CENTRO UNIVERSITÁRIO UNIVATES
CENTRO DE CIÊNCIAS EXATAS E TECNOLÓGICAS
CURSO DE ENGENHARIA DA COMPUTAÇÃO

RAFAEL MALLMANN

UMA PLATAFORMA DE CMS ORIENTADA A SERVIÇO PARA A GESTÃO DE CONTEÚDO COM METAMODELOS

Lajeado, junho de 2016

RAFAEL MALLMANN

UMA PLATAFORMA DE CMS ORIENTADA A SERVIÇO PARA A GESTÃO DE CONTEÚDO COM METAMODELOS

Trabalho de Conclusão de Curso apresentado ao Centro de Ciências Exatas e Tecnológicas do Centro Universitário UNIVATES, como parte dos requisitos para a obtenção do título de bacharel em Engenharia da Computação.
Área de concentração: Centro de Ciências Exatas de Tecnológicas.

Orientador: Prof. Ms. Pablo Dall'Oglio

Lajeado, junho de 2016

RAFAEL MALLMANN

UMA PLATAFORMA DE CMS ORIENTADA A SERVIÇO PARA A GESTÃO DE CONTEÚDO COM METAMODELOS

A Banca examinadora abaixo aprova a Monografia apresentada na disciplina de Estágio Supervisionado III, na linha de formação específica em Administração de Empresas, do Centro Universitário UNIVATES, como parte da exigência para a obtenção do grau de Bacharela em Administração:

Banca Examinadora:

Prof. Ms. Pablo Dall'Oglio – orientador
Centro Universitário Univates
Mestre pela UNISINOS – Universidade do Vale dos Sinos

Prof. Juliano Dertzbacher, UNIVATES
Mestre pela Universidade Federal do Rio Grande do Sul – Porto Alegre, Brasil

Prof. Mouriac Halen Diemer, UNIVATES
Mestre pela Universidade Federal do Rio Grande do Sul – Porto Alegre, Brasil

Lajeado, junho de 2016

AGRADECIMENTOS

Ao professor orientador Ms. Pablo, pela paciência e dedicação durante todo o desenvolvimento deste trabalho.

À UNIVATES e aos colegas de trabalho que dia a dia tornam a vivência da profissão mais agradável.

Em especial, a minha família: meus pais, que sempre foram fundamentais na minha vida, e minha irmã, pelos bons momentos compartilhados durante todo o período da faculdade.

Por último, a minha noiva por todo o incentivo e paciência nessa jornada de estudos.

RESUMO

Desde o advento da WEB 2.0 nos anos 90, o conteúdo apresentado na internet ganhou cada vez mais destaque e importância, carecendo assim de ferramentas específicas para o gerenciamento de conteúdo na internet. Muitas ferramentas foram desenvolvidas com a finalidade de facilitar o processo de criação e administração de dados para sites dinâmicos na internet; atualmente, porém, ainda encontramos lacunas que podem ser preenchidas ou aprimoradas, casos em que os CMS disponíveis no mercado não apresentam o retorno esperado. O presente trabalho apresenta, além de um estudo de ferramentas de gerenciamento de conteúdos disponíveis, o desenvolvimento de uma plataforma WEB para esse fim; buscando suprir lacunas apresentadas nas plataformas de mercado como facilidade de desenvolvimento de cadastros dinâmicos e comunicação isolada com o sistema de gerenciamento. A aplicação proposta oferece uma comunicação com demais funcionalidades por intermédio de *Web Services* e se utiliza de uma estrutura dinâmica de metamodelos para atender às diferentes estruturas de dados apresentadas em sites na internet; esta composição de funcionalidades; torna a plataforma diferenciada das demais soluções de mercado disponíveis atualmente.

Palavras-chave: CMS. Metamodelo. Gerenciamento de conteúdo.

ABSTRACT

Since the advent of Web 2.0 in the 90s, the content presented on the Internet gained more prominence and importance, thus lacking specific tools for content management on the Internet. Many tools have been developed in order to facilitate the process of creating and administering data for dynamic sites on the internet today but still find gaps that can be filled or improved, where the CMS available in the market do not have the expected return. This study aims to present, as well as a study of available content management tools, the development of a web platform for this purpose. Seeking to fill gaps presented in market platforms such as ease of development of dynamic entries and isolated communication with the management system. The proposed application provides communication with other functionality via Web Services and uses a dynamic meta structure to meet the different data structures presented in websites; this composition features, makes the platform differentiated from other market solutions available today.

Keywords: CMS. Metamodel. Content management.

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE ABREVIATURAS

API:	*Application Programming Interface*
CETEC:	Centro de Ciências Exatas e Tecnológicas
CMS:	*Content Management System*
CRUD:	*Create, Read, Update e Delete*
CSS:	*Cascading Style Sheets*
ER:	Entidade e Relacionamento
HTML:	*HyperText Markup Language*
HTTP:	*Hypertext Transfer Protocol*
MDA:	*Model-Driven Architecture*
MDD:	*Model Driven Development*
MVC:	*Model, View e Controller*
OMG:	*Object Management Group*
PHP:	*Hypertext Preprocessor*
SAC:	Serviço de Atendimento ao Cliente
SLA:	*Service Level Agreement*
SOA:	*Service Oriented Architecture*
SOAP:	*Simple Object Access Protocol*
SQL:	*Structured Query Language*
SSL:	*Secure Socket Layer*
UML:	*Unified Modeling Language*

XML: *Extensible Markup Language*

WEB: *World Wide Web*

WSDL: *Web Service Definition Language*

SUMÁRIO

1 INTRODUÇÃO

No início da era da informatização das empresas, os dados ainda não eram utilizados em sua plenitude, e muitas vezes a sua gestão era considerada somente como uma forma de redução de custos. As empresas adotavam sistemas com o intuito de substituir o papel, não levando em consideração o valor agregado pelos dados armazenados. Com o passar das décadas, cada vez mais se percebeu a importância não só do seu armazenamento, mas também o seu valor como fonte de transformação em informações. Nos dias atuais, as informações presentes nos bancos de dados corporativos são grandes fontes de consulta para mineração de dados, descoberta de preferências de usuários, relação de vendas, dentre outros dados estratégicos fundamentais nas empresas (OBRIEN, 2001).

