
Curso de Ciência da Computação
Universidade Estadual de Mato Grosso do Sul

Busca Livro: Um Sistema de Buscas Efetivas em Acervos Bibliográficos

Rodolpho Pivetta Sabino
Rogers Prates de Pelle

Prof. Dra. Glaucia Gabriel Sass (Orientadora)
Me. Gabriel Dieterich Cavalcante (Co-orientador)

Dourados - MS
2015

Busca Livro: Um Sistema de Buscas Efetivas em Acervos Bibliográficos

Rodolpho Pivetta Sabino

Rogers Prates de Pelle

Trabalho de Conclusão de Curso para obtenção do Diploma de Bacharel em Ciência da Computação, na Área de Ciências Exatas e da Terra da UEMS.

Dourados, 30 de novembro de 2015

Prof. Dra. Glaucia Gabriel Sass
(Orientadora)

Me. Gabriel Dieterich Cavalcante
(Co-orientador)

Busca Livro: Um Sistema de Buscas Efetivas em Acervos Bibliográficos

Rodolpho Pivetta Sabino
Rogers Prates de Pelle

Novembro de 2015

Banca Examinadora:

Prof. Dra. Glaucia Gabriel Sass (Orientadora)
Área de Computação - UEMS

Prof. Dr. André Chastel Lima
Área de Computação - UEMS

Prof. Dr. Felipe José Carbone
Área de Computação - UFGD

Dedico este trabalho aos meus pais, irmãs e aos meus avós que mesmo sem entenderem o que faço, me apoiam e incentivam.

Rodolpho Pivetta Sabino

Dedico este trabalho aos meus pais, Carlos Alberto de Pelle e Márcia Aparecida Prates, que tiveram que deixar seus sonhos de lado, para que eu realizasse o meu.

Rogers Prates de Pelle

*“A tarefa não é tanto ver aquilo que ninguém viu, mas pensar o que ninguém ainda
pensou sobre aquilo que todo mundo vê.”
(Arthur Schopenhauer)*

AGRADECIMENTOS

A Deus pela vida e força pra segui-la.

A minha família, em especial minha mãe Ivete Maria Pivetta e a Carolini Casarini Cardoso por acreditarem e me darem forças para seguir essa caminhada.

Aos meus orientadores Glaucia Sass e Gabriel Cavalcante pela paciência, ajuda, motivação e conhecimento transmitido durante todo o processo de elaboração deste trabalho, cumprindo seus papéis de forma impecável.

A Universidade por me proporcionar essa e tantas outras oportunidades em minha carreira, e seu corpo docente por transmitir seu conhecimento da melhor forma possível.

Aos meus amigos e colegas que cruzaram meu caminho durante essa jornada, fazendo com que ele fosse melhor e mais alegre de ser seguido.

A todos que direta ou indiretamente influenciaram para que este trabalho se tornasse real.

Rodolpho Pivetta Sabino

A Deus pela vida e por me conceder a oportunidade de chegar a universidade, mesmo vindo de uma família onde quase ninguém teve essa chance.

A minha namorada Geisiane Martini, minha família e aos meus amigos, que deram apoio e motivação para seguir em frente e nunca desistir.

Aos meus orientadores Glaucia Gabriel Sass e Gabriel Cavalcante, que não apenas cumpriram seus papéis, mas foram além e se dedicaram ao máximo para que este trabalho fosse concluído da melhor forma possível.

A todos os professores, desde o primário até a universidade, pois cada conhecimento compartilhado foi de fundamental importância para chegar até aqui.

Rogers Prates de Pelle

RESUMO

A utilização de livros é de fundamental importância para todo acadêmico, e os usuários das bibliotecas da UEMS e UFGD tem encontrado dificuldades em utilizar os sistemas disponibilizados para realizar buscas nos acervos. Este trabalho se propõe a criar um sistema de buscas nos acervos bibliográficos das universidades UEMS e UFGD, de modo que o usuário possa realizar buscas de forma unificada e com melhores resultados. O sistema Busca Livro é composto por: (i) um mecanismo de obtenção de dados, implementado utilizando conceitos de *Web Scraping*; (ii) um servidor de busca e indexador de dados baseado em técnicas e ferramentas para otimização de busca textual; (iii) duas interfaces para acesso de dados, sendo uma Web e outra móvel para dispositivos Android. Para obter os dados de todos os livros contidos nos acervos, foram implementados algoritmos que capturam e selecionam os dados dos sistemas das bibliotecas. Esses dados são enviados para o servidor de buscas (i) que os indexa e armazena, o motor de buscas (ii) utilizado permite apenas consultas de forma estruturada, no formato de *queries*, por isso foi desenvolvida uma API para converter pesquisas realizadas em *queries* válidas. A interface Web (iii) permite ao usuário realizar buscas, que são enviadas para o servidor por meio de requisições. O aplicativo para dispositivos Android (iii) tem os mesmos recursos e funciona de forma análoga a interface Web. Este trabalho apresenta resultados de testes de desempenho e de precisão das buscas, comparando o Busca Livro e os outros dois sistemas. São apresentados também os resultados das pesquisas aplicadas à usuários em relação a usabilidade e eficiência do sistema. Com o Busca Livro é possível realizar buscas de forma mais efetiva nos acervos das bibliotecas e verificar os dados, localização e disponibilidade dos itens dos acervos por meio de interfaces amigáveis e intuitivas.

Palavras-chave: *Web Scraping*. Motores de Busca. Sistema de Informação Web. Sistema de Informação móvel.

ABSTRACT

The use of books is of fundamental importance to all academic students, and the user of the UEMS and UFGD libraries have found it difficult to use the systems available for performing searches in collections. This study aims to create a search system in the library collections of the UEMS and UFGD universities, so that the user can perform searches in a unified way and with better results. The Busca Livro system consists of: (i) a mechanism for data collection, implemented using concepts of Web Scraping; (ii) a search server and data index based on techniques and tools for textual search engine optimization; (iii) two interfaces for data access, one Web and another mobile for Android devices. For the data of all the books contained in the collections were implemented algorithms that capture and select data from the library systems. This data is sent to the search server (i) that indexes and stores, the search engine (ii) used to allow only structured queries, the query format, so it was developed an API to convert research in queries valid. The Web interface (iii) allows the user to perform searches, which are sent to the search server through requests. The application for Android devices (iii) has the same features and functions analogously to the Web interface. This paper presents performance test results and accuracy of searches, comparing the Busca Livro and the other two systems. Also presents the results of research applied to users regarding usability and system efficiency. With Busca Livro can conduct searches more effectively in the collections of libraries and verify the data, location and availability of items from the collections through friendly and intuitive interfaces.

Keywords: Web Scraping. Search Engines. Web Information System. Mobile Information System.

SUMÁRIO

1	INTRODUÇÃO	23
1.1	Objetivos	24
1.2	Estrutura do Trabalho	24
2	REFERENCIAIS TEÓRICOS	25
2.1	World Wide Web	25
2.2	Tecnologias para o desenvolvimento de um Sistema de Informação Web e móvel	27
2.3	Trabalhos Relacionados	32
3	MATERIAIS E MÉTODOS	35
4	RESULTADOS	39
4.1	Base de dados do Busca Livro	39
4.2	Web Scraping da UEMS	40
4.3	Web Scraping da UFGD	44
4.4	Armazenamento e indexação das informações capturadas	45
4.5	Execução do sistema de obtenção dos arquivos	49
4.6	API de requisições	49
4.7	Interfaces de Usuário	52
4.8	Análise dos resultados	55
5	CONCLUSÃO	61
5.1	Sugestões e Recomendações	62
	REFERÊNCIAS	63
	APÊNDICES	65
	APÊNDICE A – QUESTIONÁRIO DE AVALIAÇÃO	67
	APÊNDICE B – RESPOSTAS DO QUESTIONÁRIO	69
	APÊNDICE C – TESTES DE INTERFACE PARA DISPOSITIVOS MÓVEIS	71

LISTA DE ABREVIATURAS E SIGLAS

AJAX	Asynchronous Javascript and XML
API	Application Program Interface
BD	Banco de Dados
BSD	Berkeley Software Distribution
CRC	Class Responsibility Card
CWI	Centrum Wiskunde & Informatica
DO	Digital Ocean
DRY	Don't Repeat Yourself
GCM	Goal Question Metric
HTML	Hypertext Markup Language
HTTP	Hypertext Transfer Protocol
IDE	Integrated Development Environment
JVM	Java Virtual Machine
LASCA	Laboratório de Segurança e Criptografia
MTV	Model Template View
REST	Representational State Transfer
SGBD	Sistema Gerenciador de Banco de Dados
SI	Sistema de Informação
SQL	Structured Query Language
UEMS	Universidade Estadual do Mato Grosso do Sul
UFGD	Universidade Federal da Grande Dourados
UNICAMP	Universidade Estadual de Campinas
URL	Uniform Resource Locator

USP	Universidade de São Paulo
VM	Virtual Machine
VPS	Virtual Private Server
W3	World Wide Web
W3C	World Wide Web Consortium

LISTA DE ILUSTRAÇÕES

Figura 1 – Modelo cliente-servidor	25
Figura 2 – Representação do paradigma REST	27
Figura 3 – Processo da XP	30
Figura 4 – Telas: Sistema Integrado de Bibliotecas SIBi	33
Figura 5 – Estrutura de camadas do sistema Busca Livro	35
Figura 6 – Esquema de funcionamento do Django	37
Figura 7 – Modelo Entidade Relacionamento	40
Figura 8 – Página de busca da Biblioteca da UEMS	41
Figura 9 – Página de informações dos itens do acervo	41
Figura 10 – Página de informações com Código inválido	42
Figura 11 – Página de informações com Código válido	42
Figura 12 – Fluxograma do Web Scraping da UEMS	43
Figura 13 – Árvore de combinações de buscas do <i>Web Scraping</i> da UFGD	45
Figura 14 – Fluxogramas de execução do Web Scraping da UFGD	46
Figura 15 – Casos de uso das interfaces	52
Figura 16 – Protótipo das Telas	53
Figura 17 – Interface Web: Página Inicial do sistema	53
Figura 18 – Interface Web: Tela de resultados	54
Figura 19 – Interface Web: <i>Modal</i> com informações detalhadas	54
Figura 20 – Telas da interface móvel	55
Figura 21 – Interfaces Web dos sistemas Busca Livro, UEMS e UFGD	56
Figura 22 – Interfaces Web dos sistemas Busca Livro, UEMS e UFGD	56
Figura 23 – Interfaces Web dos sistemas Busca Livro, UEMS e UFGD	57
Figura 24 – Resultados: Área do curso apontada pelos usuários	59
Figura 25 – Resultados: Indisponibilidade dos sistemas apontada pelos usuários	59
Figura 26 – Resultados: Informações mais relevantes apontadas pelos usuários	60
Figura 27 – Resultados: Vantagens apontadas pelos usuários	60
Figura 28 – Teste de interface para dispositivos móveis - Busca Livro	71
Figura 29 – Teste de interface para dispositivos móveis - UEMS	72
Figura 30 – Teste de interface para dispositivos móveis - UFGD	73

LISTA DE TABELAS

Tabela 1 – Versões das Plataformas Android	31
Tabela 2 – Comparação das especificações dos servidores utilizados	49
Tabela 3 – Comparativo do Teste de Desempenho dos serviços de buscas	58
Tabela 4 – Respostas do questionário	69

1 INTRODUÇÃO

A utilização de livros é de fundamental importância no processo de aprendizagem de todo estudante, sendo assim, os acadêmicos necessitam consultar frequentemente os acervos das universidades em busca de materiais para estudo. Na Cidade Universitária de Dourados - MS, onde se encontram a Universidade Estadual do Mato Grosso do Sul (UEMS) e a Universidade Federal da Grande Dourados (UFGD), existe um acordo bilateral que garante aos alunos o privilégio de utilizar as duas bibliotecas, fazendo com que seus acervos se complementem.

Entretanto, atualmente não é possível fazer uma busca nestes acervos de forma unificada. Isto torna cansativo o processo de encontrar material relevante, pois o aluno precisa fazer a busca em sistemas distintos. A indisponibilidade de um sistema unificado bem como a falta de uma interface apropriada ao uso em dispositivos móveis também são problemas enfrentados diariamente pelos usuários das bibliotecas.

Uma solução para esses problemas pode ser encontrada no uso de tecnologias que permitam o desenvolvimento de um sistema de busca mais eficiente, como técnicas de captura de dados (*Scrapings* e *Crawlers*), indexação de dados com alta performance em servidores de busca e *big data* no armazenamento massivo e variado de informações. A utilização destes recursos permite a criação de repositórios de informação bibliográfica adequados ao uso de poderosos motores de busca. A partir dessas informações corretamente capturadas e armazenadas, torna-se viável a criação de interfaces entre dispositivos e pessoas. Essas interfaces podem disponibilizar esses dados de maneira clara através de aplicações Web e móvel, facilitando a disponibilidade e usabilidade do serviço.

Desse modo, a proposta deste trabalho compreende o desenvolvimento de um Sistema de Informação (SI), com interfaces Web e móvel, para buscas nos acervos da UEMS e UFGD de forma unificada, e com resultados relevantes, permitindo aos usuários a economia de tempo e a possibilidade de comparação dos itens apresentados. Esse sistema é composto de um mecanismo de coleta de dados, um servidor de buscas e dois aplicativos para realização de buscas, um para navegadores Web e a outro para dispositivos móveis, ambos com interfaces amigáveis.

1.1 Objetivos

Desenvolver um SI com interfaces Web e móvel para a realização de buscas nos acervos bibliográficos da UEMS e UFGD.

Objetivos Específicos

- I Estudo teórico dos conceitos relacionados ao desenvolvimento de ferramentas de busca, e aplicações Web e móvel.
- II Implementação de um mecanismo de coleta de dados que, sistematicamente atualiza e indexa os acervos das bibliotecas da UEMS e UFGD.
- III Implantação de um servidor de buscas que retorne resultados confiáveis e condizentes com as buscas realizadas.
- IV Desenvolvimento de um aplicativo de buscas para navegadores Web com interface responsiva.
- V Desenvolvimento de um aplicativo de buscas para dispositivos com sistema operacional Android.

1.2 Estrutura do Trabalho

O capítulo 1: Introdução, apresenta uma breve descrição do trabalho, bem como a motivação, justificativa e os objetivos gerais e específicos para o desenvolvimento deste trabalho. No Capítulo 2: Referenciais Teóricos, são abordados os conceitos teóricos sobre todas as tecnologias no qual o trabalho é baseado. No Capítulo 3: Materiais e Métodos, são descritas as ferramentas utilizadas para o desenvolvimento do sistema de buscas Busca Livro e suas funcionalidades. No capítulo 4: Resultados, são apresentadas as formas de implementação e as soluções de problemas encontrados no decorrer do trabalho. No capítulo 5: Conclusão, é realizada uma reflexão a fim de comparar a proposta do trabalho e o resultados apresentados.

2 REFERENCIAIS TEÓRICOS

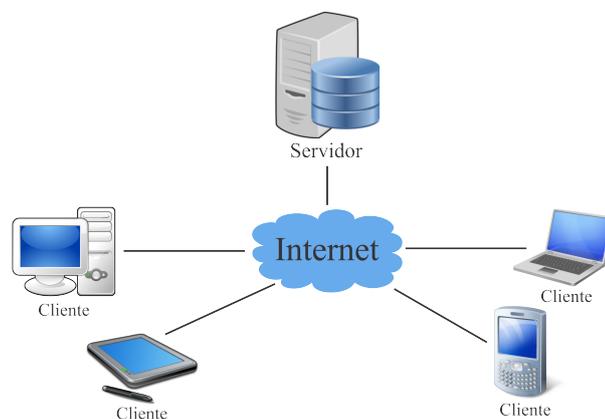
Neste capítulo são apresentados os estudos teóricos que serviram de base para o desenvolvimento do sistema Busca Livro. São abordadas a Internet e tudo que torna sua execução possível, ferramentas de busca, bancos e indexadores de dados, técnicas de captura de dados como *Web Scraping* e *Crawler*, as linguagens de programação Python e Java e o sistema operacional Android.

2.1 World Wide Web

A *World Wide Web* (W3), ou W3, é um sistema de compartilhamento de informação através de documentos em hipermídia, onde todos esses documentos estão de alguma forma interligados. A W3 é organizada e padronizada pelo *World Wide Web Consortium* (W3C). Toda a estrutura lógica da W3 só pode existir graças a estrutura física da Internet (WORLD WIDE WEB, 1989).

Kurose e Ross (2010) definem a Internet como uma rede de computadores mundial, que interconecta milhões de dispositivos em todo o mundo através de outras redes menores utilizando um conjunto de protocolos. Esses protocolos provêm acesso universal para a transmissão de dados que são armazenados em máquinas conectadas a rede. A Internet está baseada no modelo cliente-servidor, demonstrado na 1, onde um cliente, que é um sistema final qualquer, que pode realizar uma requisição de algum dado ou serviço, e o servidor é responsável pelo provimento da informação transmitindo-a para o cliente. Desde sua criação em 1969 até meados da década de 1980 toda essa estrutura era utilizada apenas para o compartilhamento de dados, organizados em diretórios.

