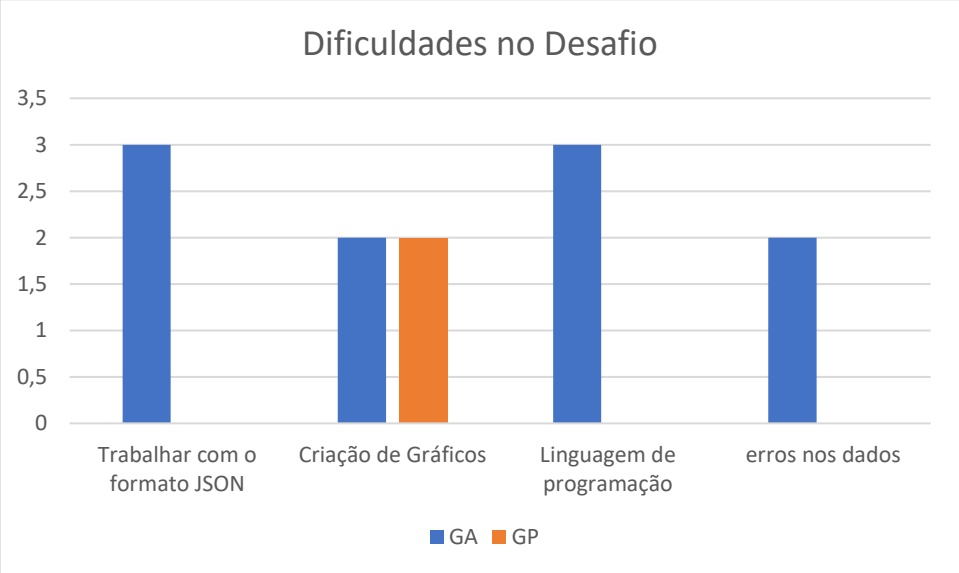


Figura 46: Dificuldades no Desafio de ambos grupos

As maiores dificuldades relatadas pelos usuários do grupo GA foram “Trabalhar com o objeto JSON”, “Criação de gráficos” e “Linguagem de Programação”. Nota-se que apesar de ter a opção, nenhum dos participantes mostrou dificuldades diretas na utilização da API DAG Prefeituras em si, e sim dificuldades relacionadas ao próprio processo da criação de um software e do trabalho com objetos do tipo JSON. Apesar das dificuldades da utilização de JSON, o

formato é aceito segundo os princípios dos DAG e é amplamente utilizado em diversas aplicações. Uma hipótese para a dificuldade para o uso do JSON seja a experiência em programação dos avaliados, já que o grupo que mostrou essa dificuldade foram os alunos da graduação. Mas essa verificação mostra a possibilidade da disponibilização dos dados da API DAG Prefeituras em outros formatos além do JSON, mas claro esses formatos precisam ser abertos e aceitos segundos os princípios dos DAG.

A linguagem de programação foi relatada também com uma dificuldade pelos participantes do grupo GA para resolver o desafio. Durante a validação alguns relataram algumas dificuldades com a linguagem de programação, muitos devido à pouca experiência no desenvolvimento utilizando linguagens para Web. No desafio não foi estabelecida linguagem de programação específica para a criação do software, porém a API DAG Prefeituras mostra exemplos de utilização usando Java Script e PHP, logo a maioria dos participantes optaram por usar linguagens web. Porém, no outro grupo (GP), um dos participantes optou em desenvolver um aplicativo para Android invés de fazer uma página Web com os resultados. Através disso na validação foi detectada outra demanda, de disponibilizar mais exemplos de utilização da API em outras linguagens de programação. Apesar do JSON funcionar com a maioria das linguagens, mostrar exemplos em mais linguagens facilitaria a utilização da API.

Apesar de relatados com menos frequência, erros de português e dados escritos incorretamente foram relatados. A API DAG Prefeituras coleta os dados como foram publicados no site original, logo dados publicados com erros são capturados com erros. Como sugestões foi proposto uma filtragem e correção de dados errados antes da disponibilização dos mesmos pela API.

Outra dificuldade relatada foi em relação a criação de gráficos. Essa dificuldade por sua vez foi apontada pelos dois grupos (GA e GP). Esse ponto era esperado, pois um dos itens para tornar essa validação um “desafio de programação” foi justamente criar um gráfico com os dados capturados. Apesar de existirem diversas APIs para criação de gráficos, os participantes tiveram o trabalho de pesquisar, escolher e aprender a utilizar uma API, isso durante a validação. Alguns relataram inclusive que nunca tinham gerados gráficos anteriormente no desafio. Apesar dessas dificuldades, todos os participantes conseguiram concluir o desafio no período proposto.

Em relação a tecnologias, na proposta do desafio foi deixado a livre escolha do participante que linguagens de programação, frameworks, etc. seria utilizado para desenvolver

o software. A API DAG Prefeituras possui na sua página inicial exemplos de código utilizando a API com Java Script e com PHP Figura 47.

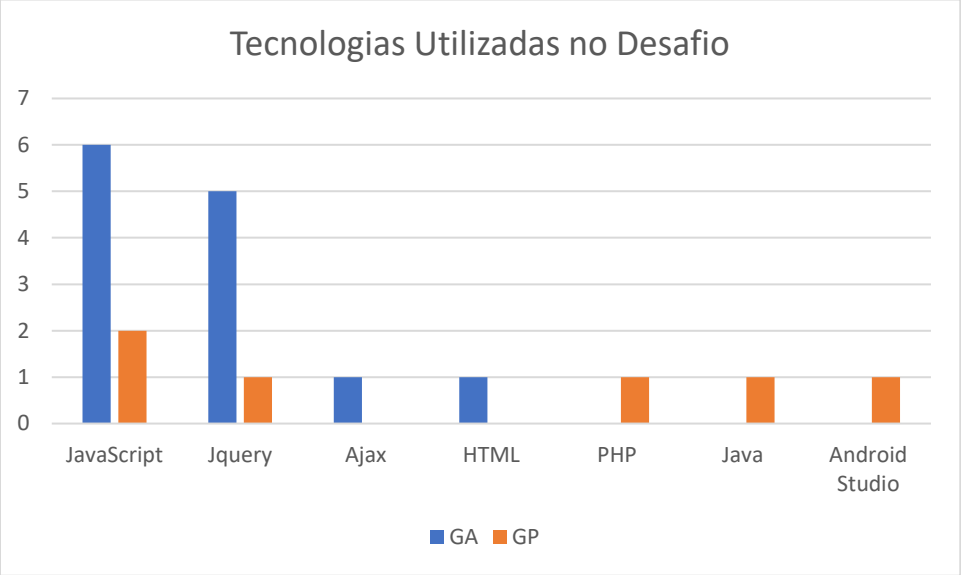


Figura 47: Tecnologias Utilizadas em ambos grupos

Além de usar a API DAG Prefeituras, os participantes precisaram utilizar outra API para gerar os gráficos. A escolha da API ficou livre para os participantes. Percebe-se que grande parte dos participantes de ambos os grupos utilizaram a API Charts.js seguida da Google Charts. Isso deve-se que a maioria dos participantes terem utilizado Java Script para resolução do desafio e essas APIs serem as mais atuais. Nota-se também que durante o experimento, vários usuários do grupo GA relataram não ter gerado gráficos anteriormente ao desafio.

Foi feita uma pergunta aberta, para os participantes relatarem críticas e sugestões na API DAG Prefeituras. Dentre elas estão em relação a correções de português dos dados capturados, já que os mesmos são disponibilizados da forma que foram escritos na origem, e também em relação a padronização dos campos e nome das funções de ambas cidades. Outra questão é também sobre a estaticidade dos dados disponibilizados pela API. As aplicações dependentes da API podem parar de funcionar caso a mesma seja modificada. Outra sugestão foi a criação de uma função que retornasse os nomes das funções disponíveis pela API.

5.9.2.1. Utilidade e Facilidade de Uso da API DAG Prefeituras

Além dos itens relatados anteriormente, foi verificado a facilidade de uso e utilidade da API baseado em TAM (DAVIS, 1989). Esses itens estão relacionados à seguintes questões levantadas no início da validação: “Q1: A utilização da API facilita a execução das tarefas propostas”, onde é utilizada como métrica a Utilidade da API e “Q2: Qual o nível de dificuldade da utilização da API levando em consideração aos conhecimentos prévios em programação.”, onde são utilizadas as métricas Facilidade de Uso juntamente com a análise inicial dos participantes considerando seus níveis de conhecimento em programação.

Primeiro foi verificado o grau de facilidade de uso da API DAG Prefeituras para a realização do desafio. No grupo GA, 4 atribuiram nota 8 para a API e 2 nota 9. A média de nota da API considerando o grau de satisfação foi de 8,33. Já no grupo GP, 2 relataram nota 9 e 1 nota 10, com uma média aritmética de 9,33.