Com o passar dos anos, a informação passou a ter uma relevância não somente dentro das organizações, mas também fora delas, o que ocorreu com maior intensidade após a revolução da internet nos anos 1990. Nos dias atuais, existe um volume imenso de informações, e sua organização é de suma importância (YORAM, 2003).

No início dos anos 2000, o volume de informação disponibilizado na internet teve um grande aumento. Porém, a grande maioria dos *websites* eram, então estáticos, ou seja, as informações não eram armazenadas de maneira estruturada em bases de dados que permitissem fácil manipulação, gerenciamento e consulta (SANTOS, 2009).

Buscando suprir a demanda de gerenciamento de conteúdo na WEB, surgiram os Sistemas de Gerenciamento de Conteúdo (Content Management System CMS), inicialmente desenvolvidos por organizações que trabalhavam com publicação e edição de conteúdo on-

line, tais como jornais e revistas. Porém, até aquele momento, cada empresa desenvolvia um sistema especializado para atender suas demandas. Somente em 1995, a empresa de mídia WEB CNET passou a comercializar seu CMS, permitindo assim que as demais empresas pudessem realizar a tarefa de gerenciamento de conteúdo WEB, sem necessariamente depender de uma solução própria (KAMPFFMEYER, 2006).

Ao longo do tempo, novos projetos e plataformas com o objetivo de permitir o fácil gerenciamento de dados para criação de conteúdo on-line foram sendo desenvolvidos e difundidos no meio, dos quais podem ser citados o Drupal[1] (criado no ano 2000), o WordPress[2] (fundado em 2003) e o Joomla[3] (criado em 2005). Mais de 10 anos após o lançamento do Drupal, passaram a existir diversas plataformas e sistemas disponíveis para atuar como CMS:

Santos (2009) descreve os sistemas CMS como ferramentas de atualização on-line que exigem baixo ou nenhum nível de conhecimento técnico, abstraindo linguagens como o HTML, CSS e outros. A grande maioria das ferramentas de gestão de conteúdo são voltadas ao gerenciamento e disponibilização de dados relativos às informações de uso comum no meio WEB, permitindo facilmente a criação e disponibilização de notícias, formulários de contato, galerias de imagens, e outros elementos.

Ferramentas de CMS acabam por ser facilitadoras na gestão e atualização eficiente de portais *WEB*, sendo fundamentadas na abstração do conhecimento técnico e voltadas para profissionais específicos de cada área, sejam eles escritores, jornalistas, fotógrafos, dentre outros encarregados pela produção e atualização de conteúdo nos mais diversos veículos de comunicação on-line. Tais sistemas simplificam o trabalho de manter e atualizar complexas estruturas de dados apresentados na internet.

1.1 Motivação

A diversificação de conteúdo na WEB muitas vezes resulta em diferentes estruturas de dados necessárias para suportá-los. Buscando firmar-se e atingir um determinado público, os portais WEB apresentam cada vez mais conteúdos segmentados, de acordo com assunto ou público-alvo, o que gera maior diversidade nas estruturas de dados e registros.

[1] https://www.drupal.org/
[2] https://br.wordpress.com/
[3] http://www.joomla.org/

Na atualidade, existem diversas ferramentas de CMS que buscam ser flexíveis, e disponibilizar variados recursos na tentativa de atender diferentes cenários de utilização em relação à gestão de conteúdo. Entretanto, dificilmente uma ferramenta consegue atender a todos os requisitos impostos pela grande variedade de modelos específicos de dados existentes no ambiente organizacional.

Apesar da grande quantidade de soluções para gerenciamento de conteúdo, existe uma dificuldade de encontrar uma solução que seja simples e genérica o suficiente para atender às demandas de diferentes estruturas de conteúdo de um portal WEB, além de uma comunicação facilitada entre diferentes servidores ou aplicações. Os sistemas de CMS mais populares oferecem facilidades para criação de conteúdos geralmente direcionados para blogs, ou então oferecem componentes de baixo nível, os quais demandam a construção de uma camada extra de programação, que por sua vez exige maior tempo de desenvolvimento.

A maioria das ferramentas voltadas para o gerenciamento de conteúdo WEB não dispõem de APIs ou outras facilidades para integração com outros sistemas, sendo que estas, em geral, são moldadas para o uso direto da sua interface, e sempre que é necessário realizar integrações com sistemas de terceiros cabe ao desenvolvedor implementá-las.

Tendo em vista esse cenário, percebe-se que existe uma demanda por uma ferramenta que permita construir e disponibilizar conteúdos sobre modelos de dados dinâmicos, que se adaptem a distintas necessidades de negócio e, ao mesmo tempo, facilite a integração destas informações com outros sistemas. Tais necessidades são mais frequentemente encontradas em médios e grandes portais, com grandes fluxos de dados e diversidade de estruturas de cadastros.

Conciderando o contexto apresentado, o presente trabalho busca propor um modelo de dados dinâmico (metamodelo) para representação de informações para gestão de conteúdo on-line, que compreenda diferentes contextos de utilização. Propõe-se também a criação de uma camada de abstração por meio da disponibilização de serviços que permitam a manipulação deste modelo. O metamodelo e os serviços serão aplicados inicialmente para a formação de uma plataforma de CMS.