Figura 1 – Modelo cliente-servidor



Fonte: Elaborada pelos autores

A Web como a conhecemos teve início em 1989, conceituada por Tim Burners-Lee. Era uma Web só de leitura, ou seja, as informações eram disponibilizadas estaticamente, sem interação com seus usuários. A evolução dessa Web deu origem a Web 2.0, definida em 2004 por Dale Dougherty, centrada em usuários. Algumas ações dos usuários são facilitadas nesta nova Web, como leitura e escrita, flexibilidade de *design*, construção de conteúdo colaborativo, atualizações e modificações (AGHAEI; NEMATBAKSH; FARSANI, 2012).

Os documentos em texto e as hipermídias da Web são disponibilizados em páginas desenvolvidas em linguagem de marcação. A principal linguagem de marcação usada para criar documentos na Web é a *Hypertext Markup Language* (HTML). Essa linguagem é definida basicamente por *tags* (marcações) que definem o *layout* e estrutura de um documento que é interpretado e exibido por um navegador de Internet (SHELLY; WOODS; DORIN, 2008).

A maioria do conteúdo que trafega pela Web é transferido através do *Hypertext Transfer Protocol* (HTTP), um protocolo que executa transações na W3. Ele possui muitas aplicações, mas as mais conhecidas são em navegadores Web e servidores Web, podendo ser simplificada como uma linguagem moderna da Internet. Quando um usuário utiliza um navegador e faz uma solicitação de alguma página na Internet, este navegador entra em contato com o servidor de tal página através do protocolo HTTP (GOURLEY et al., 2002).

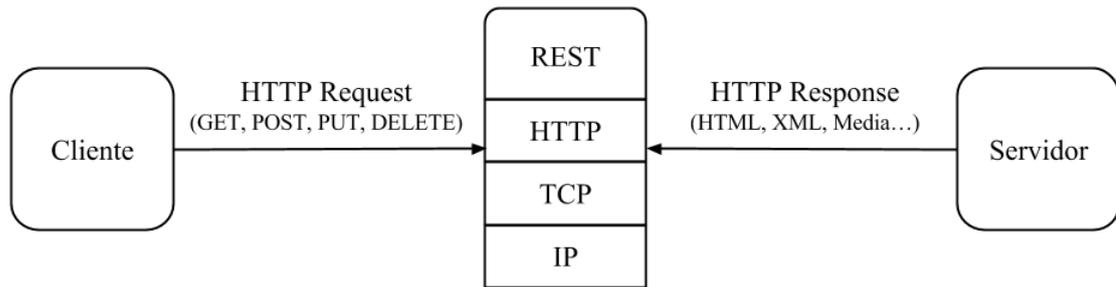
Para cada requisição, como um clique em um *link* ou submissões de formulários, uma chamada ao protocolo é realizada. Este protocolo possui muitos métodos de envio e recebimento de informações, que foram sendo incluídos de acordo com o tempo de desenvolvimento no decorrer dos anos (GOURLEY et al., 2002).

Uma das formas de utilização do protocolo é através do *Representational State Transfer* (REST), que é um paradigma arquitetônico idealizado por Roy Fielding, um dos principais autores do HTTP 1.0 e 1.1. A arquitetura REST promove um ciclo de mensagens (requisições e respostas) entre clientes e servidores, e usa os métodos *GET*, *POST*, *PUT* e *DELETE* para isto, representado na Figura 2 (HILL et al., 2012).

Existem serviços Web que utilizam essa arquitetura, uma delas é *RESTful*, que basicamente utiliza as quatro operações básicas para facilitar o envio e recebimento de dados. “Receber algum dado” (*GET* e *POST*), “Deletar algum dado” (*DELETE*) e “Substituir com algum dado” (*PUT*) são os métodos mais utilizados (RICHARDSON; RUBY, 2008).

O objetivo dos métodos *GET* e *POST* são basicamente os mesmos, realizar uma requisição passando como parâmetro algumas informações que serão processadas e recebendo dados correspondentes a essas informações. A diferença entre esses métodos é que o método *GET* transmite seus parâmetros na *Uniform Resource Locator* (URL)

Figura 2 – Representação do paradigma REST



Fonte: Elaborada pelos autores

da requisição, enquanto o método *POST* transmite esses dados de forma implícita no cabeçalho do pacote de requisição (RICHARDSON; RUBY, 2008).

A Internet foi criada com o objetivo de compartilhar dados que antes estavam isolados em repositórios. Com o avanço e a popularização da Internet, o volume de dados aumentou, e houve a necessidade de criar instrumentos para localizar informações dentro desses repositórios. Nesse contexto nasceram as ferramentas de busca. Essas podem ser definidas como um software que busca por palavras-chave fornecidas por usuários em um repositório de dados, podendo ou não retornar uma lista de referências que, de alguma forma, se relacionam com algum critério estabelecido (CENDÓN, 2001).

“Existem dois tipos básicos de ferramentas de busca na *Web*: os motores de busca e os diretórios. Entretanto, a partir dessas duas categorias básicas, outros tipos de ferramentas têm surgido, fazendo o mundo dos serviços de busca complexo e volátil.” (CENDÓN, 2001, p. 39).

2.2 Tecnologias para o desenvolvimento de um Sistema de Informação Web e móvel

O'Brien (2004) apresenta uma estrutura conceitual para organizar as áreas de conhecimento de um SI, entre elas a área de Tecnologia da Informação. Nessa área o SI é apresentado com uma estrutura composta de *hardware*, *software*, redes, gerência de dados e várias tecnologias baseadas na Internet.

Esse SI que utiliza as tecnologias da Internet permite “transferir” as tradicionais características dos SIs para a Web. Assim, surgiram os Sistemas de Informação Web que armazenam, gerenciam e disponibilizam dados aos usuários. Os SIs Web podem oferecer seus serviços em computadores conectados a Internet ou em dispositivos móveis como *smartphones* e *tablets*, independente de plataforma e sistema operacional. Esse tipo de SI,

assim como grande parte da Internet, é operado no modelo cliente-servidor (O'BRIEN, 2004).

A função principal do SIs Web é fornecer serviços para os usuários. Um dos serviços mais utilizados atualmente é o de busca, fornecido pelos Sistemas de Busca, pois através dele é possível alcançar variados tipos de conteúdo e até mesmo outros serviços. Os sistemas de busca podem ser aplicados em diversos cenários, auxiliando na recuperação de dados de forma mais simples e rápida, podendo inclusive ser integrado a sistemas existentes. Na implementação de um sistema de busca é de fundamental importância a utilização dos motores de busca (O'BRIEN, 2004).

“Todos os motores atuais utilizam o método de robôs sendo formado por quatro componentes: um robô, que localiza e busca documentos na Web; um indexador, que extrai a informação dos documentos e constrói a base de dados; o motor de busca propriamente dito; a interface, que é utilizada pelos usuários.” (CENDÓN, 2001, p. 41)

Uma das técnicas de construção de robôs para motores de busca é a *Web Scraping*, que consiste em um processo de capturar informações desestruturadas da Web e transformá-las em informações estruturadas e possíveis de serem armazenadas e analisadas. Essa técnica pode se assimilar com o conceito de *Web Crawler*, porém cada termo corresponde a uma forma diferente de se obter dados (HANRETTY, 2013).

Aplicações dos conceitos de *Web Scraping* são bastante abrangentes, sendo largamente utilizados no interesse, por exemplo, de empresas em monitoramentos de preços e tecnologias. Muitas vezes não existe uma *Application Program Interface* (API) que disponibilize as informações desejadas, sendo necessária a utilização de uma ferramenta criada a partir dos conceitos de *Web Scraping* (CORDING, 2011).

Os documentos encontrados são processados pelos indexadores, que extraem as informações julgadas importantes, como títulos, resumos e palavras contidas das páginas. O processo de extração normalmente é realizado pelo *parser*, que consiste em um analisador que, a partir dos dados de entrada, constrói uma estrutura de dados, geralmente uma árvore, onde é possível selecionar as informações. Após a extração, os dados são classificados, ordenados por meio de índices e armazenados em uma base de dados (CENDÓN, 2001).

As bases de dados ou Bancos de Dados (BDs) são blocos de arquivos que armazenam dados de forma organizada, com o objetivo de agilizar a pesquisa. Desde sua criação, são peças fundamentais dos SIs. As operações dos BDs são realizadas pelos Sistemas Gerenciadores de Bancos de Dados (SGBDs). O SGBD relacional é o mais comumente utilizado, e suas operações são realizadas através de consultas escritas em *Structured Query Language* (SQL). Existem ainda outros tipos de SGBDs como os Orientado a Objetos e os não-relacionais (SILBERSCHATZ; KORTH; SUDARSHAN, 2012).

Esses tradicionais SGBDs por muito tempo gerenciaram um volume limitado e controlado de dados. O avanço tecnológico dos últimos anos resultou em um aumento massivo do volume desses dados, gerando a necessidade de serviços escaláveis e implicando no surgimento da computação em nuvem. Com isso tornou-se necessária a criação de BDs que fossem mais ágeis e atendessem os requisitos dessas novas tecnologias, nesse contexto surgiram os Bancos de Dados não-relacionais (*No-SQL*). Nesta tecnologia os BDs não exigem esquemas de tabelas fixas e na maioria das vezes não dão suporte a SQL. *No-SQL* é uma tecnologia comumente utilizada em motores de busca devido a seu alto desempenho e seus recursos de indexação (SADALAGE; FOWLER, 2013).

A cada pesquisa realizada, o motor de busca procura em seus índices documentos que contenham os termos buscados ou similares para retornar apenas referências que se relacionem com esses termos. De posse dos resultados, o indexador precisa ordená-los, e é neste momento que ele aplica operadores relacionais (*AND*, *NOT*, *NEAR* e *SAME*) e outras técnicas, para que o resultado da pesquisa seja relevante ao usuário. Um exemplo desta classificação, é o *PageRank* da Google que avalia e ordena as páginas pela sua importância e pelos links que apontam para ela (CENDÓN, 2001).

Existem atualmente vários motores de buscas, bibliotecas e APIs, alguns proprietários e outros de código fonte aberto. Desses podemos destacar o Solr, Nutch e Elasticsearch, baseados na API Lucene, na biblioteca Xapian e no OpenSearchServer. Todos estes recursos oferecem interfaces de comunicação com um grande número de linguagens de programação. As técnicas utilizadas na implementação dos motores de busca podem ser desenvolvidas em quase todas as linguagens de programação, entretanto algumas linguagens possuem recursos que melhoram o desempenho e facilitam a implementação.

O desenvolvimento de um SI se inicia com a fase de planejamento e a escolha de um processo que atenda as necessidades dos requisitos do projeto e se adeque a realidade da equipe de desenvolvimento. Quando o SI a ser desenvolvido é de pequeno ou médio porte, o processo de desenvolvimento escolhido geralmente aborda uma metodologia ágil (PRESSMAN, 2009).

“A Extreme Programming (XP) é uma metodologia ágil para equipes pequenas e médias que desenvolvem software baseado em requisitos vagos e que se modificam rapidamente [...] XP enfatiza o desenvolvimento rápido do projeto e visa garantir a satisfação do cliente, além de favorecer o cumprimento das estimativas. As regras, práticas e valores da XP proporcionam um agradável ambiente de desenvolvimento de software para os seus seguidores, que são conduzidos por quatro valores: comunicação, simplicidade, *feedback* e coragem.” (SOARES, 2004, p. 3).

O processo de desenvolvimento utilizando a metodologia XP é dividido em 4 fases: planejamento, projeto, codificação e teste. A Figura 3 ilustra o processo XP e mostra algumas das tarefas que são associadas a cada fase. Na fase de planejamento, cliente

e desenvolvedores trabalham juntos a fim de levantar os requisitos do sistema. Na fase de projeto são produzidos cartões CRC e soluções pontuais, formando um protótipo do projeto. A codificação é geralmente realizada no conceito de programação dupla, conforme cada dupla finaliza sua parte, ela é integrada ao trabalho dos outros, nesta fase acontece também a refabricação. A fase de teste deve ser executada com uma unidade de testes automatizados, e após isso, são executados os testes de aceitação (PRESSMAN, 2009).

Figura 3 – Processo da XP



Fonte: (PRESSMAN, 2009)

Para a implementação de SI Web existem várias linguagens de programação como PHP, Javascript, C# .NET, Python e Java. Dentre essas as que são mais utilizadas pelos desenvolvedores são PHP e Python.¹

O Python é uma linguagem de programação de alto nível, interpretada, orientada a objetos e de tipagem dinâmica e forte. Ela permite associar diferentes comportamentos e características com os dados que são manipulados. Foi criada em 1990 por Guido van Rossum no *Centrum Wiskunde & Informatica* (CWI) e é considerada também uma linguagem multi paradigmas, podendo ser adaptada desde linguagens orientadas a objetos como Java ou Ruby, até linguagens funcionais como Lisp ou Haskell (BORGES, 2014).

O Java é uma linguagem de programação orientada a objetos. A versão 1.0 foi lançado em 1995 pela Sun Microsystems, Inc., criada por James Gosling, Patrick Naughton, Chris Warth, Ed Frank e Mike Sheridan. O processo de compilação é diferentes de outras linguagens, o Java é compilado para um *bytecode* que é executado por uma máquina

¹ Fonte: <<http://redmonk.com/sogrody/2015/01/14/language-rankings-1-15>>

virtual, tornando possível sua execução em qualquer plataforma que tenha uma *Java Virtual Machine* (JVM). Sua popularidade entre os desenvolvedores é tão grande que é a segunda linguagem mais utilizada do mundo, e a principal linguagem para desenvolvimento de aplicativos para dispositivos móveis com sistema operacional Android (SCHILDT, 2015).

O sistema operacional Android é uma plataforma de código fonte aberto, projetada para dispositivos móveis. Propriedade da *Open Handset Alliance*, essa plataforma é mantida pela Google. Criado por Andy Rubin inicialmente para utilização em telefones, que posteriormente vendeu sua propriedade e se tornou Diretor de Plataformas móveis para a Google. O desenvolvimento desta tecnologia, juntamente com outras plataformas como Apple iPhone, RIM Blackberry, Nokia Symbian e Windows Phone da Microsoft trouxeram grandes evoluções, transformando este espaço emergente na Internet 2.0, que permite aos utilizadores acessarem dados à partir de qualquer plataforma através de uma variada gama de dispositivos eletrônicos como *tablets* e *smartphones* (GARGENTA; NAKAMURA, 2014).

Hoje, a plataforma Android é utilizada em centenas de milhões de dispositivos espalhados por mais de 190 países. Devido às contribuições da comunidade Linux e suas parceiras, o Android cresceu com grande velocidade como sistema operacional móvel (ANDROID DEVELOPERS, 2015). Ele possui várias versões e suas utilizações variam de acordo com a Tabela 1:

Tabela 1 – Versões das Plataformas Android

Version	Codename	API	Distribution
2.2	Froyo	8	0.2%
2.3.3 - 2.3.7	Gingerbread	10	3.8%
4.0.3 - 4.0.4	Ice Cream Sandwich	15	3.4%
4.1.x	Jelly Bean	16	11.4%
4.2.x	Jelly Bean	17	14.5%
4.3	Jelly Bean	18	4.3%
4.4	KitKat	19	38.9%
5.0	Lollipop	21	15.6%
5.1	Lollipop	22	7.9%

Fonte: Adaptada de Android Developers (2015)

Nota: Dados recolhidos durante um período de 7 dias em 5 de outubro de 2015. Todas as versões com distribuição inferior a 0,1% não são mostradas (ANDROID DEVELOPERS, 2015).

Com base nas técnicas e tecnologias aqui apresentadas foi possível estruturar e desenvolver um SI Web capaz de capturar, tratar e armazenar informações obtidas da Internet. Também foi possível desenvolver um motor de busca capaz de apresentar essas informações de modo muito mais dinâmico e inteligente.

2.3 Trabalhos Relacionados

Foram realizadas pesquisas com o intuito de identificar trabalhos semelhantes, que utilizassem as mesmas ferramentas e cumprissem papéis similares. Dentre os sistemas que utilizam Solr foi encontrado o Ainfo e na busca por aplicativos para bibliotecas foi encontrado o SIBi.

Ainfo é um sistema desenvolvido pela Embrapa Informática Agropecuária para gestão de bibliotecas e recuperação de informações. Ele integra bases de dados documentais, cadastrais e de processos bibliográficos. Recentemente recebeu uma atualização, e passou a utilizar como mecanismo de indexação e busca textual o Solr, apresentando melhora significativa no desempenho (NASCIMENTO; VACARI, 2010).