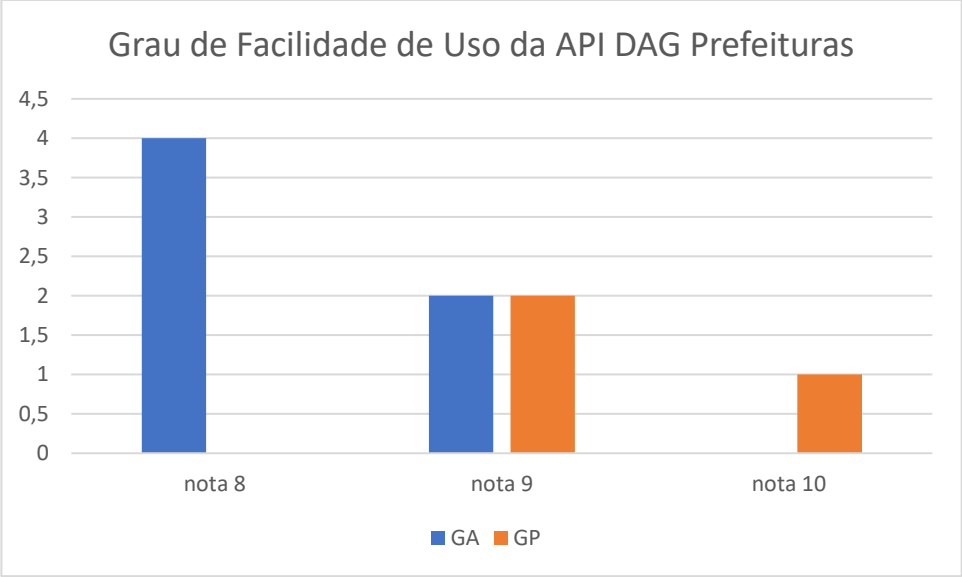


Figura 48: Grau de Facilidade de Uso da API DAG Prefeituras em ambos grupos

Analisando os dados, nota-se que a aceitação da API DAG Prefeituras foi boa pelos participantes de ambos grupos, tendo uma média do grupo GA de 8,33, do grupo GP de 9,33 e uma média geral de 8,66 Figura 48.

Partindo de uma análise mais qualitativa, durante a execução das validações foi percebido que o processo de captura dos dados pela API não foi tão demorado, principalmente pela forma de acesso a API e tempo médio de exibição dos dados. Durante o desafio os

participantes foram orientados a primeiramente capturar os dados desejados utilizando a API para que depois fosse criado o gráfico com os mesmos. Nessa faze de captura os participantes conseguiram com uma certa rapidez, principalmente devido os exemplos de chamada à API oferecidos na página inicial da API DAG Prefeituras. Como conclusão, um desenvolvedor, independente do seu nível de experiência não terá grandes problemas na captura dos dados em si utilizando a API. Mas como a API apenas disponibiliza os dados, o desenvolvedor pode ter problemas com outras etapas na construção do software. No caso dessa validação, os participantes tiveram como desafio criar um gráfico com os dados.

Em relação a utilidade, foi considerado essa análise a API DAG Prefeituras no contexto DAG. No grupo GA, 1 atribuiu nota 9 e 5 deram nota 10. A Média aritmética entre os participantes foi de 9,16. E no grupo GP, Figura 49 todos os participantes do grupo atribuíram nota 10. A média aritmética entre as notas foi de 10 (Figura 49).

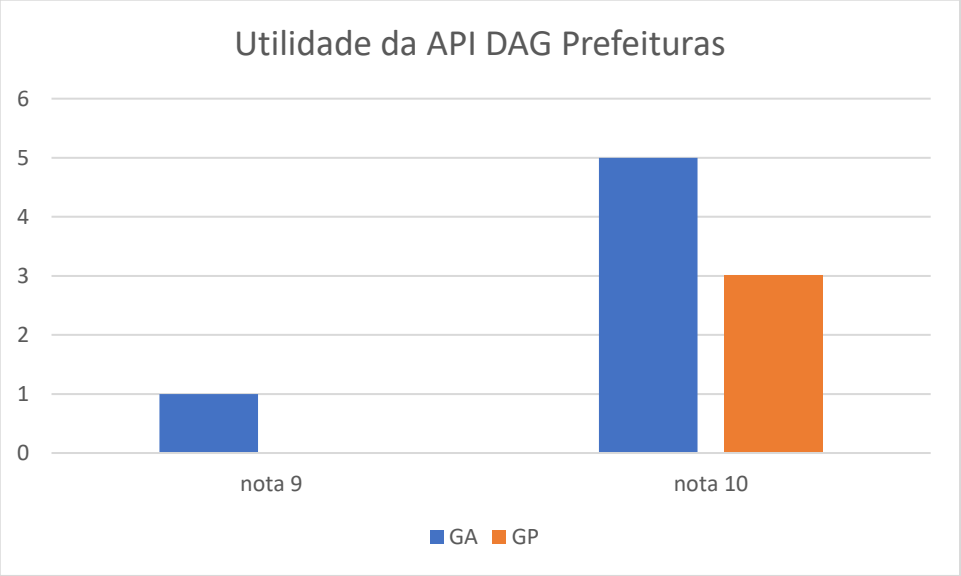


Figura 49: Grau de Utilidade da API DAG Prefeituras

No contexto de DAG, o grau de satisfação dos participantes foi alto, eles deram notas entre 4 e 5 (numa escala de 1 a 5) para a importância da API neste contexto. Em relação à média, o GA teve uma média de 9,16, o GP de 10 e a média geral foi de 9,44.

5.9.2.2. Análise da Importância de DAG entre os grupos GA e GP

Durante a validação em ambos grupos, além do desafio de programação foram feitos alguns questionamentos sobre DAG. Uma das questões o participante tinha que apontar a importância de DAG sobre quatro perspectivas diferentes.

As perspectivas analisadas foram as seguintes:

P1: Aumentar a participação política da população

P2: Combater a Corrupção

P3: Mostrar a Gerência do Governo

P4: Ajudar no desenvolvimento de novas aplicações para ajudar a população

P5: Ajudar no planejamento estratégico

Para cada uma dessas perspectivas os usuários tinham cinco opções de resposta entre concordância total e discordância total. Para cada resposta foi atribuído uma pontuação, como mostra a Tabela 3.

Tabela 3: Níveis de concordância com as perspectivas de DAG

Nível de Concordância	**Pontuação**
Discordo totalmente	0
Discordo parcialmente	2,5
Não concordo e nem discordo	5
Concordo parcialmente	7,5
Concordo totalmente	10

Analisando as respostas de cada grupo separadamente, para o grupo GA tem-se os seguintes resultados: Tabela 4

Tabela 4: Perspectivas DAG dos participantes do grupo GA

Perspectiva	Somatório	Média	Mediana	Moda
P1	57,50	9,58	10,00	10,00
P2	57,50	9,58	10,00	10,00
P3	55,00	9,17	10,00	10,00

P4	47,50	7,92	7,50	7,50
P5	42,50	7,08	6,25	5,00

Ao analisar as perspectivas, as que tiveram maiores notas foram as P1, P2 e P3. Já as perspectivas P4 e P5 tiveram boas notas, mas entre 2 e 3 pontos abaixo das perspectivas anteriores (Figura 50).

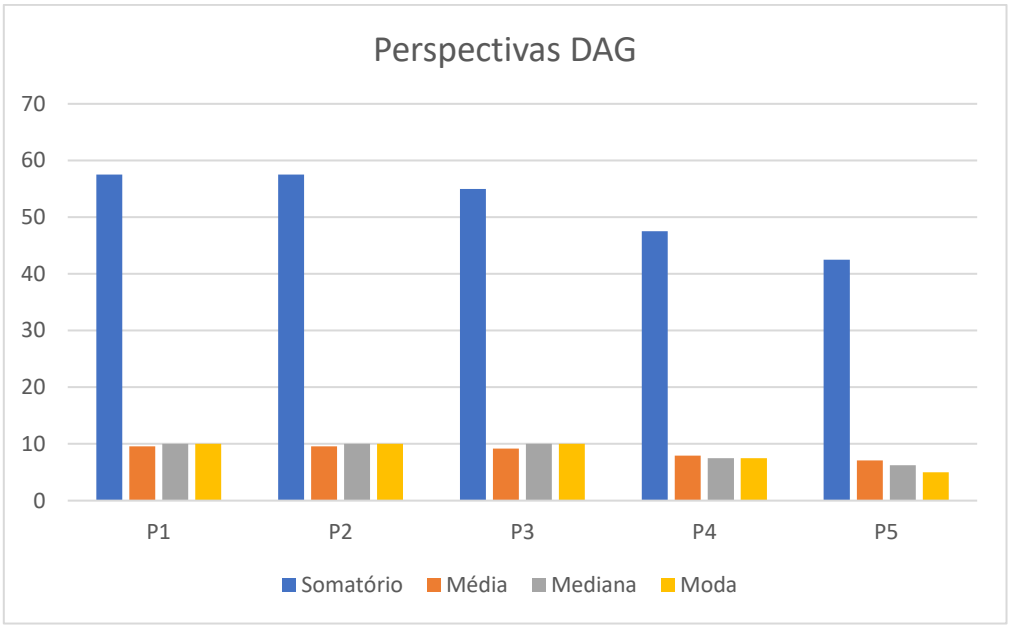


Figura 50: Perspectivas DAG dos participantes do grupo GA

Já no grupo GP, os resultados estão na Tabela 5

Tabela 5: Perspectivas DAG dos participantes do grupo GP

Perspectiva	Somatório	Média	Mediana	Moda
P1	27,50	9,17	10,00	10,00
P2	27,50	9,17	10,00	10,00
P3	27,50	9,17	10,00	10,00
P4	30,00	10,00	10,00	10,00
P5	25,00	8,33	10,00	10,00