1.2 Objetivos

1.2.1 Objetivo geral

O objetivo deste trabalho é implementar uma plataforma de CMS por meio da disponibilização de um conjunto de *Web Services* que permitam a manipulação de um metamodelo de representação de conteúdo.

1.2.2 Objetivos específicos

Este trabalho busca atender os seguintes objetivos específicos:

- Desenvolver um metamodelo para representação de conteúdos que suporte a diversidade das estruturas de dados;
- Organizar uma aplicação para gerenciamento de conteúdo orientada a serviços sobre uma estrutura de metamodelo;
- Verificar a aplicação desenvolvida sob cenários pré-definidos;
- Validar a aplicação desenvolvida perante usuários reais;
- Analisar e interpretar os resultados obtidos.

1.3 Estrutura

Para a melhor compreensão do presente trabalho, a seguir é apresentada a estrutura de capítulos do trabalho:

No capítulo 2, é realizada uma revisão bibliográfica sobre gestão de conteúdo e evolução dos sistemas CMS. Este capítulo também aborda as tecnologias envolvidas nestas plataformas de gerência de conteúdo em relação a estruturas e comunicação dos dados, apresentando um estudo sobre banco de dados, metamodelos e comunicação via SOA.

O capítulo 3 apresenta a metodologia de pesquisa utilizada no presente trabalho, como se deu o desenvolvimento do mesmo e também como seus resultados foram apresentados e validados.

O capítulo 4 expõe uma análise das principais características e um comparativo entre as ferramentas de gerenciamento de conteúdo mais relevantes na atualidade.

No capítulo 5, detalha-se o desenvolvimento da ferramenta proposta. São apresentados sua arquitetura de funcionamento, seu modelo, bem como seus casos de uso, requisitos e tecnologias utilizadas.

No capítulo 6, o enfoque é a avaliação da plataforma de modo geral, trazendo dados sobre os avaliadores e uma análise dos resultados obtidos nos testes.

O capítulo 7 apresenta as considerações finais deste trabalho, bem como uma discussão das avaliações realizadas e trabalhos futuros a serem realizados.

2 REVISÃO BIBLIOGRÁFICA

Neste capítulo, serão apresentadas as referências bibliográficas utilizadas para o desenvolvimento do presente trabalho, buscando fundamentação e embasamento teórico. Cada item abordado assume importância para a compreensão do trabalho proposto. Para tal, serão apresentados os conceitos de Gestão de conteúdo, Ferramentas de gestão de conteúdo, Metamodelos e SOA.

2.1 Gestão de conteúdo

Santos (2009) define que o conteúdo deve ser visto como toda a informação útil aos usuários de modo geral, englobando assim tudo que é apresentado em um portal (imagens, relatórios, documentos, notícias, etc.). Porém, o conteúdo é muitas vezes interpretado de forma errônea, sendo confundido com o texto *stricto senso:* documentos, artigos, notícias; apesar de que a predominância ainda seja textual.

Portais WEB são definidos como uma vitrine embarcada, em que a empresa passa a divulgar conteúdo ao público, seja este aberto destinado a clientes, comunidade entre outros ou restrita, disponível mediante *login* ou senha, destinado a grupos fechados como sistemas internos de empresas, grupos de pesquisa, entre outros (SANTOS, 2009).

A mídia apresentada na internet sofreu uma grande alteração nos últimos anos, impulsionada pelo crescimento da internet corporativa no Brasil e no mundo, além de fatores, como a mudança da cultura digital, evolução dos navegadores e aprimoramento dos *hardwares* (SANTOS, 2009).

CGI (2014) descreve a utilização da internet pelas empresas para divulgação de diferentes tipos de dados, sendo conteúdos relacionados a produtos, serviços e a própria empresa, além de material relacionado à pós-venda e SAC.

O crescimento em massa do volume de conteúdo gerado para a internet, nos últimos anos, ocasionou uma demanda de organização deste conteúdo dentro das empresas, sendo que passaram a realizar uma segmentação e customização da informação, modelando seu conteúdo de acordo com o seu público-alvo; ou então, personalizando o ambiente e estabelecendo um diferencial atrativo para o usuário (SANTOS, 2009).

2.2 WEB 2.0

Bill Gates (1996) manifesta já na década de 90 a ideia de que "Um site com um bom conteúdo traz não só um melhor posicionamento nos motores de busca, mas também aumenta o engajamento do internauta.". Assim, o conteúdo disponível necessita ter qualidade de informação para se diferenciar perante os demais e também para atrair a atenção do leitor. O IBGE (2016) reforça a importância da divulgação de conteúdo no contexto WEB, apresentando resultados que demonstram que o acesso à internet cresceu 266% na comparação de 2005 a 2013, mostrando assim um grande aumento na abrangência de possíveis leitores.

A expansão da utilização da internet e o foco cada vez mais atrativo neste meio acabou por gerar uma demanda por websites cada vez mais atrativos e com seus conteúdos mais estruturados. Essa necessidade acabou por impulsionar o desenvolvimento de ferramentas que auxiliassem no desenvolvimento destes portais.

Buscando uma interface visual mais amigável, softwares que permitem a edição de um site sem ter que alterar os códigos HTML acabaram por surgir. Dentre os primeiros softwares desta linha, podem ser citados o Microsoft FrontPage, Macromidia, Dreamweaver e o Adobe GoLive. Estas ferramentas com um apelo mais visual permitiam edições de conteúdo através de botões e linhas de comando (SANTOS, 2009).

Nos anos 2000, a evolução e popularização da internet, agregada às novas ferramentas de desenvolvimento deram origem ao termo popular WEB 2.0, que Santos (2009) descreve este como uma nova geração da internet, onde os próprios usuários, por meio de ferramentas e

serviços, passam a produzir conteúdo, e não apenas consumi-lo. Como parte deste advento, podemos citar plataformas como Facebook, MySpace e Linkedin, que fizeram os usuários interagirem com o conteúdo.