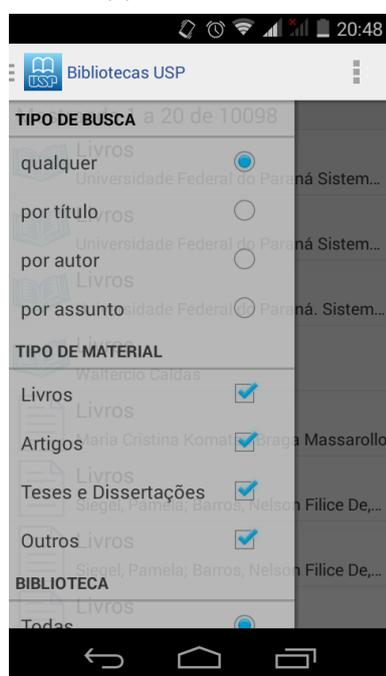
“O uso da ferramenta Solr mostrou-se adequado para a construção da camada de indexação e busca do módulo de consulta institucional do sistema Ainfo, bem como para conquista de autonomia tecnológica em software livre para recuperação de informação. Por fim, o uso de software livre tem-se consolidado nas atividades de pesquisa, desenvolvimento e inovação da Embrapa.” (NASCIMENTO; VACARI, 2010, p. 146).

A Universidade de São Paulo (USP), criou o aplicativo Sistema Integrado de Bibliotecas SIBi, onde é possível realizar buscas online e através de uma consulta, são retornados o resultados dos acervos das bibliotecas da universidade. O aplicativo agiliza e facilita a procura por livros realizadas pelos alunos. A Figura 4 apresenta duas telas do aplicativo, que pode ser acessado em: <https://play.google.com/store/apps/details?id=br.usp.bibliotecas_osp>.

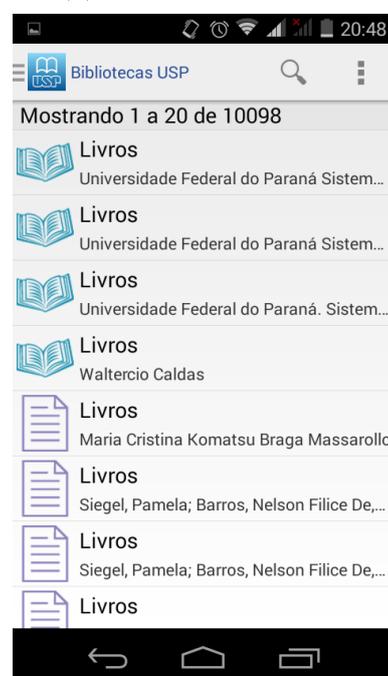
Os sistemas Ainfo e SIBi são semelhantes ao sistema Busca Livro por atuarem no mesmo domínio, a busca em acervos bibliográficos. O Ainfo faz uso da mesma tecnologia utilizada no sistema Busca Livro para indexação e busca, o Solr, e o SIBi, assim como o Busca Livro, possui uma interface móvel que faz busca em uma base de dados na Web.

Figura 4 – Telas: Sistema Integrado de Bibliotecas SIBi

(a) Menu lateral



(b) Resultados de busca



Fonte: Captura de tela do aplicativo SIBi

3 MATERIAIS E MÉTODOS

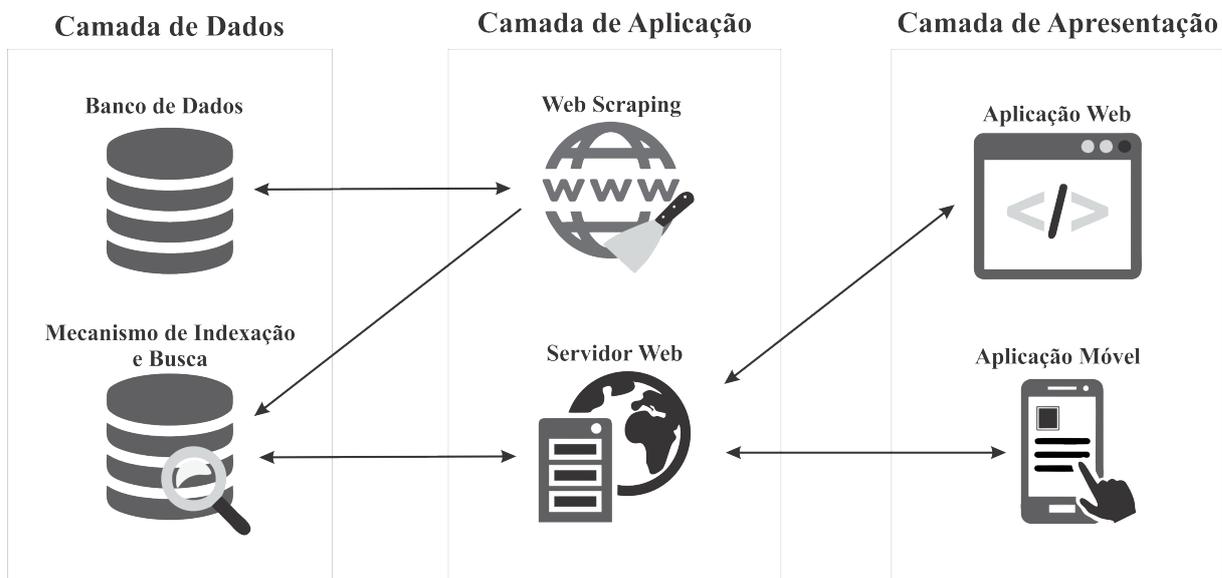
O desenvolvimento do presente estudo foi dividido em cinco fases descritas a seguir:

Na fase de estudo teórico foi realizada uma pesquisa na literatura e em fontes *online*, a fim de obter um embasamento teórico dos conceitos utilizados no desenvolvimento do projeto.

Na fase de planejamento foram estipuladas as atividades a serem realizadas na implementação do sistema e as tecnologias utilizadas. Nesta fase também foi definida a metodologia de desenvolvimento de sistema: XP (descrita na subseção 2.2).

Na fase de implementação foi definida uma arquitetura em três camadas: Camada de Dados, Camada de Aplicação e Camada de Apresentação. Com isso, foi possível organizar os conceitos e ferramentas para melhor atender aos requisitos do sistema. As camadas são representadas na Figura 5.

Figura 5 – Estrutura de camadas do sistema Busca Livro



Fonte: Elaborada pelos autores

A Camada de Dados consiste em um repositório de informações recebidas da camada de aplicação. Para que o repositório funcionasse, foram necessárias tecnologias de armazenamento e indexação de dados, que fornecem informações ao motor de busca:

- O PostgreSQL é um SGBD objeto-relacional de código aberto, conhecido pela sua confiabilidade, integridade de dados e conformidade a padrões. Como um Banco de Dados de nível corporativo, o PostgreSQL possui várias funcionalidades, como:

a replicação assíncrona, o controle de concorrência multiversão, transações agrupadas (*savepoints*) e ser altamente escalável na grande quantidade de dados que pode gerenciar (MOMJIAN, 2001).

- O Solr é um mecanismo de indexação e busca textual de código aberto baseado na biblioteca Lucene, mantido pela *Apache Software Foundation* e que oferece recursos sofisticados de indexação e busca textual. Possui busca com operadores *booleanos*, busca específica por campo, *highlighting*, *facets* e *caching* sobre o resultado da busca, integração com Banco de Dados relacionais, replicação de dados e uma interface de administração Web fácil e prática para utilização. O Solr é altamente escalável e tolerante a falhas, com uma rápida configuração de recursos ele suporta grandes quantidades de dados e de requisições, aproximando o ambiente de desenvolvimento ao de produção. A utilização do Solr é simples, mas a configuração da busca demanda um estudo aprofundado dos filtros de busca (SMILEY; PUGH, 2009).

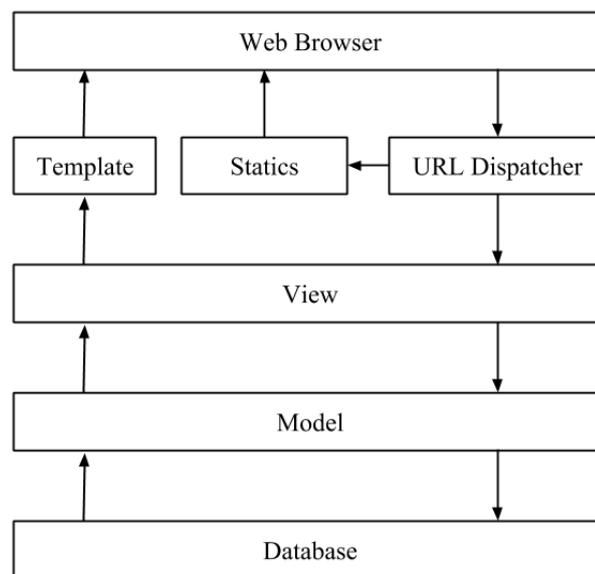
A Camada de Aplicação abrange a parte lógica do trabalho, onde foram implementadas as rotinas de captura, *parser*, validação de dados, estruturação desses dados, implementação do servidor Web e a API de comunicação com a camada de dados:

- O Requests é uma biblioteca de requisições HTTP escrita em Python amplamente utilizada e bastante popular. Existe uma biblioteca padrão do Python chamada *urllib2* que fornece esta possibilidade, porém exige mais linhas de código para realizar a mesma ação, por este motivo foi escolhida a biblioteca Requests (REITZ, 2015).
- A biblioteca lxml escrita em Python é bastante utilizada como uma ferramenta de *parser*, capaz de transformar um documento HTML em um elemento de árvore. Baseada nas bibliotecas em C *libxml2* e *libxslt*, esta ferramenta possui uma grande eficiência para conversão destes documentos em *ElementTree*¹ (PYTHON SOFTWARE FOUNDATION, 2014).
- O Celery é uma aplicação simples e robusta para criação de filas e agendamentos de tarefas. Com esta aplicação é possível criar tarefas assíncronas para serem executadas em momentos específicos de forma distribuída através de *workers*, que são instâncias que executam essas tarefas. Tais tarefas são responsáveis por alimentar a base de dados. Para que elas sejam executadas, é necessário um gerenciador de filas de tarefas que seja capaz de mantê-las e organizá-las, de tal modo que essas tarefas sejam executadas em ordem correta e estejam agendadas mesmo que algo interrompa a execução. O RabbitMQ é uma aplicação que preenche esses requisitos (CELERY, 2009-2014).

¹ É um tipo de classe flexível para armazenar objetos hierarquicamente na memória. Fonte: <<https://docs.python.org/2/library/xml.etree.elementtree.html>>

- Para facilitar a conexão do Solr com a aplicação foi utilizado o PySolr, que é um módulo Python utilizado para abstrair as *queries* realizadas no Solr, convertendo as consultas em métodos e o retorno em objeto Python, facilitando a implementação (LINDSLEY, 2008-2015).
- O Django é um *Framework* escrito em Python utilizado no desenvolvimento de aplicações Web. É um projeto de código aberto sob licença *Berkeley Software Distribution* (BSD), e assim como outros *Frameworks* ágeis, ele utiliza o padrão *Model Template View* (MTV), baseado no princípio *Don't Repeat Yourself* (DRY), que leva o desenvolvedor a aproveitar ao máximo o código já feito evitando repetições. A Figura 6 mostra o esquema de execução da aplicação. O navegador realiza uma requisição, a URL é analisada e realiza a chamada da *View* correspondente que realiza o processamento, e caso seja necessário, é realizado o registro de dados no *Model*, que é uma abstração do BD (HOLOVATY; KAPLAN-MOSS; AL., 2009).

Figura 6 – Esquema de funcionamento do Django



Fonte: Elaborada pelos autores

A Camada de Apresentação é formada por duas interfaces que possuem interação direta com os usuários, sendo elas uma interface Web e outra móvel. Essas interfaces são responsáveis por apresentar resultados para as buscas realizadas de forma clara e organizada:

- O *Material Design* é um padrão de interface criado e projetado pela Google, com o objetivo de desenvolver um sistema de *design* que permite unificar a experiência do usuário de todos seus produtos em qualquer plataforma. O *MaterializeCSS* é

um *Framework front-end* moderno e responsivo² baseado no *Material Design*, para facilitar e padronizar o desenvolvimento da interface de aplicações Web, e que conta com vários componentes pré-definidos e que permitem personalização (WANG et al., 2015).

- *Asynchronous Javascript and XML* (AJAX) é um conjunto de tecnologias providas por navegadores para tornar páginas Web mais interativas com o usuário, utilizando requisições assíncronas de informações. No desenvolvimento da aplicação Web deste projeto o AJAX foi utilizado em quase todas as ações, tornando o carregamento de informações mais rápido, pois o *layout* é carregado apenas uma vez pelo cliente, após isso, apenas as informações trafegam pela rede (SILVA, 2009).
- O Android Studio é a *Integrated Development Environment* (IDE) oficial da Google desenvolvida especialmente para ajudar no desenvolvimento produtivo e organizado de aplicações para o sistema operacional Android. Esta IDE foi utilizada em conjunto com o guia de telas *Material Design*, criado para padronizar os movimentos e transições entre plataformas e dispositivos.

Na quarta fase foi aplicada uma pesquisa piloto em forma de questionário online. A proposta do questionário foi verificar junto ao público alvo a utilidade do sistema e buscar erros de implementação. Para a elaboração do questionário foram utilizadas as orientações do *Goal Question Metric* (GQM). A partir dos resultados da pesquisa piloto foi possível ajustar questões e corrigir erros. Em seguida uma nova pesquisa foi aplicada junto aos usuários (SOLINGEN; BERGHOUT, 1999).

Na quinta e última fase foram analisados e interpretados os dados resultantes dos questionários e realizada uma comparação de desempenho e usabilidade dos sistemas das bibliotecas da UEMS e UFGD e o sistema Busca Livro.

² São *Layouts* amigáveis e ajustáveis a qualquer formato e tamanho/resolução de tela. Fonte: <<https://www.webgenium.com.br/blog/voce-sabe-o-que-e-layout-responsivo.html>>

4 RESULTADOS

Neste capítulo são apresentados o processo de implementação, os problemas encontrados durante esse processo e as soluções aplicadas. Também são descritos o projeto e as interfaces desenvolvidas, resultados de desempenho e análises do questionário aplicado.

A Figura 5 apresentada na seção anterior mostra a arquitetura do sistema Busca Livro, que foi dividida em cinco partes. Cada uma dessas partes é abordada nesta seção, mostrando as formas de implementação e as soluções para problemas encontrados durante esse processo. O Banco de Dados é descrito na subseção 4.1, os *Web Scrapings* da UEMS e UFGD são descritos nas subseções 4.2 e 4.3 respectivamente, o mecanismo de indexação e busca é descrito na subseção 4.4, o servidor Web é descrito na subseção 4.6 e as interfaces Web e móvel são descritas na subseção 4.7.

A utilização do XP como modelo permitiu que o desenvolvimento fosse realizado de forma ágil mesmo com uma equipe de apenas dois programadores. O uso de suas técnicas auxiliaram na organização do código e na separação dos recursos e sua ordem de implementação por prioridade.

4.1 Base de dados do Busca Livro

A base de dados do sistema Busca Livro foi dividida em dois repositórios de dados, um utilizando um BD armazenado no PostgreSQL, outro diretamente no Solr.

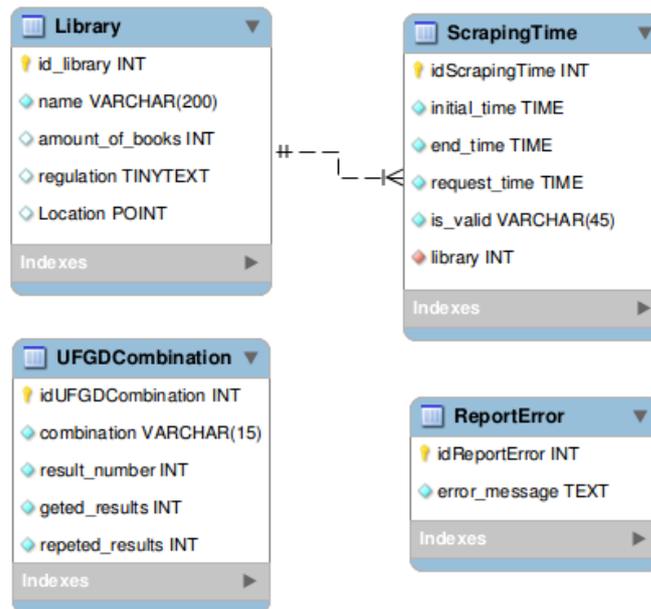
Todos os dados dos documentos (Livros, Periódicos entre outros) capturados foram armazenadas em um *collection*¹ Solr, que é uma coleção de índice único que pode abranger vários núcleos de informações. Esses documentos foram armazenados em forma de dicionário de dados suportado pelo Solr, que recebe as informações e as armazena e indexa de forma dinâmica. Esta opção foi escolhida devido ao alto desempenho de acesso às informações que o Solr disponibiliza.