Já os dados do grupo GP se comparado com os dos participantes do grupo GA estão com notas ligeiramente maiores, tendo uma variação entre 8,33 e 10. A perspectiva com menor nota foi a P5 e as com maiores notas as P1, P2 e P3 (Tabela 5 e Figura 51).

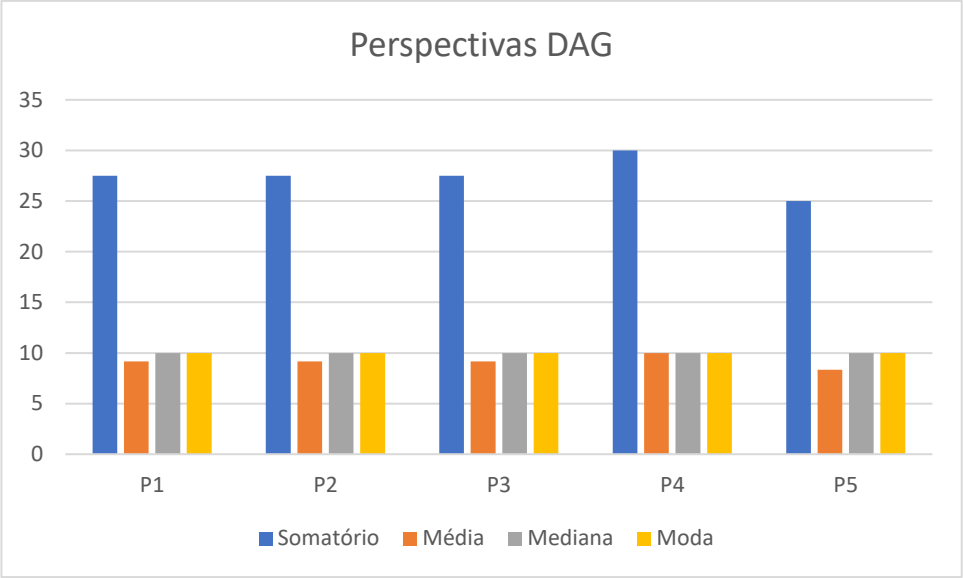


Figura 51: Perspectivas DAG dos participantes do grupo GP

Agora se for analisada as notas das perspectivas de ambos grupos juntamente, os resultados são mostrados na Tabela 6.

Tabela 6: Perspectivas DAG de todos participantes

Perspectiva	Somatório	Média	Mediana	Moda
P1	85,00	9,44	10,00	10,00
P2	85,00	9,44	10,00	10,00
P3	82,50	9,17	10,00	10,00
P4	77,50	8,61	10,00	10,00
P5	67,50	7,50	7,50	10,00

Ao juntar os resultados de ambos grupos se teve um crescimento em comparação aos dados do grupo GA e uma pequena queda se comparar com os dados do grupo GP (Figura 52).

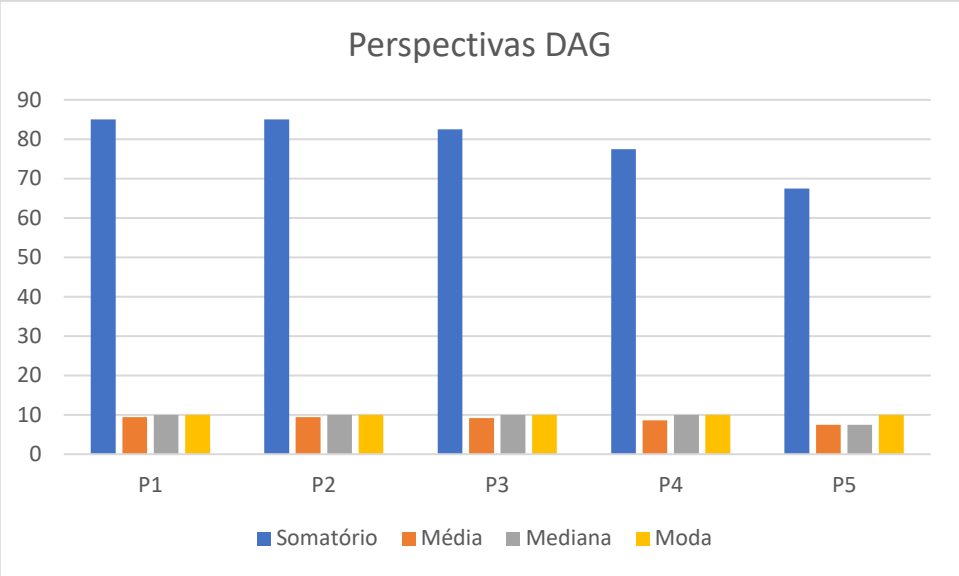


Figura 52: Perspectivas DAG de ambos grupos

5.9.2.3. Análise da API DAG Prefeituras entre os grupos GA e GP

A validação da API DAG Prefeituras consistiu da proposta de um desafio de programação para dois grupos de diferentes experiências: usuários com pouca experiência – alunos da graduação (GA) e usuários com muita experiência – profissionais do mercado (GP).

Para a análise seguindo os princípios de Travassos (TRAVASSOS, GUROV e AMARAL, 2002) e Brasili (BASILI, SELBY e HUTCHENS, 1986), a partir do Questionário Final, foram estabelecidas as seguintes competências:

C1: Satisfação da API DAG PREFEITURAS

C2: Utilidade da API DAG PREFEITURAS no contexto DAG

As notas dadas por cada usuário de cada grupo (GA e GP) estão descritas a seguir.

Tabela 7: Competências de Utilidade e Facilidade de Uso da API DAG Prefeituras do grupo GA

Competência	Somatório	Média	Mediana	Moda
C1	52,00	8,67	8,00	8,00
C2	58,00	9,67	10,00	10,00

De maneira geral, as notas paras as competências pelos participantes do grupo GA foram satisfatórias. Tabela 7 A competência C1 teve um somatório de 52, média de 8,67, mediana de

8 e moda de 8. Já a competência C2 teve um somatório de 58, média de 9,67, mediana de 10 e moda de 10. De forma geral ambas competências tiveram desempenho satisfatório (Figura 53).

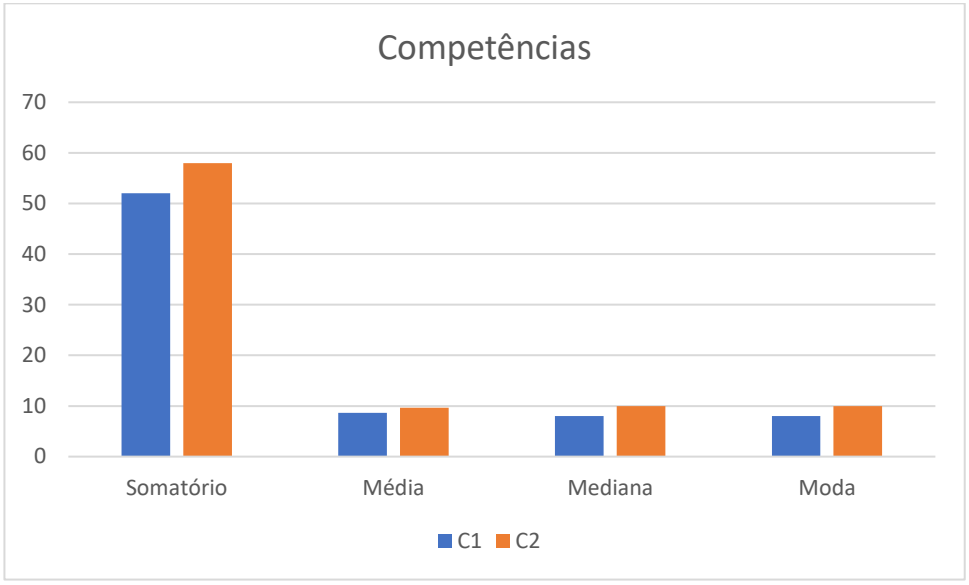


Figura 53: Competências dos participantes do grupo GA

Já as notas dadas por cada participante do grupo GP estão descritas na Tabela 8 .