Segundo Tarapanoff (2015), neste novo conceito de internet WEB 2.0, é necessária uma atenção diferenciada ao design apresentado, à busca de um conteúdo de relevância. E um foco específico ao usuário que se queira atingir. Buscando atender esses requisitos, é importante ter-se uma ferramenta de gestão de conteúdo para esta tarefa.

2.3 Ferramentas de gestão de conteúdo

A importância de ter um conteúdo original e atrativo na WEB incentivou a sua diversificação, trazendo assim diversas estruturas de conteúdo. Para suprir a demanda de atualização e produção de conteúdo na WEB, surgiram as ferramentas de gerenciamento de conteúdo, os CMS. Conforme Santos (2009, p. 24), "Os sistemas de gestão de conteúdo ganharam um papel mais central na organização, passando a ser uma ferramenta indispensável para o dia a dia do colaborador".

Dentre as principais vantagens de utilizar um CMS, podemos citar:

- Ganho de produtividade;
- Redução de riscos operacionais (Ciclo de vida de conteúdo);
- Descentralização;
- Aumento de segurança e monitoramento;
- Maior usabilidade;
- Flexibilidade nas possibilidades de apresentação de conteúdo.

Os CMS possuem vários níveis e têm foco em diferentes públicos, constituindo ferramentas de atualização on-line que exigem baixo ou nenhum nível de conhecimento técnico, abstraindo as linguagens WEB, como o HTML e CSS. Por exemplo, podem ser citadas as ferramentas de gerenciamento de *blogs* como o Tumblr e o WordPress. Em contrapartida, existem CMS's que possibilitam a customização não só da parte visual

apresentada, mas também da camada de desenvolvimento, possibilitando a criação de módulos de cadastro e exibição específicos atendendo assim às necessidades específicas de cada caso. Dentre as necessidades, podem ser apontados o Joomla, CodeIgniter e o WordPress, soluções que geralmente são instaladas em servidores próprios.

Ao longo do tempo, novos projetos e plataformas foram sendo desenvolvidas e difundidos nos meios, dos quais podem ser citados: Drupal (criado no ano 2000), WordPress (fundado em 2003) e o Joomla (desenvolvido em 2005), o qual recebeu em 2011 o prêmio de melhor CMS *Open Source*. Essa tecnologia evoluiu, e mais de 10 anos após o lançamento do Drupal, encontram-se temos disponíveis diversas plataformas e sistemas para atuar como CMS. Dentre as ferramentas de gerenciamento de conteúdo, são apresentadas diversas funcionalidades e diferentes características, que são melhor aprofundadas no capítulo 4 deste trabalho.

2.4 Metamodelos

Kühne (2006) descreve modelo como uma caracterização geral: "A descrição de alguma coisa". O autor também define que metamodelo é uma abstração de um sistema, seja ele real ou baseado em linguagem.

Conforme Neto (2013, p.31), "Um modelo é uma instância de um metamodelo, o que implica que um metamodelo é um modelo de outro modelo.".

Guedes (2012) afirma que "Um modelo captura uma visão de um sistema físico". Descreve um modelo como uma abstração do sistema, e este com um determinado propósito, seja para descrever aspectos comportamentais ou estruturais de um determinado software. O autor esclarece que o modelo determina o que é importante e deve ser contemplado e o que é irrelevante não deve estar contido nessa abstração. Desse modo o modelo descreve os aspectos do sistema físico que são importantes e relevantes ao seu propósito, no nível de detalhamento apropriado.

Uma análise literal sobre o significado de metamodelo pode ser feita com base na acepção do prefixo "meta", de origem grega, que indica "que engloba", "que ultrapassa", "que transcende". O prefixo "meta" é utilizado para identificar operações que são realizadas mais de uma vez.

Metamodelos são utilizados para definir determinadas linguagens, que são expressadas por modelos. Um metamodelos tem a importância de definir a semântica para modelar elementos dentro de um modelo sendo instanciado. Dessa forma, um modelo é uma instância de um metamodelo (GUEDES, 2012).

Neto (2013) descreve que metamodelos representam um nível superior de abstração. Este modelo deve ser composto por componentes conceituais, com base em uma rica estrutura semântica e estrutura lógica. Esses conceitos possibilitam uma melhor adaptação, comparação, análise e integração destes modelos.

2.4.1 Hierarquias e tipos

Kühne (2005) descreve que um sistema qualquer pode ser representado por um modelo, sendo esta uma ocorrência de um metamodelo; logo, é possível afirmar que o metamodelo é uma instância do metametamodelo e assim subsequentemente.

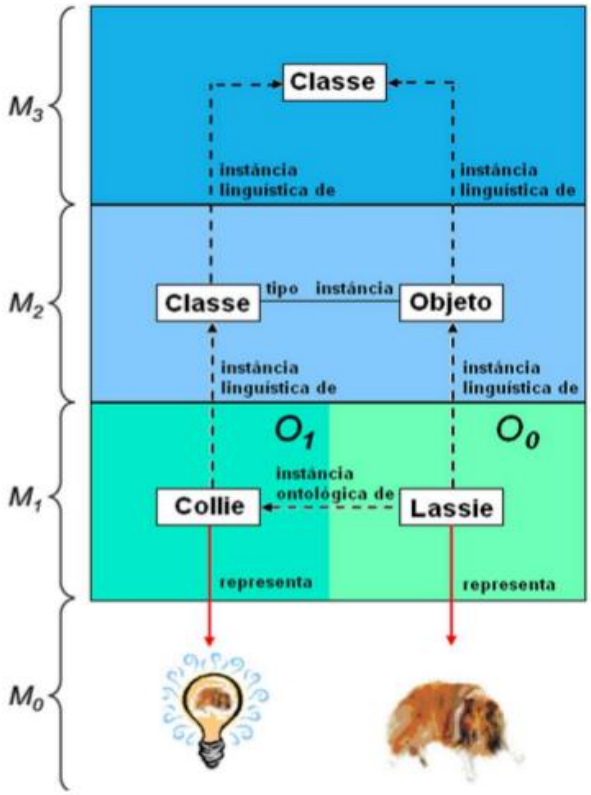
De acordo com os autores Atkinson e Kühne (2003a e 2003b), os objetos do metamodelo podem ser identificados de duas formas distintas de instanciação:

- Metamodelo linguístico: que se refere à definição da linguagem;
- Metamodelo ontológico: definição do tipo de objeto ou do domínio ao qual pertence.