Outras Informações foram armazenadas no PostgreSQL simplesmente por não precisar de busca textual e indexação dinâmica fornecidos pelo Solr. Os dados armazenados no PostgreSQL são referentes as bibliotecas, combinações utilizadas pelo *Web Scraping* da UFGD, os dados de tempo de execução e as mensagens de erros reportados pelos usuários. O Modelo Entidade Relacionamento das tabelas utilizadas no PostgreSQL é apresentado na Figura 7. Nela é possível observar que foram utilizadas quatro tabelas. A tabela “Library” armazenou as informações básicas das bibliotecas como nome e quantidade de itens no acervo. A tabela “ScrapingTime” foi utilizada para armazenar os tempos de

¹ Fonte: <<https://cwiki.apache.org/confluence/display/solr/How+SolrCloud+Works>>

requisições dos algoritmos de captura de dados. A tabela “UFGDCombination” armazenou as combinações utilizadas para realizar as capturas no acervo da UFGD, e por fim a tabela “ReportError” foi utilizada para armazenar as mensagens enviadas pelos usuários sobre os erros reportados.

Figura 7 – Modelo Entidade Relacionamento



Fonte: Elaborada pelos autores

4.2 Web Scraping da UEMS

O primeiro desafio de implementar o algoritmo de *Web Scraping* do acervo da UEMS, foi encontrar uma forma de atingir todos os itens do acervo. Para isso se fez necessário um estudo de como é o funcionamento das buscas no acervo. A biblioteca da UEMS disponibiliza um serviço de consultas ao seu acervo por meio de uma página na Internet, que pode ser visualizada na Figura 8. Essa página contém um campo de busca onde é possível inserir palavras-chave e alguns filtros que ajudam a configurar a busca. Quando essa busca é executada, os resultados são dispostos e o usuário pode clicar nos itens, sendo redirecionado para a página de informações.

Essa página utiliza o método *GET* na requisição, ou seja, permite que os dados do formulário possam ser alterados através de sua URL <<http://www.uems.br/biblioteca/informacoes.php>>. Cada item do acervo da UEMS possui um identificador chamado de Código, este código é enviado para a página de informações por meio do método *GET* juntamente com a variável Local conforme demonstrado pela Figura 9. A variável Local

Figura 8 – Página de busca da Biblioteca da UEMS



Fonte: Captura de tela de <www.uems.br/biblioteca>

define a unidade universitária em que a busca será realizada, podendo assumir como valor um identificador da unidade ou “TODOS” para mostrar de todos os acervos.

Figura 9 – Página de informações dos itens do acervo

Exemplares			
Num. de Exemplares	Volume	Localização	Disponíveis
1		DOURADOS	1
1		PARANAIBA	1

Fonte: Captura de tela de <http://www.uems.br/biblioteca/informacoes.php>

Após alguns testes foi constatado que, mesmo quando a variável `Codigo` assume um valor inválido, a página não retorna mensagem de erro, mas sim a mesma página, porém com os campos de informações vazios, como mostra a Figura 10. Quando o campo `Local` assume o valor de um determinado acervo, por exemplo “DOURADOS”, e o

campo `Codigo` assume valor válido de um item, mas este item não pertence ao acervo de DOURADOS, as informações são retornadas, porém a tabela de exemplares fica vazia, como mostra a Figura 11.

Figura 10 – Página de informações com `Codigo` inválido

The screenshot shows a web browser window with the URL `www.uems.br/biblioteca/informacoes.php?Codigo=X&Local=TODO`. The page header includes the logo of the UEMS library and the text 'Universidade Estadual de Mato Grosso do Sul'. The main content area is divided into two sections: 'Detalhes' and 'Exemplares'. The 'Detalhes' section contains a table with the following fields: Título, Autor, Indicação de Responsabilidade, Material, Código, Editora, Local, Série, CDD, PHA, Descrição Física, Edição, and Ano. All these fields are empty. The 'Exemplares' section contains a table with the following columns: Num. de Exemplares, Volume, Localização, and Disponíveis. This table is also empty. At the bottom of the page, there is a 'Voltar' button.

Fonte: Captura de tela de `<www.uems.br/biblioteca/informacoes.php?Codigo=X&Local=DOURADOS>`

Figura 11 – Página de informações com `Codigo` válido

The screenshot shows a web browser window with the URL `www.uems.br/biblioteca/informacoes.php?Codigo=10000&Local=DOURADOS`. The page header includes the logo of the UEMS library and the text 'Universidade Estadual de Mato Grosso do Sul'. The main content area is divided into two sections: 'Detalhes' and 'Exemplares'. The 'Detalhes' section contains a table with the following fields: Título: **Curso de direito civil brasileiro responsabilidade civil**, Autor: **Diniz, Maria Helena**, Indicação de Responsabilidade, Material, Código: **10000**, Editora: **SARAIVA**, Local: **São Paulo**, Série, CDD: **346.81**, PHA: **D612c**, Descrição Física: **xiii, V. 7 ; 23 cm.**, Edição: **9. ed. aum. e atual. 1995.**, and Ano: **1995**. The 'Exemplares' section contains a table with the following columns: Num. de Exemplares, Volume, Localização, and Disponíveis. This table is empty. At the bottom of the page, there is a 'Voltar' button.

Fonte: Captura de tela de `<www.uems.br/biblioteca/informacoes.php?Codigo=10000&Local=DOURADOS>`

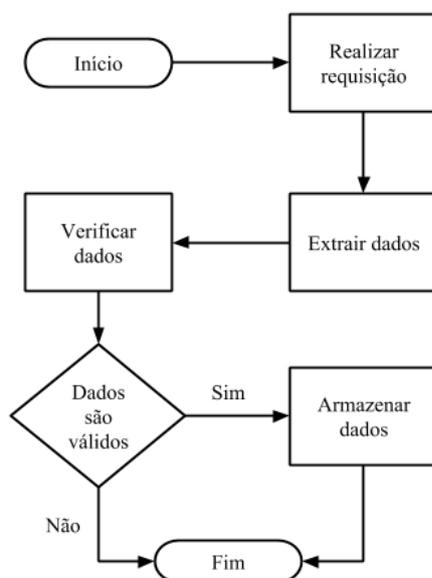
Com base nos resultados dos testes realizados, foi possível desenvolver uma condição para definir se determinado `Codigo X` é um código válido para determinado `Local Y`.

Para estes requisitos serem satisfeitos, e conseqüentemente identificar os itens válidos no acervo, foi necessário executar os seguintes passos:

- Realizar uma requisição na URL de informações: <http://www.uems.br/biblioteca/informacoes.php?Codigo=X&Local=Y>
- Extrair os dados necessários da página retornada
- Verificar se a tabela de informações contém dados e se a tabela de exemplares não está vazia.

A função `uems_web scraping (code)` recebe como parâmetro uma variável `code` que realiza os passos descritos anteriormente a fim de verificar se o valor é um código válido, e caso seja, o item é inserido na base de dados do Busca Livro. Este processo pode ser observado na Figura 12.

Figura 12 – Fluxograma do Web Scraping da UEMS



Fonte: Elaborada pelos autores

Para capturar os dados de todos os itens do acervo seria necessário conhecer os códigos de todos os itens. Esses códigos seguem uma sequência, embora ela não seja consecutiva. Foi então implementada uma rotina que realiza chamadas da função `uems_web scraping (code)` com um determinado intervalo de códigos. A função foi executada até que esse intervalo fosse refinado, a ponto de se definir que os códigos de todos os itens do acervo se encontram entre 1 e 45000. O processo de coleta de dados dos periódicos é o mesmo dos outros itens alterando apenas a URL para <http://www.uems.br/biblioteca/informacoes_peri.php?Codigo=X&Local=1> e o fim do intervalo para 4000.

Analisando os dados coletados, foi observado que existem quatro tipos de itens que a biblioteca da UEMS armazena, sendo Livro, Periódico, Monografia/TCC e Tese. Para este trabalho foi definido que o campo Tese fosse representado pelo campo Tese/Dissertação a fim de ser agrupado aos itens da biblioteca da UFGD. Foi coletado um total de 22045 itens válidos.

4.3 Web Scraping da UFGD

A implementação do algoritmo de *Web Scraping* da biblioteca da UFGD contou com vários desafios. Primeiramente foi necessário entender como realizar buscas que retornassem resultados satisfatórios, de maneira que abrangessem todos os objetos que a plataforma pudesse disponibilizar. Posteriormente, implementar funções que realizassem requisições com essas buscas para obter assim as páginas que contivessem informações relevantes para o trabalho.

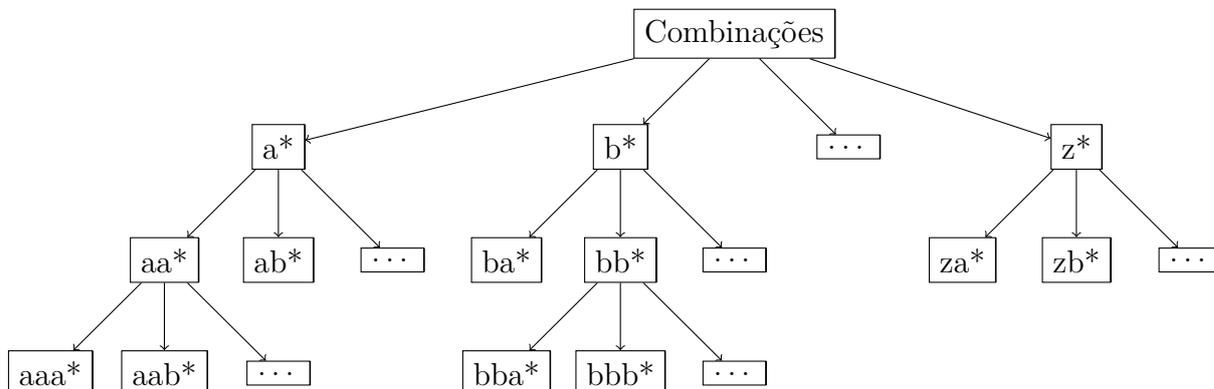
O processo de realizar buscas consistiu em gerar combinações com letras que quando buscadas no sistema da biblioteca da UFGD, retornam três formatos de resultados diferentes:

1. Nenhum item é retornado
2. São retornados mais de 2 mil itens, e neste caso o sistema de buscas da UFGD limita a lista em 2 mil itens
3. É retornado ao menos 1 item e menos que 2 mil itens

Através destes resultados, foi possível construir uma sequência de combinações que garantisse que todos os itens existentes no banco de dados da biblioteca fossem apresentados em algum momento.

Ao inserir um sinal de asterisco (*), o sistema da biblioteca interpreta que possa haver “qualquer coisa” após o texto que precede o símbolo. Supondo o exemplo de busca do termo **ter***, o sistema de buscas da UFGD retorna resultados como “terra”, “terráqueo”, “terreno”, etc. Isso quer dizer que é possível gerar combinações com todas as letras do alfabeto precedendo o símbolo de asterisco.

Neste caso, buscar por **a***, **b***, **c***, ..., **z*** garantiria que todos os itens do banco de dados da biblioteca fossem retornados, pelo simples fato de todos os itens iniciarem com alguma das letras buscadas. O que impediu a utilização dessa forma de busca, foi o fato de o sistema restringir os resultados da busca à 2 mil itens, ou seja, o caso de a busca por **a*** retornar mais de 2 mil itens, ocasionou em descer um nível na árvore de combinações, ou seja, **aa***, **ab***, **ac***, ..., **az***, como pode ser visualizado na árvore da Figura 13:

Figura 13 – Árvore de combinações de buscas do *Web Scraping* da UFGD

Fonte: Elaborada pelos autores

Após a descoberta desta técnica de geração de combinações, o processo de obter os dados consistiu em dois processos. Primeiro foram capturadas todas as combinações válidas, representado no fluxograma da Figura 14a, depois foram capturados todos os livros para cada combinação, representado pelo fluxograma da Figura 14b, de acordo com os seguintes passos:

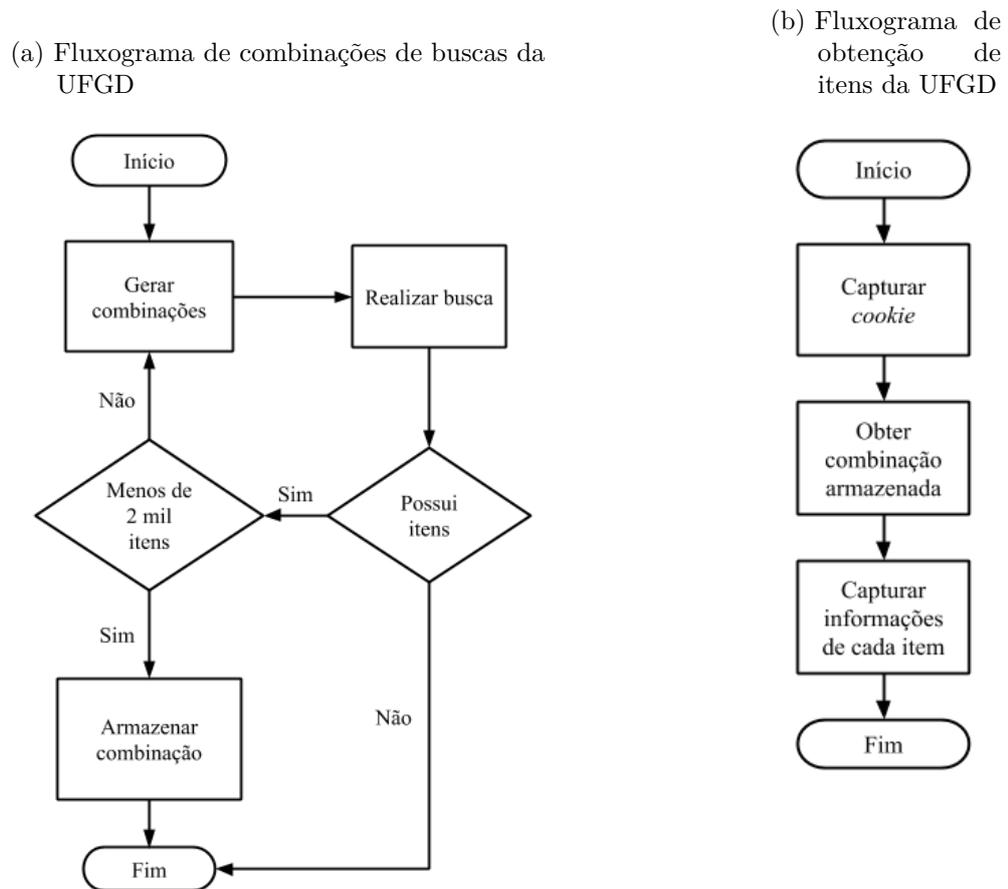
1. Capturar *cookie*: Capturar o identificador do *cookie* de sessão chamado JSESSIONID gerado pelo sistema da biblioteca
2. Obter combinação armazenada: Executar requisições para cada combinação gerada no primeiro processo
3. Capturar informações de cada item: Executar uma requisição *POST* para cada item retornado, simulando um clique em um item específico através do navegador Web e executar o *parser* para cada item requisitado

Após os testes e observações, foi concluído que existem três tipos de itens que o sistema da biblioteca da UFGD armazena, sendo eles Livro, Tese/Dissertação e Mídia. Para este trabalho, foi definido que o campo Mídia fosse representado pelo campo Outro.

4.4 Armazenamento e indexação das informações capturadas

Após a execução dos *parsers* da UEMS e UFGD, os dados capturados precisaram ser armazenados. Para que isso fosse padronizado, foi preciso definir um descritor para estes dados. O descritor criado consistiu em um dicionário de dados com os campos *id*, *title*, *description*, *library*, *kind*, *author*, *publication*, *cdd*, *pha*, *year*, *periodicity*, *country*, *language*, *editor*, *cnn*, *collections* e

Figura 14 – Fluxogramas de execução do Web Scraping da UFGD



Fonte: Elaborada pelos autores

knowledge_area. O campo id foi obtido através da geração de *hash*² de todos os campos do item capturado.

Através da aplicação PySolr foi possível transferir tal dicionário preenchido pelos dados capturados no processo de *parser* para que fossem armazenados na coleção de dados criada no Solr.

Para que o Solr recebesse e indexasse de maneira correta as informações, foram criados *fields*, configurados especialmente para determinados campos do trabalho. Um dos campos que recebeu um *field* especial foi o *title*, cuja implementação pode ser observada no Algoritmo 4.1.