Tabela 8: Competências de Utilidade e Facilidade de Uso da API DAG Prefeituras do grupo GP

Competência	Somatório	Média	Mediana	Moda
C1	28,00	9,33	9,00	9,00
C2	30,00	10,00	10,00	10,00

De maneira geral o desempenho do grupo GP também foi satisfatório (Tabela 8). A Competência C1 teve um somatório de 28, uma média de 9,33, uma mediana de 9 e moda de 9. Já a competência C2 teve um somatório de 30, média de 10, mediana de 10 e moda de 10 (Figura 54).

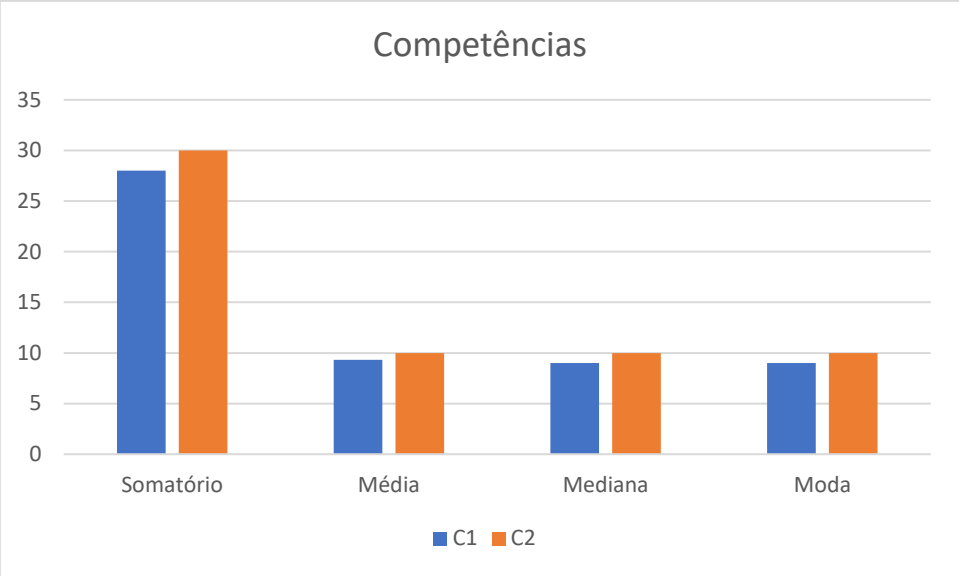


Figura 54: Competências dos participantes do grupo GP

Comparando os resultados de ambos grupos, os participantes do grupo GP atribuíram notas ligeiramente maiores para ambas competências, em torno de 0,5 pontos acima.

Ao analisar os mesmos dados, mas com os participantes dos grupos GA e GP, obteve-se os seguintes resultados: Tabela 9

Tabela 9: Competências de Utilidade e Facilidade de Uso da API DAG Prefeituras de todos os participantes

Competência	Somatório	Média	Mediana	Moda
C1	80,00	8,89	9,00	8,00
C2	88,00	9,78	10,00	10,00

Os grupos juntos tiveram-se uma ligeira queda nos resultados, ainda sim obteve-se bons resultados. Para a competência C1 o somatório foi de 80, a média de 8,89, a mediana de 9 e a moda de 8. Já a competência C2 teve somatório de 88, média de 9,78, mediana de 10 e moda de 10 (Figura 55).

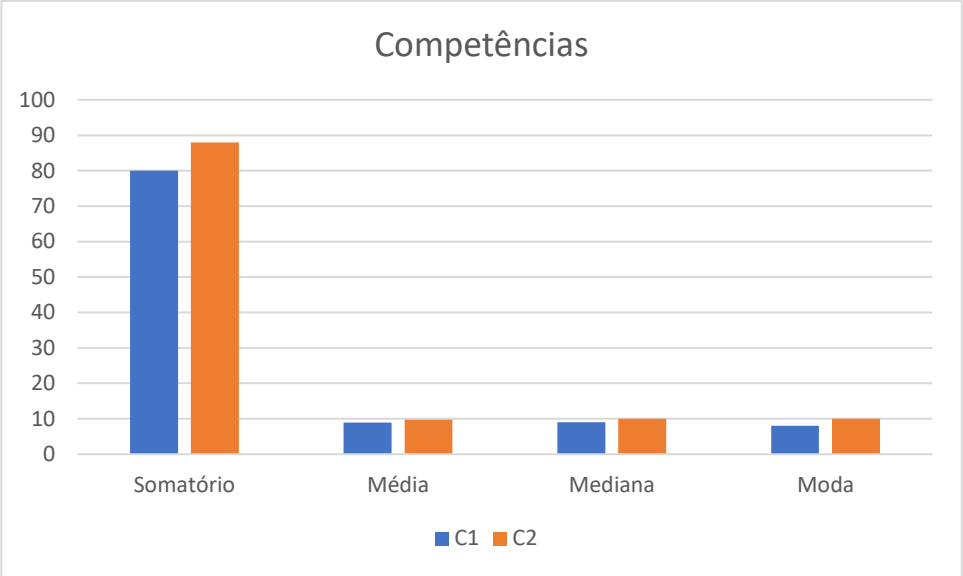


Figura 55: Competências de todos os participantes

5.10. CONSIDERAÇÕES FINAIS

Para a validação da proposta desenvolvida (Web Crawler e API DAG Prefeituras) foi feita uma avaliação da proposta considerando desenvolvedores com diferentes níveis de experiência.

Essa avaliação utilizou princípios de experimentação de software apresentados por Travassos e Brasili, além de conceitos apresentados por Davis.

Foi feito um estudo de caso onde foi proposto um desafio onde os participantes tiveram que criar uma aplicação utilizando os dados providos pela API DAG Prefeituras. Esse desafio foi aplicado para dois grupos de participantes: estudantes da graduação e profissionais do mercado.

De maneira geral a API DAG Prefeituras foi bem aceita pelos grupos GA e GP ao analisar a utilidade e facilidade de uso da API. Foi estabelecida duas competências, relacionadas a Facilidade de Uso e Utilidade da API DAG Prefeituras, correspondente as questões Q1 e Q2 respectivamente. Além disso foi analisada também a quantidade de linhas de código médias.

Em relação à Q1, as notas foram satisfatórias, girando entre 8 a 10, porém, o grupo GP a média foi ligeiramente maior.

E na Q2, as notas também foram satisfatórias, girando entre 9 e 10, neste caso também, no grupo GP as notas tiveram uma pequena variação acima comparado com o grupo GA. Ao comparar ambos grupos, o grupo GP de forma geral, obteve um grau de satisfação maior na API do que no grupo GA, apesar da diferença entre ambos grupos serem pequenas.

Em relação as linhas de código, percebe-se de forma geral a maior parte dos participantes do grupo GA utilizaram entre 101 a 200 linhas de código, já o grupo GP tive uma variação entre 10 à 300 linhas de código.

Além da API em si, os participantes consideram a publicação de DAG importantes seguindo diversas perspectivas, como mostra na Figura 43 e na Figura 44.

Mas a API precisa ser aprimorada como em relação a nomenclatura de algumas funções, ao tratamento dos dados capturados na perspectiva da escrita e padronização e também na disponibilização de dados de mais cidades, como foi apontado pelos próprios participantes durante a validação.

6. CONCLUSÃO

Atualmente, graças aos avanços tecnológicos a maciça utilização de TICs a publicação de DAG aumentou consideravelmente em grande parte dos países, como no Brasil.

A publicação de DAG é uma das áreas que envolvem o conceito de cidades inteligentes. Uma cidade inteligente prevê a utilização de tecnologia para prever as necessidades das cidades (SUAKANTO, SUPANGKAT, *et al.*, 2013). Mas DAG vão além de uma das medidas para melhorar uma cidade do ponto de vista tecnológico, ela tem um impacto social importante, pois é um valioso recurso estratégico além de ajudar na transparência dos governos.

O Brasil é um país marcado por vários escândalos de corrupção em todas suas esferas. Diariamente a imprensa publica reportagens relativas a esse tema. Muitos desses casos são relativos ao governo federal e estadual, mas engana-se que nos municípios não existe esse problema. Basta atentar-se ao noticiário local de sua cidade que verá escândalos parecidos nessa temática.