Ambas as formas de instanciação são apresentadas de forma simultânea, trabalhando para uma localização precisa de um elemento ou objeto no espaço linguístico-ontológico.

A figura 1 ilustra a arquitetura MDD *(Model Driven Development),* onde é representado um metamodelo linguístico em quatro camadas horizontais de abstração, apresentando desde o menor nível de abstração (M0) até o maior (M3). As diferentes cores apresentadas na divisão vertical em M1 mostram a representação do metamodelo ontológico.

Figura 1 - Metamodelo Linguístico-ontológico



Fonte: Atkinson e Kuhne, 2003a.

2.4.2 Aplicações dos Metamodelos

No desenvolvimento de software, existe uma grande semelhança em diversas partes do programa. Oliveira (2001) defende que a utilização de técnicas que tornem possível a reutilização de trechos ou partes do programa geram um aumento de produtividade e também de qualidade de um sistema.

Os metamodelos buscam o desenvolvimento de uma aplicação genérica, de uma estrutura capaz, que possa ser aplicada a diferentes estruturas de dados e tipos de registros, com nenhuma ou poucas adaptações.

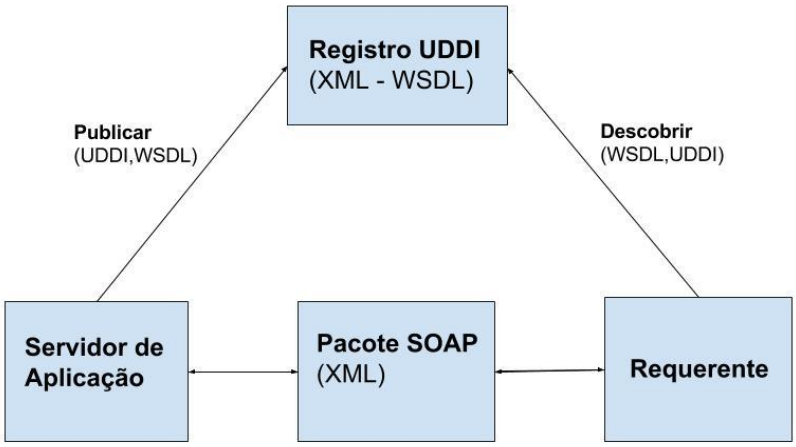
Segundo Pfleeger (2004), existem duas formas de reutilização: reutilização caixa-branca, onde são realizadas modificações de acordo com as necessidades do *software*; reutilização caixa-preta, em que toda a estrutura é reaproveitada sem modificações.

2.5 SOA e Web Services

De acordo com (BRIEN, MERSON, BASS 2007) a Arquitetura Orientada a serviço, do inglês Service Oriented Architecture(SOA), é uma metodologia de programação de interligação de *softwares* distintos, realizando essa comunicação de forma simples e confiável, sendo versátil e adaptável a diferentes situações. Já para o autor Saudate (2012, p.14), "SOA não é baseado em *Web Services*, mas sim, na parcela do sistema que possa ser acessado através de mecanismos externos ao próprio sistema.".

A arquitetura SOA é constituída por um serviço ou aplicação que fornece os dados, o cliente que acessa os dados, e um contrato de serviço que autentica a transação. Dependendo da definição de uso, o padrão SOA pode conter um elemento denominado corretor de serviço, que atua para manter a troca de informações confiáveis e persistentes (SAUDATE, 2012). A arquitetura SOA é descrita na figura 2:

Figura 2 - Arquitetura SOA *Web Service*



Fonte: Elaborado pelo autor (2016).

2.5.1 Web Service

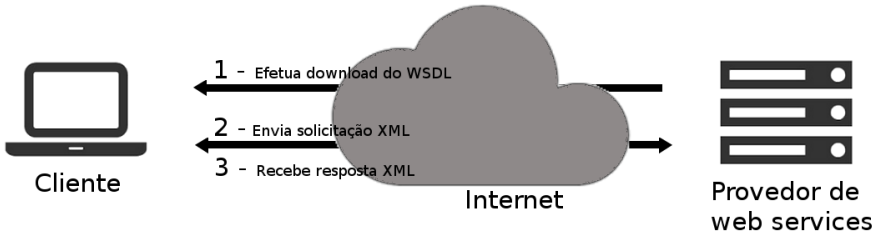
Web Service é tecnologia proveniente de grandes empresas como a IBM e W3C. Foi criado o padrão de comunicação SOAP (Simple Object Access Protocol), implementado através de *Web Services*, que, segundo Gomes (2009), é uma evolução dos modelos de computação distribuída utilizados para integrar diferentes aplicações.

Sendo uma comunicação independente da linguagem de programação utilizada, hardware ou sistema operacional, os *Web Services* têm como premissa a troca de dados através de um arquivo denominado como WSDL(Web Service Description Language), cuja finalidade é descrever um *Web Service*, sendo que esse arquivo tem o formato XML (GOMES, 2009).