Algoritmo 4.1 – Implementação do *field* Solr para o campo *title*

```

1 | <fieldType name="title_pt_text" class="solr.TextField" positionIncrementGap
   |     ="100">
2 | <analyzer type="index">

```

² Hash Table é uma estrutura de dados utilizada para implementar uma estrutura que mapeia chaves para valores. Fonte: <<http://interactivepython.org/runestone/static/pythonds/SortSearch/Hashing.html>>

```
3 <tokenizer class="solr.WhitespaceTokenizerFactory"/>
4 <filter class="solr.PortugueseMinimalStemFilterFactory"/>
5 <filter class="solr.EdgeNGramFilterFactory" minGramSize="4" maxGramSize
  ="25"/>
6 <filter class="solr.LowerCaseFilterFactory"/>
7 <filter class="solr.ASCIIFoldingFilterFactory"/>
8 <filter class="solr.StopFilterFactory" ignoreCase="true" words="lang/
  stopwords_pt.txt" format="snowball"/>
9 </analyzer>
10 <analyzer type="query">
11 <tokenizer class="solr.WhitespaceTokenizerFactory"/>
12 <filter class="solr.PortugueseMinimalStemFilterFactory"/>
13 <filter class="solr.LowerCaseFilterFactory"/>
14 <filter class="solr.ASCIIFoldingFilterFactory"/>
15 <filter class="solr.StopFilterFactory" ignoreCase="true" words="lang/
  stopwords_pt.txt" format="snowball"/>
16 <filter class="solr.SynonymFilterFactory" synonyms="synonyms_pt.txt"
  ignoreCase="true" expand="true"/>
17 </analyzer>
18 </fieldType>
```

Neste campo os analisadores aplicados foram:

- *solr.WhitespaceTokenizerFactory* que tem como função separar cada palavra do campo por espaços em branco, o que aumenta a quantidade de correspondências
- *solr.PortugueseMinimalStemFilterFactory* que realiza derivações com base no idioma português
- *solr.EdgeNGramFilterFactory* que quebra a palavra analisada em sub termos para serem comparadas com a busca. No caso deste trabalho, foi utilizado o mínimo de 2 caracteres, por exemplo palavra “nigeriano” no campo `title` seria quebrada em “ni”, “nig”, “nige”, “niger”, “nigeri”, “nigeria”, “nigerian” e “nigeriano”, e todas as sub palavras geradas seriam utilizadas na busca
- *solr.LowerCaseFilterFactory* que transforma todas as letras do campo em minúsculas antes de serem comparadas
- *solr.ASCIIFoldingFilterFactory* que transforma caracteres fora dos padrões *unicode*³ (letras acentuadas por exemplo) para seus correspondentes em *unicode*
- *solr.StopFilterFactory* que descarta palavras desnecessárias com base em um dicionário de palavras. Neste analisador, palavras como “a”, “o”, “que”, “do”, “para”, etc. são descartadas do campo antes de executar a comparação

³ É um padrão universal de codificação de caracteres mantido pela *Unicode Consortium*. Ele fornece a base para processamento, armazenamento e intercâmbio de dados em qualquer língua do mundo. Fonte: <<http://www.unicode.org/>>

- *solr.SynonymFilterFactory* que cria correspondências com base em um dicionário de sinônimos. Neste analisador, a palavra “admissão” obteria correspondência com a palavra “recepção” devido ao dicionário de sinônimos

Com base nesse *field* construído especialmente para o campo `title`, a análise diferenciada pôde promover melhores resultados para as buscas.

O sistema de classificação de resultados do Solr conta com um campo chamado *score* que é atribuído de acordo com a qualidade de correspondências dos analisadores. Palavras que são correspondidas através de sinônimos por exemplo, obtém pontuações menores que palavras com correspondências exatas. Isso contribui para que os resultados sejam mais concisos com os termos buscados e garante que assuntos relacionados também possam ser apresentados.

O Solr conta ainda com um recurso chamado *boost*, que consiste em acrescentar pontos para itens no momento da inserção. Isso garante que quando um item é encontrado por uma busca, seja acrescentado no seu score esta pontuação definida no momento da inserção. Para este trabalho, foi adicionado 2.0 pontos como *boost* para itens que fossem Livros. Essa medida foi tomada por conta de a maior parte dos interesses das buscas serem em relação à Livros. Essa diferença pode ser observada no Algoritmo 4.2 retornado pelo Solr quando buscado por “contraste”, onde a diferença do score entre um item do tipo Livro e outro item do tipo Monografia/TCC tem uma diferença de pouco mais de 2 pontos.

Algoritmo 4.2 – Trecho de uma resposta do Solr para a busca “contraste”

```
1  "response": {
2    "docs": [{
3      "kind": "Livro",
4      "title": "O poder e o grande motivador. Uma estrategia para a
          diversificacao dos negocios. Participacao dos trabalhadores:
          contrastes em tres paises",
5      "author": "McClelland, David C Gadon, Herman Hanan, Mack Foy, Nancy
          Burham, David H",
6      "score": 58.480904
7    },
8    {
9      "kind": "Monografia/TCC",
10     "title": "A epifania em contraste com a existencia em\" O jantar\"
          e\"Os obedientes\", de Clarice Lispector.",
11     "author": "Laboissier, Karen Grazieli Amaral.",
12     "score": 56.350735
13   }]
14 }
```

4.5 Execução do sistema de obtenção dos arquivos

Ao final da implementação foram realizados testes locais, onde se constatou a presença de alguns erros no código, esses foram prontamente corrigidos e o sistema se tornou apto para entrar em funcionamento. O Sistema Busca Livro é composto de três camadas, cada uma circundada de um ambiente de execução próprio, esse tipo de aplicação demanda recursos mínimos para ser executada. O processo de *deploy* foi realizado pela primeira vez em uma *Virtual Private Server* (VPS) contratada junto a empresa de *cloud computer* Digital Ocean (DO)⁴, e a segunda em uma *Virtual Machine* (VM) Laboratório de Segurança e Criptografia (LASCA)⁵ da Universidade Estadual de Campinas (UNICAMP). As especificações de cada uma das máquinas pode ser observada na Tabela 2:

Tabela 2 – Comparação das especificações dos servidores utilizados

Recurso	Digital Ocean	UNICAMP
Memória RAM	512 MB	4 GB
CPU <i>Core</i>	1	6
<i>Clock</i>	1.8 GHz	2.6 GHz
Espaço em disco	20 GB <i>Solid-State Drive</i>	40 GB <i>Hard Disk Drive</i>

Fonte: Elaborada pelos autores

O tempo médio de execução para captura de cada item do acervo da biblioteca da UEMS foi de 0,7011 segundos, enquanto o tempo total de execução para todos os itens foi de 5 horas, 29 minutos e 12 segundos, somando 22045 requisições válidas. Já os tempos de execução do algoritmo de *Web Scraping* da UFGD foi maior devido a complexidade do processo. Para cada item, o algoritmo levou em média 0,9940 segundos para executar a captura das informações, enquanto o tempo total de execução para a geração de combinações e a captura de todos os itens do acervo foi de 49 horas, 48 minutos e 23 segundos, totalizando 754188 requisições. Durante todo o processo de execução dos algoritmos, os servidores da UEMS e UFGD foram monitorados através de acessos aos serviços e nenhuma instabilidade foi notada.

Toda essa implementação precisou ser executada de forma autônoma. Para isso, foi utilizada a aplicação Celery para criar tarefas assíncronas e agendadas e o RabbitMQ como *broker* de mensagens.

4.6 API de requisições

Além do *boost* na inserção, foi necessário incluir prioridades maiores para determinados campos no momento da busca. Estas prioridades foram implementadas dentro da

⁴ <<https://www.digitalocean.com>>

⁵ <<http://www.lasca.ic.unicamp.br>>

API de comunicação entre as interfaces Web e móvel e a base de dados indexada no Solr.

Esta API consistiu em duas funções, `search_Q()` e `query_solr()`. A primeira é responsável por construir a requisição (*query*) que o usuário informa. Esse processo inclui o texto digitado pelo usuário no campo de busca, os tipos de itens definidos no momento da busca, a biblioteca em que se deseja buscar e em quais campos buscar. Essa função pode ser visualizada no Algoritmo 4.3.

Algoritmo 4.3 – Implementação da função de construção da *query* na API de comunicação

```

1 | def search_Q(search={}):
2 |     q = []
3 |     qf = 'title^5 author^0.5'
4 |     q.append(' (%s)' % (search['query']))
5 |     if search['field']:
6 |         qf = search['field']
7 |     if search['library']:
8 |         q.append(' (library:%s)' % (str(Library.objects.get(name=search['library']
9 |         ).id)))
10 |     if search['kind']:
11 |         q.append((' (%s)') % (' OR '.join(['kind:' + lib for lib in search['kind']
12 |         ])))
13 |     return ' AND '.join(q), qf

```

Alguns exemplos de *queries* construídas pela função são:

- (história) AND (library:1)
- (fatos e coisas douradenses) AND (library:1) AND (kind:Livro)
- (informática) AND (library:2) AND (kind:Tese/Dissertação)

É possível observar no Algoritmo 4.3 uma variável na linha 3 chamada `qf`, que corresponde aos campos em que a busca seria realizada. Nas interfaces Web e móvel, quando é feita uma busca livre, ou seja, sem escolher em qual campo buscar, a variável `qf` garante que a busca seja feita nos campos `title` e `author`, porém com uma prioridade 5 vezes maior para o campo `title` e reduz pela metade a prioridade do campo `author`.

A segunda função chamada `query_solr()` utiliza a *query* construída na função `search_Q()` juntamente com o parâmetro `qf` para realizar a busca. Primeiramente é criada uma instância Solr através da aplicação PySolr na linha 3 do Algoritmo 4.4.

Algoritmo 4.4 – Implementação da função de busca no Solr

```

1 | def query_solr(q='*:*', qf='', wt='json', fl=['*'], start=0, rows=10,
2 |     highlight=False):

```

```
2  if len(q) > 0:
3      solr = pysolr.Solr(settings.SOLR_BASEPATH + settings.SOLR_CORE)
4      params = {
5          'wt': wt,
6          'fl': ', '.join(fl),
7          'start': int(start),
8          'rows': int(rows),
9          'qf': qf,
10         'lowercaseOperators': 'true',
11         'stopwords': 'true',
12         'defType': 'edismax',
13     }
14     hl = {
15         'hl': 'true',
16         'hl.fl': 'title',
17         'hl.simple.pre': '<em>',
18         'hl.simple.post': '</em>',
19     }
20     if highlight:
21         params = dict(params.items() + hl.items())
22         docs = solr.search(q=q, **params)
23         for doc in docs.docs:
24             doc['library'] = Library.objects.get(id=int(doc['library'])).name
25     return docs
```

Posteriormente, foi construído um dicionário de dados chamado `params` com os seguintes itens:

- `wt`: corresponde ao formato de saída das informações
- `fl`: campos dos documentos retornados
- `start`: início da lista de documentos
- `rows`: quantidade de documentos
- `qf`: campos onde a busca será realizada
- `lowercaseOperators`: caso verdadeiro (*true*), a *query* buscada será transformada em letras minúsculas. Isso garante que a busca não seja *case-sensitive*⁶
- `stopwords`: caso verdadeiro (*true*), o Solr utiliza o analisador *solr.StopFilterFactory* com base no dicionário de palavras desnecessárias
- `defType`: *edismax* é um *parser* robusto do Solr que realiza análises avançadas com base nos dados informados

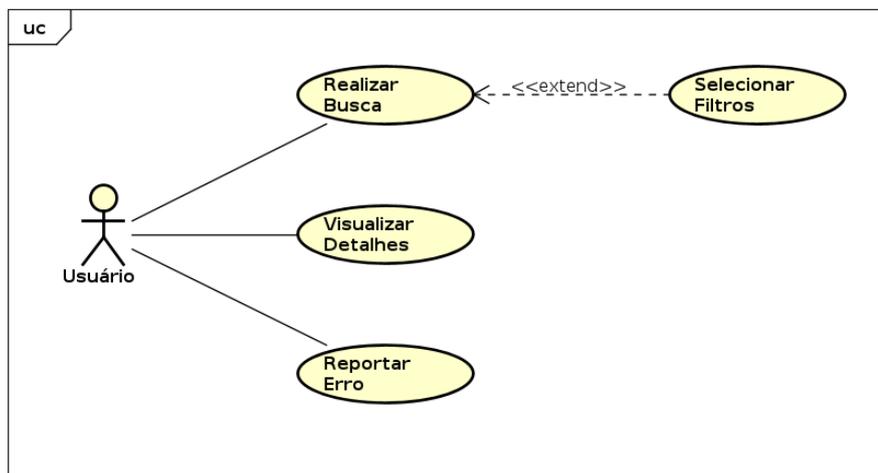
⁶ Sensível a letras maiúsculas e minúsculas.

Além dos parâmetros, existe ainda um dicionário chamado `h1` na linha 14. Esse dicionário corresponde à *highlight*, ou marcações que o Solr executa com relação às palavras em que são correspondentes com a busca. Esta função é utilizada apenas na interface Web.

4.7 Interfaces de Usuário

O acesso ao Busca Livro pode ser realizado por duas interfaces distintas, uma Web e outra móvel. As duas interfaces foram implementadas com base no diagrama de caso de uso da Figura 15, que foi desenvolvido na fase de projeto.

Figura 15 – Casos de uso das interfaces



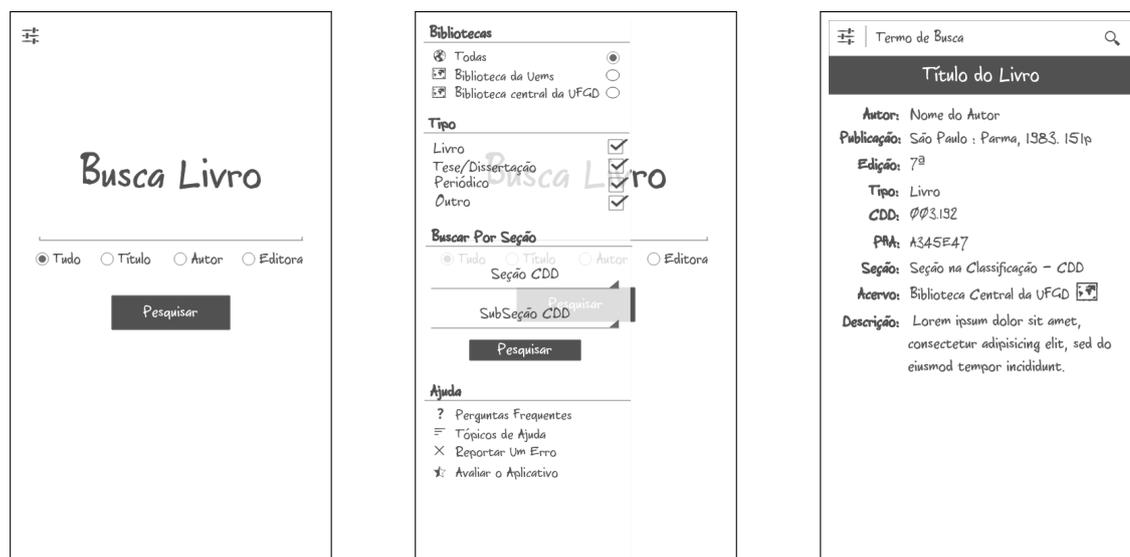
Fonte: Elaborada pelos autores

O diagrama de caso de uso é composto de um único ator, o usuário, que pode realizar as ações de: Realizar busca, Visualizar Detalhes, Reportar Erro e de forma opcional Selecionar Filtros. Ainda na fase de projeto foram desenvolvidos protótipos das telas, Figura 16, a fim de padronizar as duas aplicações, permitindo ao usuário encontrar interfaces semelhantes, independente da plataforma que utilizar.

A aplicação Web é baseada no modelo cliente-servidor, no lado do servidor foi implementada na linguagem Python utilizando o *Framework* Django, o lado do cliente foi desenvolvido utilizando a linguagem HTML utilizando o *Framework MaterializeCSS*, seguindo os padrões no conceito visual *Material Design*. A aplicação pode ser acessada através do link: `<http://buscalivro.info>`, onde o usuário encontra a página inicial, Figura 17. Nesta tela o usuário pode realizar a busca escolhendo por qual campo buscar.