Existem várias formas de se combater a corrupção, uma delas é verificando como os governantes estão gastando o dinheiro público. Logo, a utilização de DAG é uma forma de visualizar de forma prática de olhar esses dados, facilitando também, a cobrança da população para o governo para que o dinheiro seja gasto da melhor forma possível.

Uma boa forma de começar o combate a corrupção é preocupando-se primeiramente com os municípios, que estão mais pertos das pessoas, além de possuir tamanho reduzido (se comparado com um Estado inteiro e mesmo o país), facilitando o controle e cobrança. Isso foi um dos fatores que levaram a escolha de municípios brasileiros para esta pesquisa.

Outra questão foi em relação a forma de publicação dos dados em si. Durante a pesquisa foi constatado que o Brasil, de forma geral realiza a publicação de DAG, porém a maioria dos órgãos não seguem os princípios dos DAG (CARTILHA, 2016), principalmente se for analisado os dados relativos aos municípios. Com isso, tem-se um grande volume de dados, porém com pouca utilidade, se for pensando em utilizá-los de forma automatizada.

Logo, esse trabalho teve como objetivo principal o tratamento de DAG de municípios brasileiros, os disponibilizando em formatos aceitos conforme as diretrizes legais nacionais (CARTILHA, 2016) e os princípios gerais dos DAG (OPEN, 2016). Disponibilizando esses dados de forma adequadas eles podem ser utilizados em aplicações de terceiros que auxiliem na gestão de dados abertos e na mostra de gastos governamentais de forma mais facilitada.

Para que esses dados pudessem ser disponibilizados, primeiramente eles precisaram ser capturados. Essa captura deu-se de forma automatizada, através da utilização de um Web

Crawler. Um Web Crawler é um software capaz de capturar dados de páginas Web de forma automática e hierárquica, seguindo um conjunto de instruções que levam em consideração a padronizações existentes nas páginas. Esse software foi criado inicialmente para capturar os dados de duas cidades do Estado de São Paulo, São Roque e Capivari. Porém, o software pode ser adaptado para que possa ser utilizado para capturar dados de outras cidades, desde que a mesma os disponibilizes em páginas HTML.

Para a disponibilização desses dados para o usuário final foi criado uma API, denominada API DAG Prefeituras, que acesa a base de dados onde os dados capturados pelo Web Crawler e retorna os dados em um dos formatos aceitos conforme os princípios dos DAG, o JSON.

A disponibilização de dados em formatos adequados permite que desenvolvedores os utilize em diversas aplicações que possa ajudar os cidadãos, empresas e o próprio governo a melhorar suas gestões e ajudar no planejamento estratégico, seja do próprio governo como também de empresas, pois todos são órgãos da sociedade.

Foram realizados uma validação preliminar e um estudo de caso para verificar a satisfação e a utilidade da API para desenvolvedores de diferentes níveis de experiência. Nessa avaliação foram considerados itens como utilidade e facilidade de uso da API para a resolução do desafio proposto e para o contexto geral de DAG.

No estudo de caso, um grupo de desenvolvedores com pouca experiência e outro com desenvolvedores experientes desenvolveram uma aplicação utilizando a API DAG Prefeituras e apontaram suas impressões sobre ela. Analisando os resultados do experimento, de forma geral a utilização da API foi bastante satisfatória, sendo melhor aceita no grupo de desenvolvedores com mais experiência. Todos os participantes obtiveram sucesso durante o experimento, e não consideraram a API difícil de utilizar, além de reconhecerem a sua importância na disponibilização de DAG de municípios brasileiros. Porém a API ainda precisa de alguns ajustes, detectado durante o experimento e também relatado pelos próprios participantes. Alguns ajustes incluem a padronização da nomenclatura, disponibilização de mais cidades, melhorias de desempenho, tratamento ortográfico dos dados, entre outros. De forma geral os resultados mostraram-se positivos, a API ainda precisa de aprimoramentos, mas ela mostrou-se funcional e com capacidade de expansão.

O tratamento de DAG hoje mostra-se muito importante para que eles possam ser filtrados, classificados e utilizados de forma automatizada, permitindo que volumes cada vez maiores de dados possam ser analisados de forma mais rápida possível. Esse tipo de tratamento permite coletar um volume cada vez maior de informações relevantes para os governos,

permitindo uma melhor fiscalização de todos de como o dinheiro público, cada vez mais escasso, está sendo utilizado pelos governos. A boa utilização desse dinheiro melhora não só o governo em si, como a vida de toda população, que terá acesso a serviços de melhor qualidade.

6.1. TRABALHOS FUTUROS

Em Relação a trabalhos futuros, tem-se como sugestão:

- *Disponibilização de mais cidades:* Hoje a API DAG Prefeituras disponibiliza dados das cidades de Capivari e São Roque. Futuramente a API irá disponibilizar dados de outras cidades.
- *Tratamento ortográfico dos dados:* Hoje a API captura os dados da forma que estão escritos na origem. Porém muitos desses dados são publicados contendo erros ortográficos, logo uma das propostas de aprimoramento da API, é uma análise e correção dos dados capturados do ponto de vista ortográfico.
- *Padronização de nomenclaturas:* A API DAG Prefeituras precisa passar com um aprimoramento de nomenclaturas para que possa ser implementada de forma mais fácil para outras cidades.
- *Exemplos de utilização:* Apesar de compatível com diversas linguagens de programação, desde que seja compatível com Web Services, a API precisa mostrar exemplos de utilização em mais linguagens de programação. Hoje são mostrados exemplos utilizando Java Script e PHP.
- *Melhorias de Desempenho:* Apesar de apresentar um desempenho satisfatório ao comparar a quantidade de dados capturada, a API precisa ser otimizada para suportar um volume maior de dados.
- *Integração com outras plataformas:* A API DAG Prefeituras poderá ser integrada em outras plataformas que trabalham com DAG, como a desenvolvida pelo Ms. Daniel Ianegitz Vieira.

7. REFERÊNCIAS

ARAÚJO, A. C.; REIS, L.; SAMPAIO, R. C. Do Transparency and Open Data Walk Together. An analysis of Initiatives in five brazilian capitals. **PRETHODNO PRIOPĆENJE**, 2016.

BAI, Q. et al. Analysis and Detection of Bogus Behavior in Web Crawler Measurement. **2nd International Conference on Information Technology and Quantitative Management**, p. 1084-1091, 2014.

BASILI, V. R.; SELBY, R. W.; HUTCHENS, D. H. Experimentation in Software Enginnering. **IEEE Transactions on Software Enginnering**, v. SE-12, July 1986.

BRASIL. LEI Nº 12.527, DE 18 DE NOVEMBRO DE 2011. **REGULA O ACESSO A INFORMAÇÕES**, Brasília, DF, 2011.

BREITMAN, K. et al. Open government data in Brazil. **IEEE Computer Society**, p. 45-49, 2012.

BRITO, K. D. S. et al. Brazilian government open data: Implementation, challenges, and potential opportunities. **Proceedings of the 15th Annual International Conference on Digital Government Research**, p. 11-16, 2014.

BRITO, K. D. S.; COSTA, M. A. D. S.; GARCIA, V. C. Using parliamentary brazilian open data to improve transparency and public participation in Brazil. **Proceedings of the 15th Annual International Conference on Digital Government Research**, p. 171-177, 2014.

CARTILHA. Governo Federal Dados Abertos. Cartilha técnica para publicação de dados abertos no Brasil v1.0, 2016. Disponivel em: <http://dados.gov.br/cartilha-publicacao-dados-abertos/>. Acesso em: 03 December 2016.

CHEN, X. Research and Realization of E-commerce Monitor System Based on Focused Web Crawler. **Information Technology Journal**, p. 4033-4039, 2013.

CORRÊA, A. S.; CORRÊA, P. L. P.; SILVA, F. S. C. D. Transparency portals versus open government data: an assessment of openness in brazilian municipalities. **Proceedings of the 15th Annual International Conference on Digital Government Research**, p. 178–185, 2014.

DADOS. Governo Eletrônico Brasileiro. Dados abertos governamentais, 2016. Disponivel em: <http://www. governoeletronico.gov.br/ acoes-e-projetos/Dados-Abertos>. Acesso em: 03 December 2016.

DATA Viva. Visualizing the economy of Brazil. Disponivel em: <http://dataviva.info>. Acesso em: 18 May 2016.

DAVIS, D. F. Perceived Ease of Use, and User Acceptance of Information Technology. **MIS Quarterly**, September 1989.

DHENAKARAN, S. S.; SAMBANTHAN, K. T. WebCrawler – an Overview. **International Journal of Computer Science and Communication**, p. 265-267, 2011.

DING, ; PERISTERAS, V.; HAUSENBLAS, M. Linked open government data. **IEEE Computer**, 2012.