2.5.1.1 Comunicação Web Service

A comunicação por *Web Service* entre diferentes aplicações é realizada pela troca de arquivos WSDL, em que, num primeiro momento, o cliente realiza o download do WSDL do provedor de *Web Service* e, em seguida, encaminha a solicitação de XML, onde, após processada é retornada ao cliente; toda transação é realizada através da internet (GOMES, 2009). A comunicação é ilustrada na figura 3:

Figura 3 - Comunicação por *Web Service*



Fonte: Adaptado de (GOMES 2009, p. 19).

2.5.1.2 Características dos Web Services

A solução de integração entre sistemas por meio de *Web Service* possui várias características distintas (SAUDATE, 2012). Podemos citar:

- Garantia de entrega: Como a transação entre as aplicações é realizada através do serviço de internet, este se torna diretamente proporcional à garantia de sucesso da comunicação. Para (BRIEN, MERSON, BASS 2007), é necessário observar o controle de transação, uma vez que esta é iniciada e finalizada em um determinado serviço; o autor também ressalta o controle de falha para reenvio de chamada, caso o retorno for falho;

- Interoperabilidade: por meio dos padrões como o Simple Object Access Protocol (SOAP) e *Web Service* Definition Language (WSDL), é possível a comunicação entre aplicações com diferentes linguagens de programação utilizadas;

- Desempenho: não é indicado para aplicações que necessitam de resposta em tempo real. Por utilizar em protocolos como Extensible Markup Language (XML) e consultarem servidores disponíveis, aumentam a latência das requisições;

- Segurança: considerada uma comunicação segura, o padrão *Web Service* utiliza-se de protocolos confiáveis como http, SSL, além de criptografia;

- Disponibilidade: geralmente acordadas com um nível de segurança (Service Level Agreement (SLA)), as requisições por *Web Services* definem os tempos de disponibilidade, máximo de espera entre outros;

- Modificabilidade: *Web Service* apresentam um bom retorno, desde que as alterações não impactem a interface do serviço em questão;

- Testabilidade: a realização de teste em aplicações que se utilizam de *Web Services* torna-se mais danosa devido ao fato de partes do código estarem em diferentes servidores, englobando permissões de acessos e rastreamento de execução;

- Escalabilidade: não é fornecida nenhuma forma de aumentar a escalabilidade pela tecnologia de serviços *Web Services*, mas, em contrapartida, o aumento de desempenho em hardware gera um resultado positivo.

2.5.2 Exemplos de aplicação

A solução de integração através de SOA *Web Service*, por meio de suas características mencionadas no capítulo 2.5.1.2, acabou por se popularizar, sendo difundida atualmente como um meio robusto de comunicação entre duas aplicações distintas.

Atualmente, faz-se uso da comunicação por *Web Services* para os mais variados fins, podendo ser citados como exemplos a obtenção de dados da bolsa de valores Bovespa e a busca de usuários na rede social Linkedin; outro popular exemplo desta aplicação é a utilização dos recursos de geolocalização do sistema Google Maps da Google.

3 METODOLOGIA

Conforme Marconi e Lakatos (2003), para existir a ciência, é necessária a utilização de métodos científicos. Assim, compreende-se que podem ser considerados como ciência os estudos que empregam métodos científicos, mesmo que os métodos científicos não sejam empregados em todos os estudos. A metodologia científica transcende as regras de execução, exigindo esta disciplina, criatividade e organização (GOLDENBERG, 2000). A seguir será apresentado o método científico abordado neste trabalho.

3.1 Delineamento

Para Wainer (2007), a pesquisa em Ciência da Computação envolve na maioria dos casos a construção de um programa, de um modelo, de um algoritmo ou de um sistema novo. Já Fonseca (2002) descreve que "a pesquisa possibilita uma maior aproximação e entendimento da realidade".

Buscando satisfazer a afirmação anterior, que envolve a construção de um sistema, o presente trabalho desenvolverá uma plataforma de CMS orientada a serviço, o que envolverá a construção de um programa, e de um modelo. Contudo, Wainer (2007) defende que a simples criação de um protótipo não é suficiente para a sua correta avaliação, devendo este também ser avaliado de forma metodológica, constituindo um processo rigoroso.

O presente trabalho propõe a elaboração de uma ferramenta, cujo foco seja a gerência e manutenção de conteúdos dinâmicos com base em metamodelos. O objetivo é provar sua

eficiência e desempenho em um ambiente real de desenvolvimento, buscando-se comprová-lo através de Pesquisa Exploratória, conforme Gil (2010, p. 43),

> Esse tipo de pesquisa tem como objetivo proporcionar maior familiaridade com o problema com vistas a torná-lo explícito ou a construir hipóteses. Envolve levantamento bibliográfico; entrevistas com pessoas que tiveram experiências práticas com o problema pesquisado; análise de exemplos que estimulem a compreensão. Assume, em geral, as formas de Pesquisas Bibliográficas e Estudos de Caso.

Para este trabalho optou-se por uma abordagem de pesquisa qualitativa, buscando a opinião de especialistas no assunto para um parecer técnico sobre a solução proposta. Segundo Goldenberg (1997 p. 34), "a pesquisa qualitativa tem como foco a compreensão de uma amostra/grupo específico, sendo esta pesquisa despreocupada em relação à representatividade numérica". Já para Wainer (2007, p. 28) "métodos qualitativos em Ciência da Computação são métodos que se caracterizam por um estudo aprofundado de um sistema no ambiente onde ele está sendo usado, ou, em alguns casos, onde se espera que o sistema seja usado".