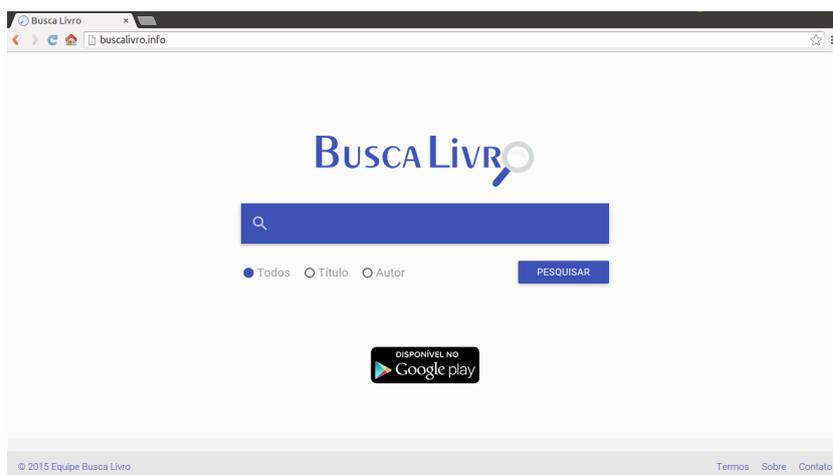
Ao realizar a busca o usuário é redirecionado para a página de resultados, Figura 18, onde é exibido o número de resultados encontrados, filtros para refinar a busca, e os itens encontrados. Existem três filtros: campo a ser buscado, biblioteca e tipo de material. Os

Figura 16 – Protótipo das Telas



Fonte: Elaborada pelos autores

Figura 17 – Interface Web: Página Inicial do sistema



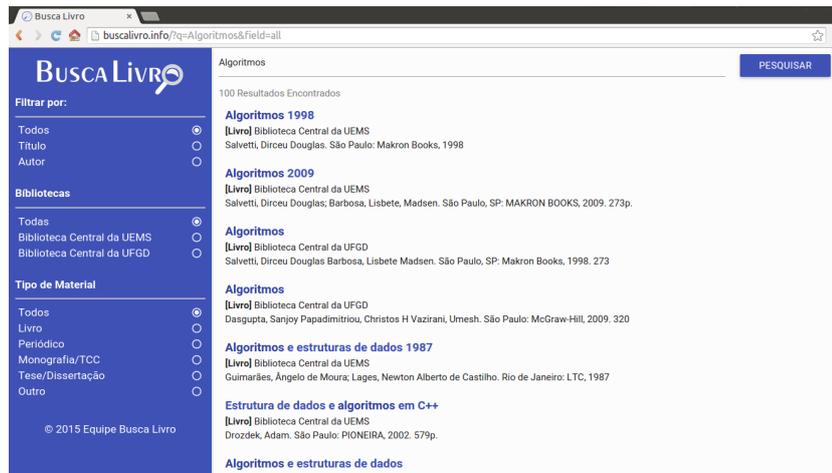
Fonte: Elaborada pelos autores

resultados são dispostos em uma lista com o título em destaque, e outras informações básicas.

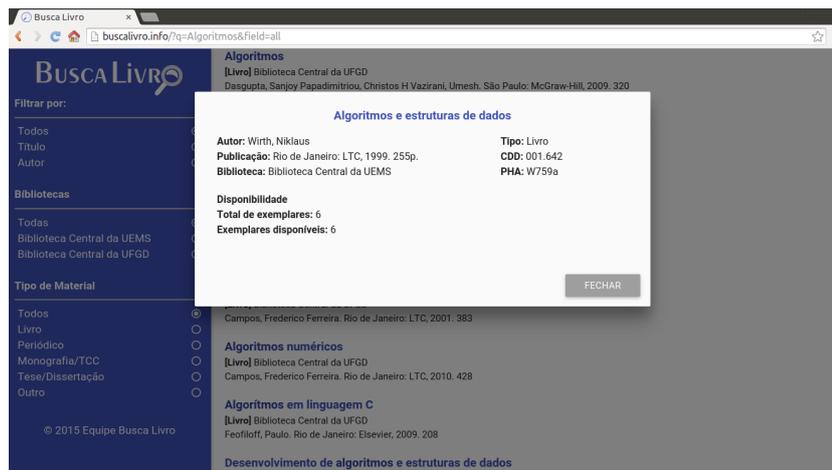
Ao clicar em um item da lista de resultados, é aberto um *modal* com informações mais detalhadas, Figura 19, buscando em tempo real a quantidade de exemplares disponíveis. Com exceção da busca na tela inicial, todas as requisições do são realizadas via AJAX, agilizando o carregamento dos dados.

A a interface móvel contou com a utilização da IDE Android Studio que promoveu o desenvolvimento de um aplicativo Android na linguagem de programação Java, baseado no conceito visual *Material Design*.

Figura 18 – Interface Web: Tela de resultados



Fonte: Elaborada pelos autores

Figura 19 – Interface Web: *Modal* com informações detalhadas

Fonte: Elaborada pelos autores

Aplicações Android são baseadas em componentes, e um desses componentes é chamado *Activity*, que tem como objetivo prover telas que usuários possam interagir. Um *Fragment* representa parte do componente *Activity*, onde é possível combinar vários *Fragments* em uma única *Activity* e construir uma interface com o usuário que seja multitelas.

A estrutura do aplicativo deste trabalho é composta de uma *Activity* que mantém os itens básicos que são apresentados em todas as sub telas. Estes itens são o menu lateral na Figura 20a, e a barra de ferramentas na Figura 20b, chamada *Toolbar* que se encontra no topo da tela. Conta também com vários *Fragments* que apresentam os conteúdos para cada caso.

Figura 20 – Telas da interface móvel



Fonte: Elaborada pelos autores

4.8 Análise dos resultados

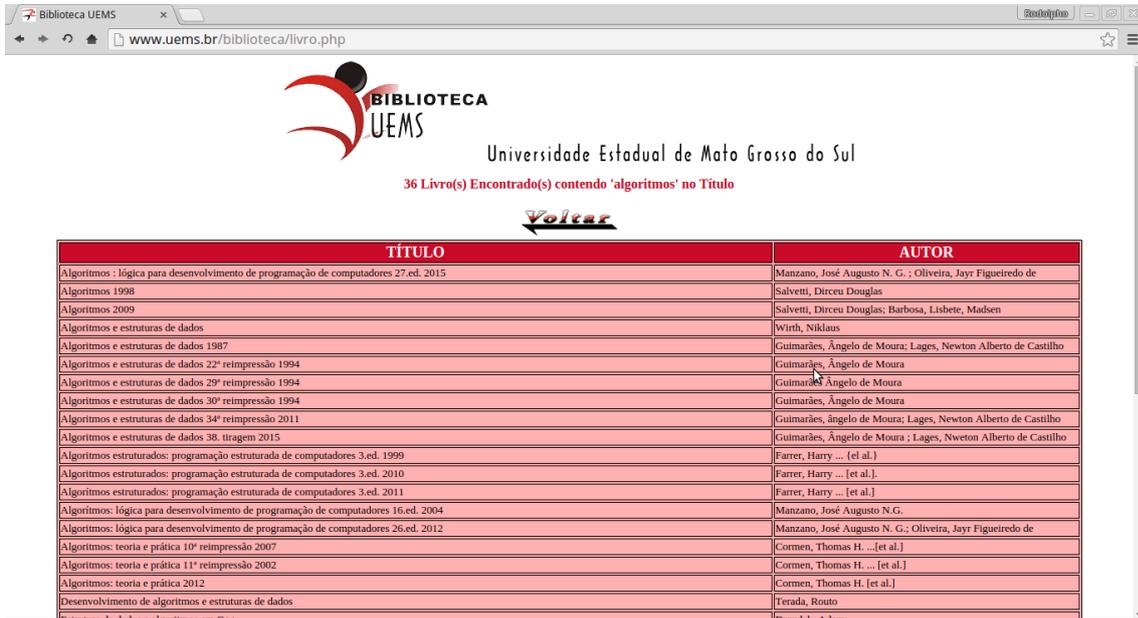
A partir do sistema Busca Livro concluído em sua fase de implementação, foi possível realizar comparativos com os sistemas das bibliotecas da UEMS e UFGD, em relação a interface, resultados de busca e desempenho da aplicação.

As Figuras 21, 22 e 23 fazem um comparativo entre a interface do sistema Busca Livro e os sistemas disponibilizados pela UEMS e UFGD. Neste caso, foram observados detalhes de experiência de usuário, disposição das informações e usabilidade das ferramentas. O grande diferencial entre a interface Web do sistema Busca Livro com relação aos outros dois sistemas é a utilização de um *layout* amigável baseado nos conceitos de *Material Design*.

Sobre a disposição das informações, nos sistemas das bibliotecas da UFGD e UEMS quando utilizados em dispositivos móveis, apresentam um *layout* desconfigurado, atrapalhando a navegação e, ocasionalmente, impossibilitando o uso. Pensando nesse problema o aplicativo Busca Livro foi projetado para se adaptar a qualquer tamanho de tela. A fim de validar a comparação, foram realizados testes na ferramenta de testes da Google, chamada *Google Developers Mobile-Friendly Test*⁷, onde é medida a compatibilidade com dispositivos móveis, o resultado está disposto no Apêndice C. Nele é possível observar

⁷ <<https://www.google.com/webmasters/tools/mobile-friendly>>

Figura 21 – Interfaces Web dos sistemas Busca Livro, UEMS e UFGD

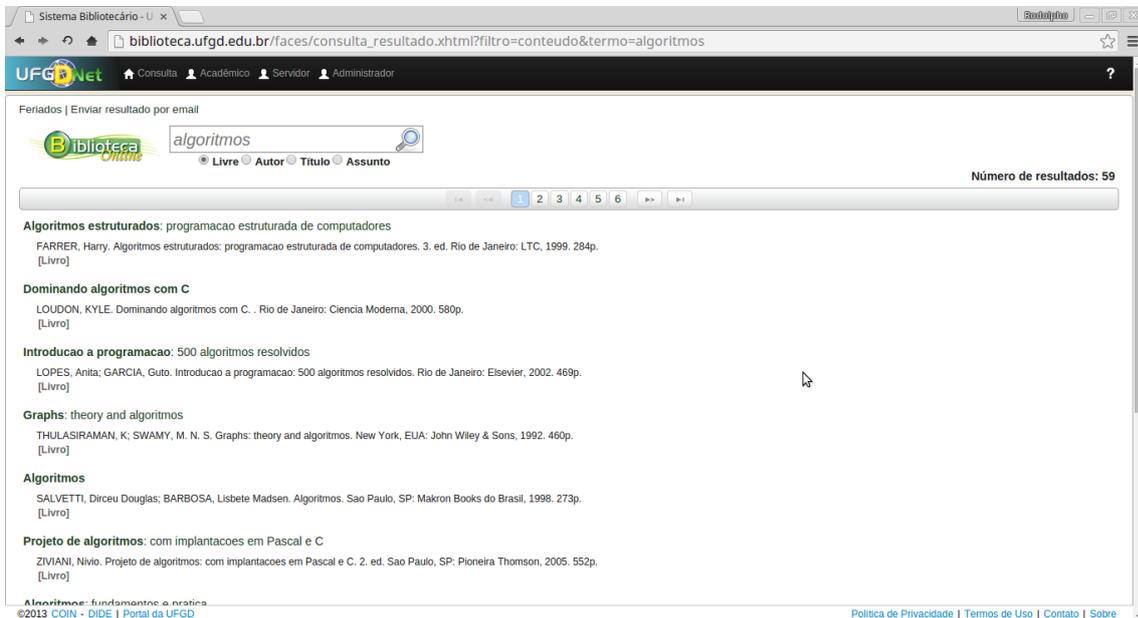


The screenshot shows the UEMS library website interface. At the top, there is a logo for 'BIBLIOTECA UEMS' and the text 'Universidade Estadual de Mato Grosso do Sul'. Below this, a message states '36 Livro(s) Encontrado(s) contendo 'algoritmos' no Titulo'. A 'Voltar' button is visible. The main content is a table with two columns: 'TÍTULO' and 'AUTOR'. The table lists various books related to algorithms, including titles like 'Algoritmos: lógica para desenvolvimento de programação de computadores' and 'Algoritmos estruturados: programação estruturada de computadores', along with their respective authors.

TÍTULO	AUTOR
Algoritmos: lógica para desenvolvimento de programação de computadores 27.ed. 2015	Manzano, José Augusto N. G. ; Oliveira, Jayr Figueiredo de
Algoritmos 1998	Salveti, Dirceu Douglas
Algoritmos 2009	Salveti, Dirceu Douglas; Barbosa, Lisbete, Madsen
Algoritmos e estruturas de dados	Wirth, Niklaus
Algoritmos e estruturas de dados 1987	Guimarães, Ângelo de Moura; Lages, Newton Alberto de Castilho
Algoritmos e estruturas de dados 22ª reimpressão 1994	Guimarães, Ângelo de Moura
Algoritmos e estruturas de dados 29ª reimpressão 1994	Guimarães, Ângelo de Moura
Algoritmos e estruturas de dados 30ª reimpressão 1994	Guimarães, Ângelo de Moura
Algoritmos e estruturas de dados 34ª reimpressão 2011	Guimarães, Ângelo de Moura; Lages, Newton Alberto de Castilho
Algoritmos e estruturas de dados 38. tiragem 2015	Guimarães, Ângelo de Moura ; Lages, Nweton Alberto de Castilho
Algoritmos estruturados: programação estruturada de computadores 3.ed. 1999	Farrer, Harry ... [et al.]
Algoritmos estruturados: programação estruturada de computadores 3.ed. 2010	Farrer, Harry ... [et al.]
Algoritmos estruturados: programação estruturada de computadores 3.ed. 2011	Farrer, Harry ... [et al.]
Algoritmos: lógica para desenvolvimento de programação de computadores 16.ed. 2004	Manzano, José Augusto N.G.
Algoritmos: lógica para desenvolvimento de programação de computadores 26.ed. 2012	Manzano, José Augusto N. G.; Oliveira, Jayr Figueiredo de
Algoritmos: teoria e prática 10ª reimpressão 2007	Cormen, Thomas H. ... [et al.]
Algoritmos: teoria e prática 11ª reimpressão 2002	Cormen, Thomas H. ... [et al.]
Algoritmos: teoria e prática 2012	Cormen, Thomas H. [et al.]
Desenvolvimento de algoritmos e estruturas de dados	Terada, Routo
Estruturas de dados e algoritmos em C++	Frank, Adam

Fonte: Captura de tela de <<http://www.uems.br/biblioteca/livro.php>>

Figura 22 – Interfaces Web dos sistemas Busca Livro, UEMS e UFGD



The screenshot shows the UFGD library website interface. At the top, there is a logo for 'UFGD Net' and the text 'Sistema Bibliotecário - UFGD'. Below this, there is a search bar with the text 'algoritmos' and a search button. The results are displayed in a list format, showing the title of the book, the author, and the publisher. The number of results is 59.

Algoritmos estruturados: programacao estruturada de computadores
 FARRER, Harry. Algoritmos estruturados: programacao estruturada de computadores. 3. ed. Rio de Janeiro: LTC, 1999. 284p. [Livro]

Dominando algoritmos com C
 LOUDON, KYLE. Dominando algoritmos com C. . Rio de Janeiro: Ciencia Moderna, 2000. 580p. [Livro]

Introducao a programacao: 500 algoritmos resolvidos
 LOPES, Anita; GARCIA, Guto. Introducao a programacao: 500 algoritmos resolvidos. Rio de Janeiro: Elsevier, 2002. 469p. [Livro]

Graphs: theory and algoritmos
 THULASIRAMAN, K; SWAMY, M. N. S. Graphs: theory and algoritmos. New York, EUA: John Wiley & Sons, 1992. 460p. [Livro]

Algoritmos
 SALVETTI, Dirceu Douglas; BARBOSA, Lisbete Madsen. Algoritmos. Sao Paulo, SP: Makron Books do Brasil, 1998. 273p. [Livro]

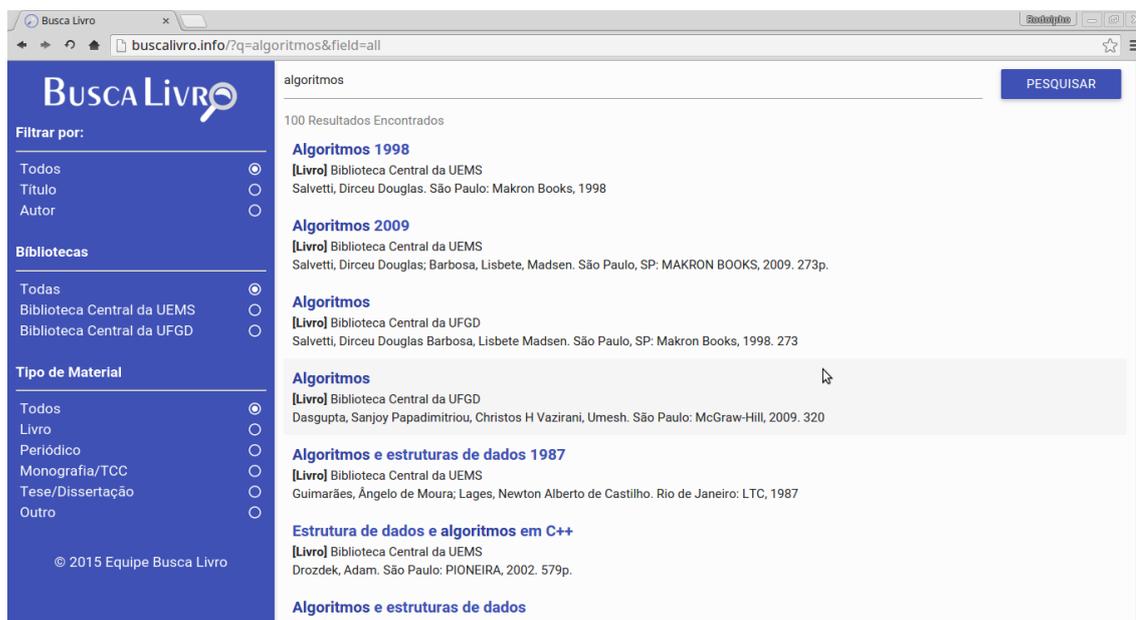
Projeto de algoritmos: com implantacoes em Pascal e C
 ZIVIANI, Nivio. Projeto de algoritmos: com implantacoes em Pascal e C. 2. ed. Sao Paulo, SP: Pioneira Thomson, 2005. 552p. [Livro]

Algoritmos: fundamentos e pratica
 ©2013 COIN - DIDE | Portal da UFGD

Fonte: Captura de tela de <http://biblioteca.ufgd.edu.br/faces/consulta_resultado.xhtml?filtro=conteudo&termo=algoritmos>

que nas Figuras 29 e 30 os resultados não são satisfatórios de acordo com os requisitos do sistema de testes, que avalia dentre outras coisas a exibição do site em telas pequenas como tamanho da fonte, já na Figura 28 é observado o resultado satisfatório do sistema Busca Livro com todos os requisitos.