EUROPEAN. European Innovation Partnership on Smart Cities and Communities. Strategic implementation plan., 2013. Disponivel em: <http://ec.europa.eu/eip/smartcities/files/sip_final_en.pdf. Acessed May 03, 2016.>. Acesso em: 03 May 2016.

JETZEK, T.; AVITAL, M.; BJORN-ANDERSEN, N. Data-Driven Innovation through Open Government Data. **Journal of Theoretical and Applied Eletronic Commerce Research**, p. 100-120, 2014.

KASCHESKY, M.; SELMI, L. 7R Data Value Framework for Open Data in Practice: Fusepool. **Future Internet 2014**, p. 556-583, 2014.

MACHADO, A. L.; OLIVEIRA, J. M. P. D. An open data architecture for egovernment. **15th IEEE International Enterprise Distributed Object Computing Conference Workshops (EDOCW)**, p. 448-456, 2011.

MENDONÇA, P. G. A. D.; MACIEL, C.; VITERBO, J. Visualizing Aedes aegypti infestation in urban areas: A case study on open government data mashups. **Information Polity 20 (2015)**, p. 119-134, 2015.

OPEN. Opengovdata. The 8 principles of open government data., 2016. Disponivel em: <http://opengovdata.org>. Acesso em: 03 December 2016.

OPEN. Open Knowledge Brasil. Projetos, 2016. Disponivel em: <http://br.okfn.org/projetos/>. Acesso em: 18 May 2016.

SALAS, P. et al. Stdtrip: Promoting the reuse of standard vocabularies in open government data. **Linking Government Data**, 2011.

SHARMA, D. K.; SHARMA, A. K. A Novel Architecture for Deep Web Crawler. **International Journal of Information Technology and Web Engineering**, p. 24-48, 2011.

SHEFER, R. P.; ZAINA, L. A. M. **Diretrizes Mobideaf: Uma abordagem para desenvolvimento de aplicações de redes sociais em dispositivos móveis para os surdos**. Dissertação de Mestrado. UFSCar - Sorocaba. Programa de Pós-Graduação em Ciência da Computação. [S.l.]. 2016.

SUAKANTO, S. et al. Smart City Dashboard for Integrating Various Data of Sensor Networks. **IEEE International Conference on ICT for Smart**, 2013.

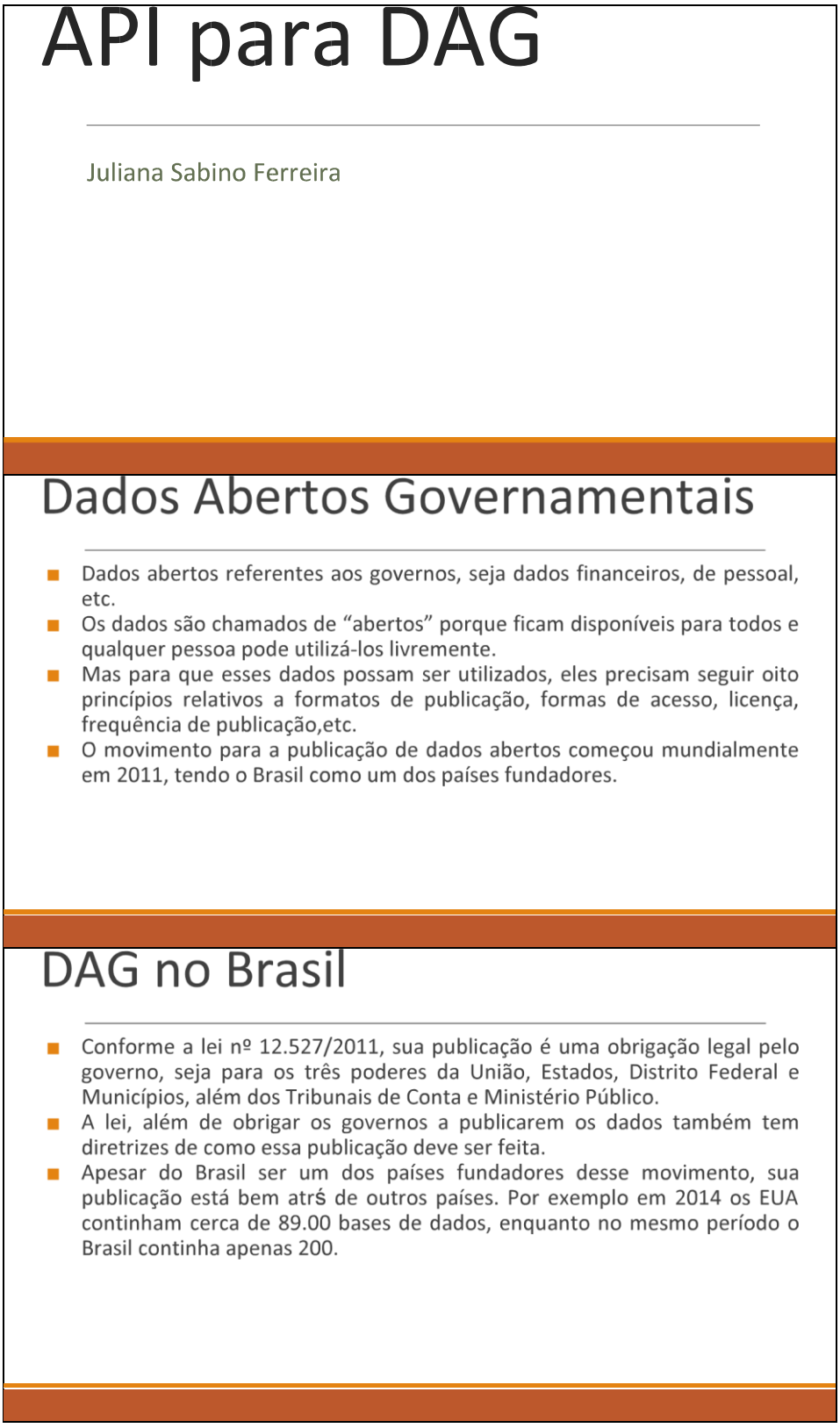
TAN, Q.; MITRA, P. Clustering-based Incremental Web Craling. **ACM Transactions on Information Systems**, p. 1-25, 2010.

TRAVASSOS, G. H.; GUROV, D.; AMARAL, E. A. G. D. **Introdução à Engenharia de Software Experimental**. Rio de Janeiro. 2002.

VIEIRA, D. I.; ALVARO, A. **Uma Plataforma para Disponibilização Centralizada de Dados Abertos Governamentais como Suporte para Aplicações no Contexto de Cidades Inteligentes**. Universidade Federal de São Carlos – Campus Sorocaba. Programa de Pós-Graduação em Ciência da Computação. [S.l.].

APENDICE - A

Materiais Utilizados na Aula



DAG: Formatos Aceitos

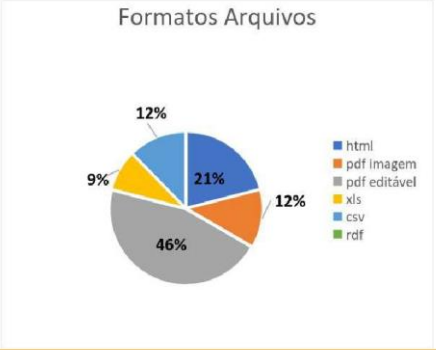
- Para a publicação de dados abertos, os mesmos precisam estar em formatos adequados, não-proprietários, e que possam ser reutilizados. Alguns formatos permitidos são:
  - JSON (Javascript Object Notation)
  - CSV (Comma-Separated Values)
  - ODS (Open Document Spreadsheet) - planilhas, usado no LibreOffice
  - RDF (Resource Description Framework) - modelo estruturado em grafos

DAG: Formatos Aceitos

- Um erro bastante comum é a utilização de arquivos PDF para publicação de dados, já que o PDF, além de ser um formato proprietário, dificulta a leitura dos dados automatizadas (ou seja, através da utilização de software).
- Quando fala-se que um dado aberto precisa estar em um formato que permita reutilização, esse formato precisa ser reconhecido por máquinas e não somente pelo olhar humano.
- O PDF é de fácil compreensão humana, mas para o computador sua leitura é difícil.

Pesquisa Prefeituras

- Foi realizada um pesquisa com 44 prefeituras do estado de São Paulo entre janeiro e março de 2016 nos seus respectivos portais da transparência.
- Conforme mostra o gráfico, a maioria usa formatos PDF e páginas HTML, formatos não previstos para a publicação de DAG



Proposta de Solução

- como solução, nesta pesquisa foi proposta a construção de um Web Crawler para que possa fazer a captura dos dados disponíveis em formatos incorretos e posteriormente a sua publicação em formatos adequados conforme os princípios da DAG.

Web Crawlers

- Software capaz de rastrear páginas na web de maneira metódica e automatizada.
- Ele é caracterizado pela leitura sequencial a partir de uma página HTML e faz a leitura dos dados dessas páginas e das outras URLs contidas nela.