Como meio de coleta de dados, será elaborado um questionário, a ser aplicado a especialistas para a avaliação da ferramenta. Estes especialistas responderão a questões fechadas e abertas, de modo a se obter um parecer sobre a ferramenta desenvolvida e suas funcionalidades. Segundo Barros (2000, p. 90), "O questionário é o instrumento mais usado para o levantamento de informações. Não está restrito a uma determinada quantidade de questões, porém aconselha-se que não seja muito exaustivo, desanimando o pesquisado".

Buscando uma melhor análise deste trabalho e um maior contexto de fontes de informação, o autor adotou uma abordagem por pesquisa experimental, que, segundo Gerhardt e Silveira (2009 p. 36), "pode ser desenvolvida em laboratório (onde o meio ambiente criado é artificial) ou no campo (onde são criadas as condições de manipulação dos sujeitos nas próprias organizações, comunidades ou grupos". Neste passo, a ferramenta e os modelos são alimentados com dados fictícios e os testes e validações realizados em ambiente controlado.

A avaliação da plataforma procura simular dados e um ambiente real; visando assim se aproximar de uma pesquisa de campo. Para Gerhardt e Silveira (2009, p. 36), "a pesquisa de campo caracteriza-se pelas investigações em que, além da pesquisa bibliográfica e/ou

documental, se realiza coleta de dados junto a pessoas, com o recurso de diferentes tipos de pesquisa".

Utilizando-se de pesquisa experimental, porém baseado em dados reais, o trabalho busca assim um aprimoramento e uma melhor análise de seus requisitos e funcionalidades, através de uma avaliação o mais próximo possível de um cenário real.

Objetivando o desenvolvimento de uma plataforma que seja estruturada sobre uma fundamentação teórica, a pesquisa bibliográfica assume grande importância, e com esta, se espera o desenvolvimento de uma solução que respeite os conceitos e princípios abordados neste *software*. Santos (1999, p. 29) caracteriza a pesquisa bibliográfica assim:

> O conjunto de materiais escritos/gravados, mecânica ou eletronicamente, que contêm informações já elaboradas e publicadas por outros autores é uma bibliografia. São fontes bibliográficas os livros (de leitura corrente ou de referência, tais como dicionários, enciclopédias, anuários etc.), as publicações periódicas (jornais, revistas, panfletos etc.), fitas gravadas de áudio ou vídeo, páginas de web sites, relatórios de simpósios/seminários, anais de congressos etc. A utilização total ou parcial dessas fontes é o que caracteriza uma pesquisa como bibliográfica.

4 ANÁLISE DE SOLUÇÕES DE MERCADO

A seguir, será apresentada uma análise sobre as mais populares ferramentas e *frameworks* de desenvolvimento WEB, que se caracterizam como CMS. Como critério de seleção, optou-se pelos sistemas que podem ser instalados localmente e que possuem estruturas que possam ser alteradas ou com possibilidades de desenvolvimento de uma camada extra de programação, ou mesmo possam ser estendidas por meio de *plugins*, e também forneçam um painel administrativo. Para um comparativo mais amplo, procurou-se analisar ferramentas de linguagens de programação conceituadas como PHP, JavaScript, Ruby e Phyton.

4.1 BrowserCMS

Baseado na linguagem Ruby, que vem ganhando muitos adeptos nos últimos anos, a plataforma BrowserCMS vem se popularizando como uma opção para os desenvolvedores (BROWSERCMS, 2015).

O BrowserCMS, que ultrapassou 100 mil downloads, apresenta opções de módulos que podem estender e ampliar as funcionalidades da ferramenta, e, atualmente, encontra-se na versão 3.5.0 (BROWSERCMS, 2015).

4.2 Django CMS

O projeto *Open Source* Django CMS, criado em 2009, baseia-se no *framework* Django. Este projeto busca simplificar a atualização de conteúdo oferecendo um editor *frontend* com recursos de "clique e escreva", além de um painel administrativo para gestão do *website*, a plataforma disponibiliza uma biblioteca de *plugins* que podem ser integrados, e são desenvolvidos pela comunidade ativa, que conta com mais de 450 mil downloads (DJANGO, 2015).

4.3 Drupal

Um dos primeiros CMS a serem lançados, em janeiro de 2001, foi o Drupal, cujo nome provém da palavra holandesa "Druppel", de significado gota, e seu responsável foi Dries Buytaert (DRUPAL, 2015).

Considerada uma solução bastante completa e robusta, atualmente na versão 8, conta com mais de 24 mil *plugins* e já ultrapassa os 15 milhões de downloads; a plataforma é encontrada em 1,6% do total de sites disponibilizados na rede, incluindo o The Economist, The White House e MTV (DRUPAL, 2015).

4.4 Joomla

Ferramenta criada a partir do CMS Mambo, teve seu lançamento em setembro de 2005, divulgada em diversos países e línguas. Possui sua estrutura em módulos e possibilita uma personalização muito grande de seus *websites* (JOOMLA, 2015).

Somando 2,7% dos sites da WEB, o Joomla se encontra na versão 6, atualmente com mais de 7 mil *plugins*, e disponibiliza vários recursos e funcionalidades. A plataforma é bastante explorada para sites de médio e grande porte, entre eles estão o Linux.com e o IHOP (JOOMLA, 2015).

4.5 KeystoneJS

Considerado o segundo CMS mais famoso sobre Node.js, o KeystoneJS é um *framework Open Source* voltado a desenvolvedores. Com o foco voltado ao desenvolvimento ágil, a ferramenta disponibiliza facilitadores, como criação automática do CRUD (criar, ler, atualizar e excluir) e validadores de formulários nativos e é voltada ao desenvolvimento com o banco de dados MongoDB. Apesar de estar apenas na versão 0.3.13, a ferramenta já se encontra presente rm grandes empresas, como a Continental, a Sony e Vodafone (KEYSTONEJS, 2015).