Figura 23 – Interfaces Web dos sistemas Busca Livro, UEMS e UFGD



Fonte: Elaborada pelos autores

Em relação aos resultados de busca, constatou-se que quando realizada uma pesquisa no sistema da biblioteca da UEMS pelo termo “Algoritmos Cormen”, se referindo ao livro *Algoritmos: teoria e prática* do autor Thomas H. Cormen, nenhum resultado é encontrado, pois o sistema não permite combinar autor e título na mesma busca. No sistema Busca Livro, as edições desse livro são os primeiros resultados exibidos, pois o sistema permite a combinação de atributos na busca. Outra característica do sistema Busca Livro é a utilização de análise de sinônimos através do *field SynonymFilterFactory*, que insere palavras relacionadas ao termo buscado, e ainda processa o morfema raiz, inserindo prefixos e sufixos buscando pelo mesmo termo em outro gênero, número e grau. Embora o número de resultados retornados seja maior com estes filtros, é realizada uma classificação por nível de similaridade com os termos buscados, quando mais próximo, melhor é seu posicionamento no *ranking* de resultados. A diferença na aplicação destes filtros pode ser notada quando é realizada uma busca pela palavra “amor”, no sistema da biblioteca da UFGD são retornados apenas 95 resultados, no Busca Livro são retornados 197 resultados.

Com relação ao desempenho do sites dos sistemas Busca Livro e das bibliotecas da UEMS e UFGD, aplicou-se os testes *Page Load Time*, *Total Page Size* e *Requests* através do site <<https://gtmetrix.com>>. Foram realizados no dia 23 de outubro de 2015 às 16:00 horas e o resultado está representado na Tabela 3. O sistema da UFGD teve o pior tempo de carregamento, devido ao tamanho da página e a grande quantidade de arquivos estáticos. O sistema da UEMS teve o menor tamanho e a menor quantidade de arquivos estáticos, mas seu carregamento se demonstrou lento em virtude da baixa velocidade de transferência do seu servidor. O sistema Busca Livro apresentou o melhor

tempo de carregamento, realizando o carregamento aproximadamente na metade do tempo do sistema da UFGD.

Tabela 3 – Comparativo do Teste de Desempenho dos serviços de buscas

Teste	UEMS	UFGD	Busca Livro
<i>Page Load Time</i>	3.2s	4.5s	2.4s
<i>Total Page Size</i>	61.7KB	805KB	675KB
<i>Requests</i>	16 <i>requests</i>	36 <i>requests</i>	20 <i>requests</i>

Fonte: Elaborada pelos autores

Além das análises apresentadas anteriormente sobre a interface, resultados de busca e desempenho da aplicação, foi possível verificar a usabilidade e os recursos oferecidos pelo Busca Livro por meio da pesquisa realizada com os usuários.

Em um primeiro momento foi aplicada uma pesquisa piloto, entre os dias 28 e 30 do junho de 2015, com um questionário *online* e contou com uma amostra de 18 usuários. Eles testaram o sistema através do site <<http://buscalivro.info>> e do aplicativo Busca Livro. Por meio da análise dos resultados, as questões foram melhoradas e reorganizadas, formando um novo questionário e resolvendo problemas encontrados na primeira versão do sistema. O novo questionário contou com 11 perguntas de formatos variados, divididas em duas seções: Dados de Identificação e Dados Sobre o Tema. O questionário e a planilha de respostas da pesquisa se encontram nos Apêndices A e B respectivamente.

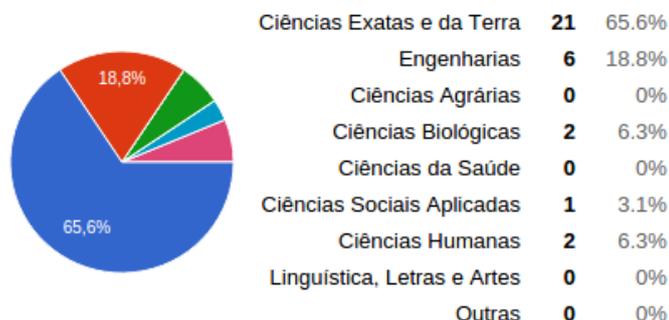
A partir do novo questionário, entre os dias 7 de outubro e 21 de outubro de 2015, foi aplicada a pesquisa contando com uma amostra de 32 usuários. O questionário foi disponibilizado de forma *online* e a divulgação foi feita por meio das redes sociais.

A seção de dados de identificação demonstrou que a amostra é composta de 68,8% de Homens 31,2% de mulheres, 84,4% tem entre 17 e 25 anos e apenas 11,6% tem mais de 25 anos. Quanto a área do curso de graduação a maioria absoluta, 65,6% é da área de Ciências Exatas e da Terra, em segundo lugar engenharias 18,8% e o restante distribuído entre Ciências Biológicas, Humanas e Sociais, conforme demonstrado na Figura 24.

Através de uma questão de múltipla escolha, os usuário foram questionados sobre quais dispositivos são utilizados para acessar a Internet na universidade. O *Smartphone* é o dispositivo mais utilizado pelos entrevistados, 90,6% o utilizam para acessar a Internet na faculdade, e 65,6% utilizam Computador ou *Notebook*. Dos que possuem *Smartphone*, 93,8% utilizam o sistema operacional Android. Esse dado reforça a necessidade de um serviço de busca nos acervos que funcione de maneira satisfatória em dispositivos móveis e justifica a escolha do desenvolvimento desse aplicativo móvel para tal sistema operacional.

Outra questão abordada foi em relação aos problemas enfrentados para realizar

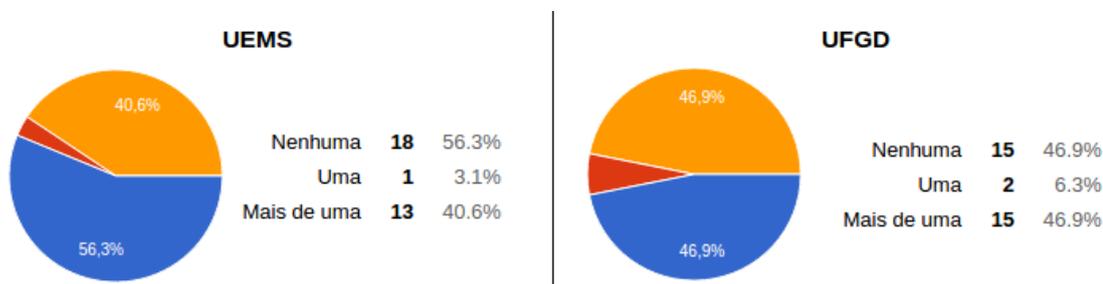
Figura 24 – Resultados: Área do curso apontada pelos usuários



Fonte: Elaborada pelos autores

buscas nos acervos das bibliotecas da UEMS e da UFGD. O gráfico da Figura 25 mostra que 40,6% responderam que já tentaram utilizar o sistema da UEMS e o encontraram indisponível, e 46,9% responderam que já encontraram o sistema da UFGD indisponível. Este é um número relevante, dado o grau de dependência dos usuários das bibliotecas em relação ao sistema.

Figura 25 – Resultados: Indisponibilidade dos sistemas apontada pelos usuários



Fonte: Elaborada pelos autores

Uma das questões buscava identificar qual o material de maior interesse dos usuários dos sistemas, 100% dos entrevistados escolheram como principal objetivo de suas buscas o livro, e esse dado justifica o seu maior peso no cálculo de relevância dos resultados. As informações mais relevantes para os usuários são demonstradas na Figura 26, onde podemos destacar os dados (Título, Autor, Editor) com 68,8% e a informação sobre a disponibilidade com 65,6%.

Para finalizar a análise da pesquisa, a Figura 27 mostra as vantagens apontadas pelos usuários na utilização do sistema Busca Livro. As principais são as possibilidade de comparar os resultados dos dois acervos, com 68,8% e a facilidade de realizar apenas uma busca para os dois acervos, com 65,6%.

Com as análises realizadas foi possível verificar a importância e relevância do Busca

Figura 26 – Resultados: Informações mais relevantes apontadas pelos usuários



Fonte: Elaborada pelos autores

Figura 27 – Resultados: Vantagens apontadas pelos usuários



Fonte: Elaborada pelos autores

Livro para os usuários das bibliotecas.

5 CONCLUSÃO

O sistema Busca Livro se mostrou eficiente em realizar buscas efetivas nos acervos bibliográficos das universidades UEMS e UFGD. Por meio dele foi possível realizar buscas em ambos os acervos simultaneamente, bem como comparar esses resultados. O sistema pode ser acessado através de duas interfaces, uma Web e outra móvel para dispositivos Android.

Na primeira fase do projeto, o estudo teórico contribuiu para o embasamento dos conceitos idealizados no início, tais como *Web Scraping*, bancos de dados não-relacionais, indexadores de dados e motores de busca. Esses conceitos relativamente novos, não são encontrados na literatura, por isso foram utilizadas outras fontes de informação, tais como documentações *online* e artigos.

A partir do conhecimento adquirido no estudo teórico, o processo de implementação do mecanismo de coleta de dados foi dividido para as duas universidades e tiveram níveis de dificuldades diferentes. O sistema da UEMS apresentou um funcionamento mais simples, o que facilitou a implementação do *Web Scraping*, bastando basicamente acessar as páginas de informações de todos os itens do acervo, selecionando os dados relevantes, e os armazenando no banco de dados. O sistema da UFGD apresentou maiores dificuldades devido ao seu funcionamento dinâmico utilizando requisições assíncronas e com resultados limitados, necessitando assim mais etapas e maior tempo de execução para capturar todos os dados necessários.

Em seguida, o servidor de busca foi implementado utilizando o Solr, que forneceu recursos de indexação e busca eficientes para o trabalho, demonstrando ótimo desempenho e confiabilidade no armazenamento dos dados. Entretanto sua utilização apresentou dois desafios: o primeiro foi devido ao seu grande poder de processamento que demandou maiores recursos de *hardware*, sendo necessária a utilização de um servidor mais robusto; o segundo ocorreu devido ao seu grande número de recursos, sua configuração se torna complexa, exigindo a execução de vários testes até que fosse encontrada uma combinação de configurações satisfatórias para o trabalho.

Com os *Web Scrapings* e o servidor de busca prontos, foi iniciada a implementação da interface Web, cujo o principal foco foi a experiência de usuário, sendo aplicadas técnicas e conceitos estudados no referencial teórico. O *Material Design* se mostrou a melhor opção tanto no quesito de usabilidade quanto de desempenho, tornando a interface mais amigável e intuitiva. Um ponto importante na implementação foi a redução da quantidade de dados que o sistema utiliza, fazendo uso de requisições assíncronas que transportam apenas as informações, permitindo ao Busca Livro apresentar melhor desempenho no teste de *Page*

Load Time.

O processo de implementação do aplicativo Android foi relativamente rápido na elaboração dos algoritmos devido a utilização da IDE Android Studio. Também foi optado pela utilização dos conceitos de *Material Design* assim como na interface Web, proporcionando maior agilidade e facilidade na utilização, além de padronizar as interfaces do sistema. A maior dificuldade neste processo foi o aprendizado de grande parte das técnicas de implementação para Android, sendo necessário um estudo mais aprofundado do assunto.

Diante dos resultados obtidos através da aplicação dos questionários, pode-se afirmar que o sistema Busca Livro atingiu seu objetivo. As respostas mostraram que os recursos do sistema são compatíveis com as necessidades dos usuários das bibliotecas.

5.1 Sugestões e Recomendações

Na área em que o sistema Busca Livro está inserido, existem várias possibilidades de expansão e melhorias que podem ser acrescentadas. Essas melhorias vão desde novas funcionalidades até novas interfaces de acesso.

Nos mecanismos de coleta de dados podem ser acrescentados algoritmos capazes de realizar buscas na Web pela capa, sumário e versão digital (caso exista) dos itens. Com isso, ao acessar um item de uma busca em qualquer interface, o usuário poderia visualizar a capa daquele item, além de apresentar ainda um sumário e o documento digital completo em forma textual para consulta. Isso poderia melhorar os resultados das buscas, uma vez que o campo de armazenamento do sumário e do conteúdo pudessem ser acrescentados aos campos de busca do Solr.

Para o mecanismo de busca, é possível implementar funções que realizam correções na grafia da busca realizada pelo usuário, facilitando o processo de pesquisa em dispositivos com telas e teclados pequenos, o que muitas vezes resulta em erros de digitação. Com isso também seria possível sugerir assuntos relacionados com a busca realizada, melhorando assim a experiência do usuário com o sistema.

Se tratando de interfaces, é possível implementar aplicativos para os sistemas operacionais *Apple iOS* e *Windows Phone*, aumentando ainda mais o público atingido pelo sistema Busca Livro.

Com relação a recursos e funcionalidades, é possível adicionar a “Área do Usuário”, com cadastro e *login*, para que seja possível armazenar o histórico de buscas dele para acessos futuros e também criar uma “lista de desejos”. Com isso, é possível alertar quando um item dessa lista estiver disponível no acervo para empréstimo.

REFERÊNCIAS

- AGHAEI, S.; NEMATBAKHSH, M. A.; FARSANI, H. K. Evolution of the world wide web: From web 1.0 to web 4.0. *International Journal of Web & Semantic Technology (IJWesT)*, n. 3, p. 1–10, jan 2012. Disponível em: <<http://airccse.org/journal/ijwest/papers/3112ijwest01.pdf>>. Acesso em: 18/06/2015.
- ANDROID DEVELOPERS. *Dashboard*. 2015. Disponível em: <<https://developer.android.com/about/dashboards/index.html>>. Acesso em: 17/05/2015.
- BORGES, L. E. *Python para Desenvolvedores*. [S.l.: s.n.], 2014. ISBN 9788575224052.
- CELERY. *Celery Documentation*. 2009–2014. Disponível em: <<http://docs.celeryproject.org/en/latest/index.html>>. Acesso em: 01/10/2015.
- CENDÓN, B. V. Ferramentas de busca na web. *Ci. Inf. Brasília*, v. 30, n. 1, p. 39–49, jan/abr 2001. Disponível em: <<http://www.scielo.br/pdf/ci/v30n1/a06v30n1>>. Acesso em: 10/05/2015.
- CORDING, P. H. Algorithms for web scraping. *Technical University of Denmark, Kongens Lyngby*, 2011.
- GARGENTA, M.; NAKAMURA, M. *Learning Android: Develop Mobile Apps Using Java and Eclipse*. [S.l.]: O’Reilly Media, 2014. ISBN 9781449336257.
- GOURLEY, D. et al. *HTTP: The Definitive Guide*. [S.l.]: O’Reilly Media, 2002. (Definitive Guides). ISBN 9781449379582.
- HANRETTY, C. Scraping the web for arts and humanities. *University of East Anglia, Norwich*, 2013.
- HILL, R. et al. *Guide to Cloud Computing: Principles and Practice*. [S.l.]: Springer, 2012. (Computer Communications and Networks). ISBN 9781447146032.
- HOLOVATY, A.; KAPLAN-MOSS, J.; AL. et. *The Django Book*. 2 ed.. ed. [s.n.], 2009. Disponível em: <<http://www.djangobook.com/en/2.0/index.html>>.
- KUROSE, J.; ROSS, K. *Redes de computadores e a internet: uma abordagem top-down*. [S.l.]: Pearson, 2010. ISBN 9788588639973.
- LINDSLEY, D. *PySolr Repository*. 2008–2015. Disponível em: <<https://github.com/toastdriven/pysolr>>. Acesso em: 01/10/2015.
- MOMJIAN, B. *PostgreSQL: Introduction and Concepts*. [S.l.]: Addison-Wesley, 2001. ISBN 9780201703313.
- NASCIMENTO, A. E. M. do; VACARI, I. Uso do software livre solr para construção da camada de indexação e busca do módulo de consulta institucional do sistema ainfo. *Embrapa Informática Agropecuária - Resumo em anais de congresso (ALICE)*, n. 15367, p. 143–146, dez 2010. Disponível em: <<http://www.alice.cnptia.embrapa.br/handle/doc/868788>>. Acesso em: 18/05/2015.