Web Crawlers

- Apesar da capacidade dos Web Crawlers, alguns elementos em uma página podem dificultar a leitura, por exemplo:
  - imagens
  - urls não amigáveis
  - urls ocultas
  - Flash
  - quantidade excessiva de elementos HTML

Web Crawlers

- Para isso, os portais pesquisados foram classificados conforme sua complexidade de leitura por um Web Crawler.
- Os portais foram classificados em: Fácil, médio e Difícil.
- Após essa classificação, como primeiro trabalho foi escolhido os portais das seguintes prefeituras: Capivari e São Roque.
- A escolha foi baseada na complexidade dos portais, que além de baixa, também dispunham o conteúdo somente no formato HTML.

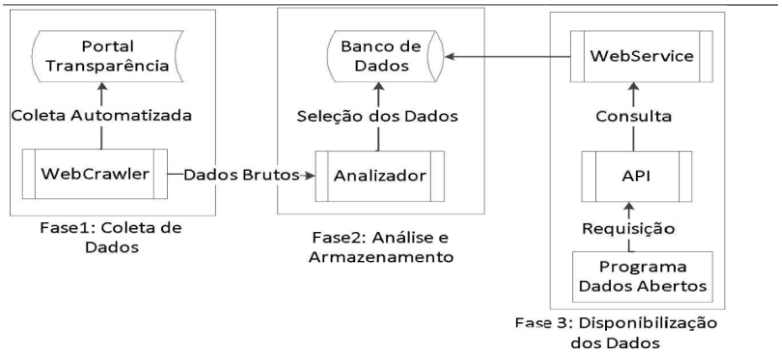
WebCrawler para Prefeituras

- Foi criado um Web Crawler capaz de ler os dados relativos às "Despesas" das duas cidades citadas, Capivari e São Roque.
- Ambas cidades, situadas no interior de São Paulo, utilizam o formato HTML para seus dados relativos à transparência, sem opções de download em outros formatos.
- O Objetivo desse Crawler foi a captura desses dados, armazenamento e sua sua publicação no formato JSON, que é aceito como um formato para DAG.

Web Crawlers para Prefeituras

- Para a construção do Web Crawler foi utilizada a linguagem de programação JAVA, juntamente com as APIs JSOUP e HtmlUnit:
- Ambas APIs foram cominadas para fazer a leitura das páginas e para disparar alguns eventos javascript das mesmas, como a seleção do ano nas páginas, que é feita através da seleção de um componente combobox na página.

Arquitetura



API (Web Service)

- A Api é baseada em um WebService, onde os dados são disponibilizados de forma on-line através de um link.
- A API contém diversas funções para retornar os dados capturados pelo WebCrawler de diversas formas, como descritos a seguir:

API - Funções

Nome Função	Descrição	Parâmetros	Cidades	Observações
gastoTodosDepartamentos	Lista todos departamentos de um determinado ano.	ano tipo (o-orçamento, e-execução)	São Roque Capivari	Em Capivari o nome da função é gasto*TodasSecretarias*
gastoDepartamento	Lista os dados de um departamento de um ano.	ano tipo (o-orçamento, e-execução) departamento	São Roque Capivari	Em Capivari o nome da função é *gastoSecretaria*

API: Funções

Nome Função	Descrição	Parâmetros	Cidades	Observações
gastoDepartamentoDetalhe	Lista os detalhes de um departamento de um ano.	ano tipo (o-orçamento, e-execução) departamento	São Roque Capivari	Em Capivari o nome da função é gastoSecretariaDetalhe
gastoTodasUnidadeOrcamento	Lista os dados resumidos de todas unidades relativas ao orçamento.	ano	São Roque Capivari	

API: Funções

Nome Função	Descrição	Parâmetros	Cidades
gastoUnidadeOrcamento	Lista os dados resumidos de uma unidade relativa ao orçamento.	ano unidade (ex: Fundo municipal de saúde)	São Roque Capivari
gastoFornecedorExecucao	Lista os fornecedores relacionados à execução de um determinado departamento.	ano texto do departamento (ex: Educação)	Capivari
departamentoGastoFornecedor	Mostra os departamentos pertencentes a um fornecedor.	nome do fornecedor	Capivari

API: Funções

Nome Função	Descrição	Parâmetros	Cidades
gastoExeuccaoMes	Mostra os dados gastos de um determinado mês de um ano.	ano mês	São Roque
gastoExecucaoTodosMeses	Mostra os dados gastos por mês de um determinado ano.	ano	São Roque
gastoDepExecucaoMes	Mostra os gastos por departamento de um mês.	ano mês	São Roque
gastoDepDetalheExecucaoMes	Mostra os detalhes de um departamento relativos a um mês.	ano mês nome do departamento	São Roque

Web Service

- Web Services são aplicações que permitem o acesso a dados remotos utilizando diferentes protocolos.
- Existem diversos protocolos usados em Web Services, como:
  - WSDL - Web Service Description Language
  - SOAP - Simple Object Access Protocol
  - UDDI - Universal Description Discovery and Integration
  - REST - Representation State Transfer

Web Service

- Neste trabalho foi utilizado um WebService REST utilizando a linguagem JAVA e o Framework Hibernate.
- A chamada do mesmo consiste na utilização de URLs, como mostrada abaixo:

  http://localhost:8085/WebServiceRest/rest/capivari/gastoTodasSecretarias/2014/o

- Essa URL tem como retorno um objeto do tipo JSON com o resultado da função chamada.

Json

- JavaScript Object Notation
- Formato baseado em texto legível pelo ser humano.
- Facilmente interpretado pelo computador em diversas linguagens de programação.
- Permite a serialização de estrutura de objetos, como listas.

Json

Exemplo:

[{"orcado_inicial":"534.484,00","variacao":"10.012,81","orcado_atualizado":"54.051.386,57","secretaria":"Secretaria Da Educação"}]

Como usar a API

- Para usar a API,é necessário utilizar a url de chamada em seu programa.
- A URL quando colocada em um navegador retorna em tela o objeto json com o resultado da função.
- A API permite ser consumida em diversas linguagens de programação, como JAVA, PHP e Javascript.

Como usar a API

Função: gastoTodasSecretarias

http://localhost:8085/WebServiceRest/rest/capivari/gastoTodasSecretarias/2014/o

2014 - parâmetro ano

o - parâmetro orçamento

gastoTodasSecretarias - nome da função

capivari - nome da cidade (capivari, saoroque)

Como usar a API

Função: gastoTodosDepartamentos

http://localhost:8085/WebServiceRest/rest/saoroque/gastoTodosDepartamentos/2014/o

2014 - parâmetro ano

o - parâmetro orçamento

gastoTodosDepartamentos - nome da função

saoroque - nome da cidade (capivari, saoroque)

Como usar a API

Função: gastoTodasSecretarias

http://localhost:8085/WebServiceRest/rest/saoroque/gastoDepartamento/2014/o/Educação

2014 - parâmetro ano

o - parâmetro orçamento

gastoTadasSecretarias - nome da função

capivari - nome da cidade (capivari, saoroque)

E*ducação* - nome do departamento

Exemplo de chamada da API com Jquery

```
$(document).ready(function() {
   $.ajax({
        type: "GET",
        url: "http://localhost:8085/WebServiceRest/rest/capivari/gastoTodasSecretarias/2014/o",
        contentType: "application/json; charset=utf-8",
        dataType: "json",
         crossDomain: true,
        cache:false,
        async:false,
        success: function (data) {
         $.each(data, function(i, item) {
                  $.each(item, function(j, objeto){
                           $(document.body).append(objeto+'<br>');
                  });
                  $(document.body).append('<br>');
          });
        },
        error: function (xhr, status, error) {
         $(document.body).append(xhr.responseText+ " - " + status );
        }
   });
});
```

Exemplo da chamada da API com PHP

```
<?php
        $response=utf8_encode(file_get_contents('http://localhost:8085/WebServiceRest/rest/capivari/gastoTodasSecretarias/2014/
o'));
        $dados = json_decode($response);
        echo "<b>Secretarias</b><br><br>";

        foreach($dados as $d){
        echo $d->orcado_inicial."<br>";
        echo $d->variacao."<br>";
         echo $d->orcado_atualizado."<br>";
         echo $d->secretaria."<br>";
         echo "<br>";
  }
  ?>
```

Proposta de Desafio

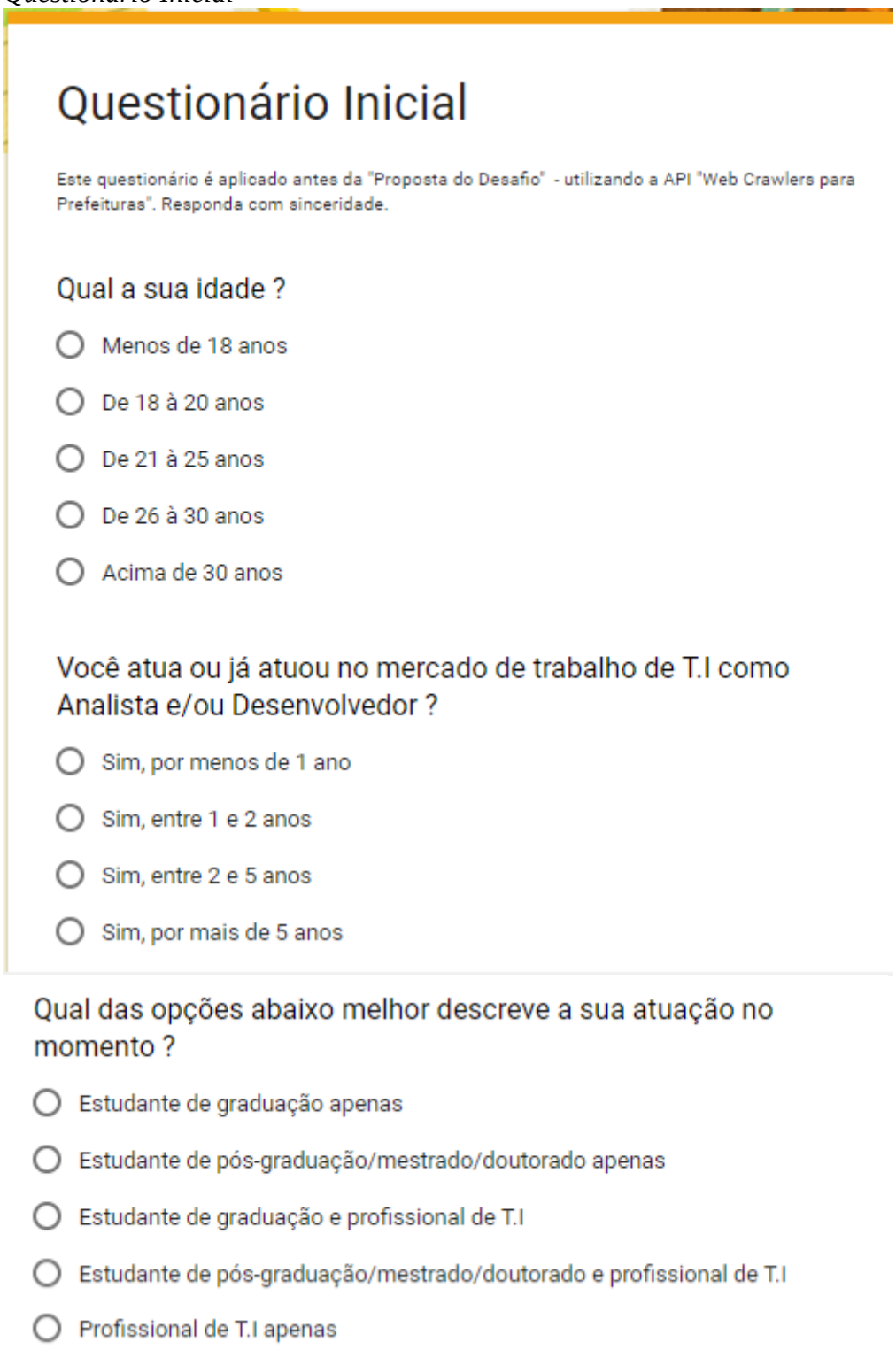
- Crie uma aplicação que mostre em forma de gráfico gastos das prefeituras da seguinte forma:
  - Use os gastos de dois setores em ambas prefeituras.
  - Mostre os gastos do período de 2011 até 2016.
  - Utilize a API proposta nesta validação para captura dos dados.
  - Utilize qualquer linguagem de programação para o desafio. Nesta aula foi mostrado exemplos em duas linguagens: JavaScript (usando o Framework Jquery) e PHP.

Proposta de Desafio

- Dicas
  - Se utilizar Jquery, um exemplo de API para a criação dos gráficos é a Chart.js
  - Independente da implementação, sua solução precisa usar como fonte de dados a API (Web Crawler para Prefeituras).
  - Um exemplo de solução é utilizar gráficos de barras para apresentar os resultados.
  - Coloque o link da API com a chamada da função no navegador. O resultado será o objeto JSON solicitado. Observe a estrutura e adapte-o a sua solução.

APENDICE – B

Questionário Inicial



Questionário Inicial

Este questionário é aplicado antes da "Proposta do Desafio" - utilizando a API "Web Crawlers para Prefeituras". Responda com sinceridade.

Qual a sua idade ?

- Menos de 18 anos
- De 18 à 20 anos
- De 21 à 25 anos
- De 26 à 30 anos
- Acima de 30 anos

Você atua ou já atuou no mercado de trabalho de T.I como Analista e/ou Desenvolvedor ?

- Sim, por menos de 1 ano
- Sim, entre 1 e 2 anos
- Sim, entre 2 e 5 anos
- Sim, por mais de 5 anos

Qual das opções abaixo melhor descreve a sua atuação no momento ?

- Estudante de graduação apenas
- Estudante de pós-graduação/mestrado/doutorado apenas
- Estudante de graduação e profissional de T.I
- Estudante de pós-graduação/mestrado/doutorado e profissional de T.I
- Profissional de T.I apenas

Aponte seu nível de experiência nas tecnologias abaixo

	Não possuo conhecimento	Possuo conhecimento acadêmico apenas	Possuo conhecimento acadêmico e prático apenas	Possuo conhecimento acadêmico, teórico e prático	Já apliquei o conhecimento em projetos do mercado de trabalho
Java	○	○	○	○	○
Web Service	○	○	○	○	○
Banco de Dados	○	○	○	○	○
Orientação à Objetos	○	○	○	○	○
Json	○	○	○	○	○
JavaScript	○	○	○	○	○
Jquery	○	○	○	○	○
PHP	○	○	○	○	○
JSP/Servlet	○	○	○	○	○
Web Crawlers	○	○	○	○	○

Qual seu conhecimento em relação à Dados Abertos Governamentais

	1	2	3	4	5	
Nenhum	○	○	○	○	○	Muito

Você costuma verificar sites que disponibilizam Dados Abertos Governamentais (Dados referentes à Transparência do Governo) ?

○ Sim

○ Não

Na sua opinião, aponte a importância da publicação de dados abertos governamentais

	Discordo Totalmente	Discordo Parcialmente	Não concordo e nem discordo	Concordo parcialmente	Corcordo totalmente
Aumentar a participação política da população	○	○	○	○	○
Combater a corrupção	○	○	○	○	○
Mostrar a gerência do governo	○	○	○	○	○
Ajudar no desenvolvimento de novas aplicações para ajudar a população	○	○	○	○	○
Ajudar no planejamento estratégico	○	○	○	○	○

Questionário Final

Questionário Final

Este questionário deve ser aplicado ao participante após o término do desafio. Analise sua solução desenvolvida e responda com sinceridade as questões abaixo:

A sua solução utilizou quantas linhas de código ?

- ○ Menos de 10 linhas
- ○ De 10 à 50 linhas
- ○ de 51 à 100 linhas
- ○ de 101 à 200 linhas
- ○ de 201 à 300 linhas
- ○ de 301 à 400 linhas
- ○ de 401 à 500 linhas
- ○ mais de 500 linhas

Quanto tempo sua solução demorou para ser finalizada ?

- ○ Menos de 30 minutos
- ○ De 30 minutos à 1 hora
- ○ De 1 a 2 horas
- ○ De 2h1min à 3h
- ○ Mais de 3 horas

Quais foram suas principais dificuldades no desafio ?

- [ ] Uso da API
- [ ] Entendimento sobre Dados Abertos Governamentais
- [ ] Trabalhar com o formato JSON
- [ ] Criação de Gráficos
- [ ] Linguagem de programação
- [ ] Outro:

Descreva críticas e/ou sugestões para API

Sua resposta

Marque as tecnologias utilizadas por você para resolver o desafio proposto

- [ ] Java
- [ ] JavaScript
- [ ] Jquery
- [ ] PHP
- [ ] JSP
- [ ] Outro:

Informe qual API para criação de gráficos você utilizou no seu trabalho (ex: Chart.js, JS Charts, etc.)

Sua resposta

Marque seu grau de satisfação da facilidade de uso da API para a realização do desafio

	1	2	3	4	5	6	7	8	9	10	
Muito insatisfeito	○	○	○	○	○	○	○	○	○	○	Muito satisfeito

Marque na sua visão a utilidade da API no contexto de "Dados Abertos Governamentais"

	1	2	3	4	5	
Pouco útil	○	○	○	○	○	Muito útil