- O'BRIEN, J. *Sistemas de informação e as decisões gerenciais na era da Internet*. [S.l.]: Saraiva, 2004. ISBN 9788502044074.
- PRESSMAN, R. S. *Engenharia de Software*. [S.l.]: McGraw Hill Brasil, 2009. ISBN 9788580550443.
- PYTHON SOFTWARE FOUNDATION. *lxml 3.4.4*. 2014. Disponível em: <<https://pypi.python.org/pypi/lxml/3.4.4>>. Acesso em: 04/05/2015.
- REITZ, K. *Requests*. 2015. Disponível em: <<http://docs.python-requests.org/en/latest/>>. Acesso em: 04/05/2015.
- RICHARDSON, L.; RUBY, S. *RESTful Web Services*. [S.l.]: O'Reilly Media, 2008. ISBN 9780596554606.
- SADALAGE, P. J.; FOWLER, M. *NoSQL Essencial: Um Guia Conciso para o Mundo Emergente da Persistência Poliglota*. [S.l.]: NOVATEC, 2013. ISBN 9788575223383.
- SCHILDT, H. *Java para Iniciantes*. 6 ed.. ed. [S.l.: s.n.], 2015. ISBN 9788582603376.
- SHELLY, G.; WOODS, D.; DORIN, W. *HTML: Comprehensive Concepts and Techniques*. [S.l.]: Cengage Learning, 2008. (Available Titles Skills Assessment Manager (SAM) - Office 2010 Series). ISBN 9781423927228.
- SILBERSCHATZ, A.; KORTH, H. F.; SUDARSHAN, S. *Sistema de Banco de Dados*. 6. ed. [S.l.]: Campus - Grupo Elsevier, 2012. ISBN 9788535245356.
- SILVA, M. S. *Ajax com jQuery*. [S.l.]: NOVATEC, 2009. ISBN 9788575221990.
- SMILEY, D.; PUGH, E. *Solr 1.4 Enterprise Search Server*. [S.l.]: Packt Publishing, 2009. (From Technologies to Solutions). ISBN 9781847195890.
- SOARES, M. dos S. Metodologias ágeis extreme programming e scrum para o desenvolvimento de software. *Revista Eletrônica de Sistemas de Informação*, v. 3, n. 1, 2004. ISSN 1677-3071.
- SOLINGEN, R. V.; BERGHOUT, E. *The Goal/Question/Metric Method: A Practical Guide for Quality Improvement of Software Development*. [S.l.]: McGraw-Hill, 1999. ISBN 9780077095536.
- WANG, A. et al. *Materialize*. 2015. Disponível em: <<http://materializecss.com/about.html>>. Acesso em: 04/07/2015.
- WORLD WIDE WEB. *W3*. 1989. Disponível em: <<http://info.cern.ch/hypertext/WWW/TheProject.html>>. Acesso em: 17/05/2015.

Apêndices

APÊNDICE A – QUESTIONÁRIO DE AVALIAÇÃO

O presente estudo visa avaliar o aplicativo Busca Livro. Solicita-se a gentileza de utilizar a versão beta do aplicativo Busca Livro: Web e/ou Mobile (<http://buscalivro.info>). Posteriormente, responder todas as questões.

Muito obrigado

1. DADOS DE IDENTIFICAÇÃO

- a) Área do seu curso:
 - i. Ciências Exatas e da Terra;
 - ii. Engenharias;
 - iii. Ciências Agrárias;
 - iv. Ciências Biológicas;
 - v. Ciências da Saúde;
 - vi. Ciências Sociais Aplicadas;
 - vii. Ciências Humanas;
 - viii. Linguística, Letras e Artes;
 - ix. Outras
- b) Gênero
 - i. Masculino;
 - ii. Feminino;
- c) Faixa Etária
 - i. 16 anos ou menos;
 - ii. 17 à 25 anos;
 - iii. 25 anos ou mais;

2. DADOS SOBRE O TEMA

- a) Quantas vezes você tentou realizar buscas e o serviço de busca da biblioteca da UEMS se encontrava indisponível?
 - i. Nenhuma;
 - ii. Uma;

- iii. Mais de uma;
- b) Quantas vezes você tentou realizar buscas e o serviço de busca da biblioteca da UFGD se encontrava indisponível?
 - i. Nenhuma;
 - ii. Uma;
 - iii. Mais de uma;
- c) Quais são as vantagens que você encontrou na utilização do aplicativo?
 - i. Comparar os resultados dos dois acervos;
 - ii. A interface apresenta melhor os resultados;
 - iii. Realizar apenas uma busca para os dois acervo;
 - iv. Resultados mais relacionados com a busca;
- d) Por quais dispositivo você acessa a internet na faculdade?
 - i. Celular/Smartphone;
 - ii. Tablet;
 - iii. Computador/Notebook;
 - iv. Outro;
- e) Qual o sistema operacional do seu Smartphone?
 - i. Android;
 - ii. iOS (iPhone);
 - iii. Windows Phone;
 - iv. Outro;
- f) Qual o tipo de material você mais busca?
 - i. Livro;
 - ii. Periódico;
 - iii. Monografia/TCC;
 - iv. Tese/Dissertação;
 - v. Outro;
- g) Quais os tipos de informações mais relevantes para você?
 - i. Localização (CDD e PHA);
 - ii. Dados (Autor, Editora);
 - iii. Disponibilidade (Número de exemplares disponíveis);
 - iv. Versão (Idioma, Edição, Ano de publicação);
 - v. Outro;

APÊNDICE B – RESPOSTAS DO QUESTIONÁRIO

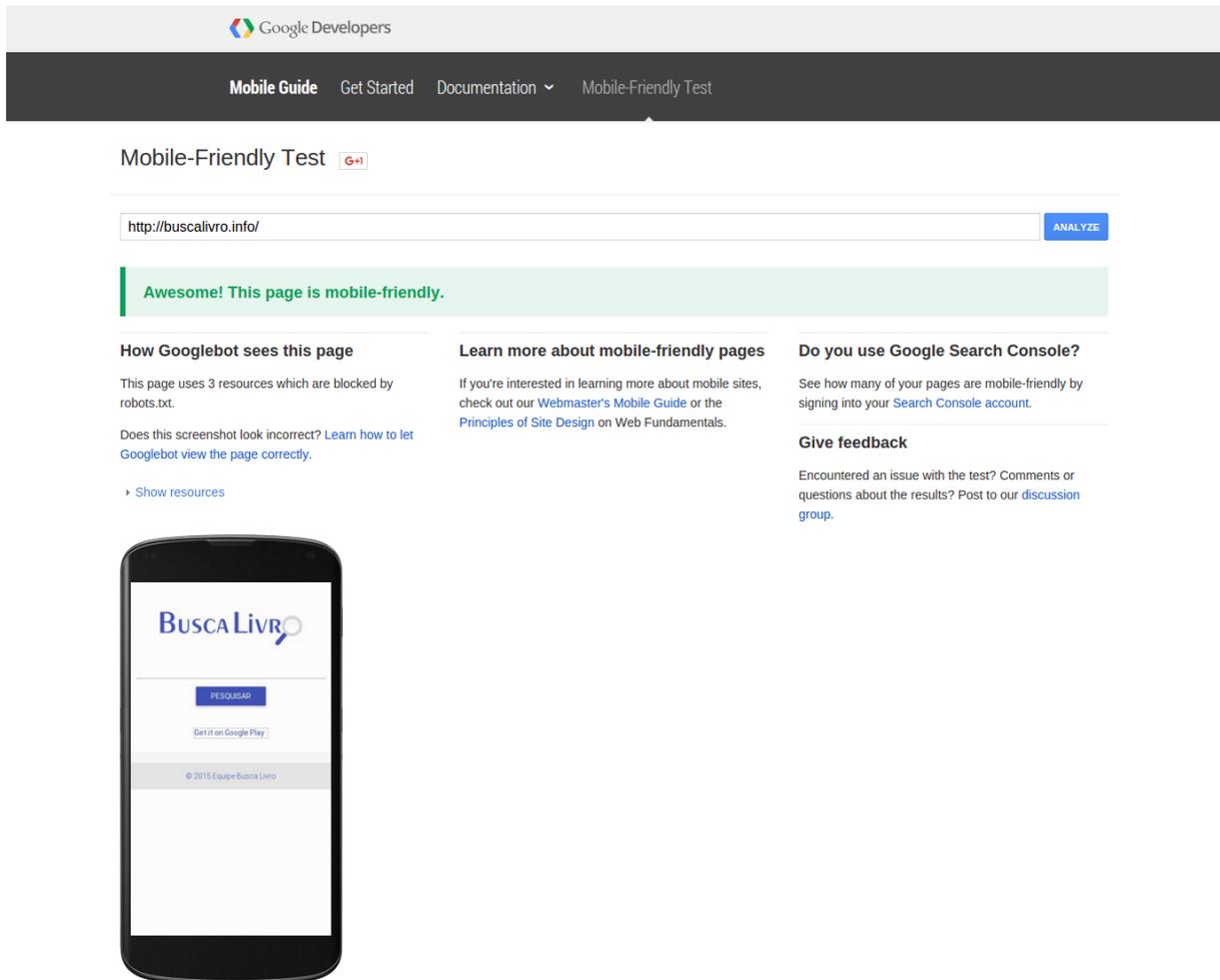
Tabela 4 – Respostas do questionário

Data e hora	Q1	Q2	Q3	Q4	Q5	Q6	Q7	Q8	Q9	Q10
07/10/2015 07:58:17	1	M	2	3	3	1, 2, 3, 4	1, 3	1	1	1, 3
07/10/2015 09:05:15	1	M	2	2	1	1, 3	1	1	1	2
07/10/2015 11:18:01	1	M	2	1	1	2, 3	1	1	1	1
07/10/2015 15:41:41	1	M	2	3	2	1, 3	1	1	1	2, 3
07/10/2015 19:29:00	1	M	2	3	3	3	1, 3	1	1	1, 2, 4
07/10/2015 19:40:43	1	M	2	1	1	1, 2, 3, 4	1, 3	1	1	2, 3, 4
07/10/2015 19:56:51	1	M	2	1	1	3	1, 3	1	1	2, 3, 4
07/10/2015 22:39:13	1	F	2	3	3	1, 2, 3, 4	1, 2	1	1	1, 3
07/10/2015 22:52:35	1	M	2	3	3	3	1	1	1	2, 3
07/10/2015 22:52:52	4	M	2	1	1	1, 2, 3	1, 3	1	1	2, 3, 4
07/10/2015 22:55:08	1	M	2	3	3	4	3	4	1	1, 2, 3
08/10/2015 18:26:34	1	F	2	3	3	1, 3, 4	1, 3	1	1	1, 2
08/10/2015 19:17:10	1	M	2	1	1	1, 3	1	1	1	1, 2, 3, 4, 5
08/10/2015 21:06:20	7	M	2	3	3	1	1, 3	3	1	1, 3
08/10/2015 21:59:00	2	M	2	3	3	1, 3	1, 3	1	1	1, 2, 3
08/10/2015 22:00:07	2	F	2	1	1	4	1	1	1	1
09/10/2015 00:12:25	7	M	2	1	3	1, 3	1, 3	1	1	1, 2, 3;
09/10/2015 09:33:22	2	F	2	3	2	1, 3	1, 3	1	1	1, 2, 3, 4
09/10/2015 09:37:59	2	M	2	3	3	2, 3	1, 3	1	1	1, 2, 3, 4
09/10/2015 10:06:04	2	M	3	1	1	1, 4	1	1	1	3
09/10/2015 10:25:16	1	F	2	1	3	2, 3	1, 3	1	1	2, 4
09/10/2015 15:05:08	1	F	3	1	1	1	1	1	1	3
09/10/2015 17:34:06	1	M	3	1	1	1, 2, 3	1, 3	1	1	1, 2, 3
09/10/2015 18:02:03	1	M	2	1	1	1, 3	1, 3	1	1	1, 2
09/10/2015 20:13:58	2	F	2	1	1	4	1, 3	1	1	1, 2
10/10/2015 17:53:23	1	M	2	1	1	1, 3	3	1	1	2
10/10/2015 17:53:27	1	M	2	3	3	4	3	1	1	4
11/10/2015 08:36:34	1	M	2	3	3	1, 2, 3, 4	1, 3	1	1	2
15/10/2015 21:20:18	4	M	2	1	3	1, 2, 3, 4	1, 3	1	1	2, 3
21/10/2015 10:25:31	6	F	2	1	3	1, 2, 3	1, 3	1	1	1, 2, 3

Fonte: Elaborada pelos autores

APÊNDICE C – TESTES DE INTERFACE PARA DISPOSITIVOS MÓVEIS

Figura 28 – Teste de interface para dispositivos móveis - Busca Livro



The screenshot shows the Google Developers Mobile-Friendly Test interface. At the top, there's a navigation bar with 'Mobile Guide', 'Get Started', 'Documentation', and 'Mobile-Friendly Test'. Below this, the page title is 'Mobile-Friendly Test' with a '+1' icon. A search bar contains the URL 'http://buscalivro.info/' and an 'ANALYZE' button. A green banner displays the message: 'Awesome! This page is mobile-friendly.' Below the banner, there are three columns of information:

- How Googlebot sees this page**: This page uses 3 resources which are blocked by robots.txt. Does this screenshot look incorrect? [Learn how to let Googlebot view the page correctly.](#) [Show resources](#)
- Learn more about mobile-friendly pages**: If you're interested in learning more about mobile sites, check out our [Webmaster's Mobile Guide](#) or the [Principles of Site Design](#) on Web Fundamentals.
- Do you use Google Search Console?**: See how many of your pages are mobile-friendly by signing into your [Search Console account](#).

Give feedback: Encountered an issue with the test? Comments or questions about the results? Post to our [discussion group](#).

At the bottom, a smartphone displays a mobile-optimized version of the 'Busca Livro' website. The mobile view features the 'Busca Livro' logo, a search bar with a 'PESQUISAR' button, a 'Get it on Google Play' button, and a footer with '© 2015 Equipe Busca Livro'.

Fonte: Captura de tela de <<https://www.google.com/webmasters/tools/mobile-friendly>>

Figura 29 – Teste de interface para dispositivos móveis - UEMS

Google Developers

Mobile Guide Get Started Documentation Mobile-Friendly Test

Mobile-Friendly Test G+

Not mobile-friendly

Page appears not mobile-friendly

- ✗ Text too small to read
- ✗ Links too close together
- ✗ Mobile viewport not set
- ✗ Content wider than screen

For details on which parts of the page are affected by these usability issues, see [Pagespeed Insights](#).

How Googlebot sees this page



Make this page mobile-friendly

Pick the option that describes how you created this site:

I used a CMS
I used software such as WordPress or Joomla.

Someone built this site for me
I hired someone to build this site and want advice for working with a developer.

I built this site myself
I built this site myself and understand how to code.

Do you use Google Search Console?
See how many of your pages are mobile-friendly by signing into your [Search Console account](#).

Give feedback
Encountered an issue with the test? Comments or questions about the results? Post to our [discussion group](#).

Fonte: Captura de tela de <<https://www.google.com/webmasters/tools/mobile-friendly>>

Figura 30 – Teste de interface para dispositivos móveis - UFGD

Google Developers

Mobile Guide Get Started Documentation Mobile-Friendly Test

Mobile-Friendly Test G+

<http://biblioteca.ufgd.edu.br/> ANALYZE

Not mobile-friendly

Page appears not mobile-friendly

- ✗ Text too small to read
- ✗ Links too close together
- ✗ Mobile viewport not set

For details on which parts of the page are affected by these usability issues, see [Pagespeed Insights](#).

How Googlebot sees this page



Make this page mobile-friendly

Pick the option that describes how you created this site:

I used a CMS
I used software such as WordPress or Joomla.
Next

Someone built this site for me
I hired someone to build this site and want advice for working with a developer.
Next

I built this site myself
I built this site myself and understand how to code.
Next

Do you use Google Search Console?
See how many of your pages are mobile-friendly by signing into your [Search Console account](#).

Give feedback
Encountered an issue with the test? Comments or questions about the results? Post to our [discussion group](#).

Fonte: Captura de tela de <<https://www.google.com/webmasters/tools/mobile-friendly>>