4.6 Mezzanine

Mezzanine é uma plataforma extensível e flexível de gerenciamento de conteúdo, que se assemelha bastante com a ferramenta WordPress. Voltada à comunidade desenvolvedora Phyton, essa ferramenta se baseia no *framework* Django (MEZZANINE, 2015).

A ferramenta possui, entre outros, o diferencial de a maioria de seus *plugins* virem instalados por padrão, tornando-a mais robusta e integrada sem necessárias instalações extras. Também é desenvolvida de modo a estimular *hacking* sobre o código, e atualmente, encontra-se na versão 4.0.1 (MEZZANINE, 2015).

4.7 PencilBlue

O recente projeto PencilBlue, lançado em junho de 2014 traz uma proposta inovadora buscando ser totalmente compatível com dispositivos móveis. O *framework* se baseia em MongoDB e Bootstrap, e pode ser utilizado juntamente com angularjs e jQueryUI (PENCILBLUE, 2015).

A ferramenta é expansível e tem à disposição alguns *plugins*, como diferenciais pode-se mencionar que a plataforma possui alteração de dados relacionais diretamente pelo painel e já possui suporte nativo a *clusters* de servidores, para um melhor desempenho na nuvem (PENCILBLUE, 2015).

4.8 Refinery CMS

Projeto *Open Source* fundado em 2004, o *framework* Refinery CMS é uma opção para desenvolvimento de sites em Ruby e banco de dados SQLite, disponibilizando uma biblioteca de *plugins* e um painel de gerenciamento. A ferramenta abrange desde *blogs* a portais mais elaborados (REFINERYCMS, 2015).

4.9 WordPress

Criado por Matt Mullenweg a partir do sistema B2/Cafelog, o WordPress foi lançado em maio de 2003, inicialmente como um simples sistema para *blogs* e atualmente já está consolidado como um CMS robusto, com mais de 140 milhões de *downloads*. Um dos principais diferenciais fica por conta da facilidade na adaptação de *layouts*, mérito de seu flexível sistema de *templates*, com foco no design e seus mais de 27 mil *plugins* à disposição (WORDPRESS, 2015).

Atualmente na versão 3, é uma das ferramentas de gerenciamento de conteúdo mais aceitas no mundo, com 14,3% dos *websites* se utilizando de sua estrutura. Dentre os grandes sites, podemos citar MTV Newsroom, The Ford Story (WORDPRESS, 2015).

4.10 Comparativo entre ferramentas de gestão de conteúdo

Após a apresentação de diversas ferramentas de gestão de conteúdo, é possível, a comparação entre elas. Para tal, é necessário estabelecer critérios de comparação, que busquem contemplar os itens de enfoque deste trabalho, tornando este comparativo relevante.

A tabela 1 busca criar uma representação das principais ferramentas de mercado com relação às funcionalidades e opções de cada solução. Nas linhas estão dispostas as soluções, e, nas colunas, os critérios de comparação, que serão detalhados posteriormente.

Tabela 1 - Comparação de ferramentas de mercado x funcionalidades

	C1	C2	C3	C4	C5	C6	C7	C8	C9	C10
BrowserCMS	Ruby	S	N	S	N	S	N	S	N	S
Django CMS	Phyton	S	N	S	N	S	N	S	N	N
Drupal	PHP	S	N	S	S	S	S	S	S	S
Joomla	PHP	N	N	S	N	S	S	S	S	N
KeystoneJS	JavaScript	S	S	N	S	N	S	S	S	S
Mezzanine	Phyton	S	N	S	N	N	N	S	S	N
PencilBlue	JavaScript	S	N	N	S	S	N	S	N	S
Refinerycms	Ruby	N	S	N	N	N	N	S	N	S
WordPress	PHP	N	N	N	S	N	S	S	S	N

Fonte: Elaborado pelo autor (2016).

Os critérios de comparação utilizados são:

C1 – Linguagem utilizada pela aplicação?

C2 – Baseado em *framework*?

C3 – Seus arquivos são organizados em estrutura de pastas hierárquicas, como por exemplo uma estrutura MVC?

C4 – Disponibiliza suporte para sua utilização com diferentes bancos de dados?

C5 – Sua estrutura de armazenamento de dados é dinâmica, ou seja, possibilita a criação de diferentes cadastros sem alteração na estrutura do banco de dados em si?

C6 – Possui ferramentas que permitam a edição ou administração de dados por linha de comando?

C7 – Disponibiliza API para integração por *Web Service*?

C8 – A ferramenta permite que sejam acoplados *plugins* para adicionar funcionalidades extras?

C9 – A ferramenta disponibiliza ou permite a criação de temas para a personalização da interface?

C10 – A ferramenta permite a criação de cadastro (CRUD) sobre demanda?

Analisando os resultados apresentados na tabela 1, pode-se verificar a diversidade de ferramentas de CMS disponíveis atualmente no mercado. As ferramentas apresentam diferentes soluções e arquiteturas, e podem ser verificados diversos enfoques para cada plataforma, como o desenvolvimento de blogs, sites complexos, sistemas, plataformas para usuários leigos, entre outros. Algumas apresentam soluções mais completas e robustas como o WordPress e Drupal; outras, plataformas mais simples, como o Mezzanine. Apesar de algumas soluções de mercado atenderem a praticamente todos os quesitos analisados como o Drupal e KeistoneJS, estas plataformas disponibilizam os recursos avaliados através de *plugins* e APIs à parte, não integrando a instalação inicial, diferentemente da plataforma proposta, que busca já oferecer todos os recursos na sua instalação padrão.

Os critérios levantados na tabela 1 servirão não somente para a comparação entre as soluções de mercado, mas também como métricas para o desenvolvimento da solução proposta. Os critérios serão retomados ao final deste trabalho, onde a plataforma proposta será comparada com as demais, a fim de se discutir a sua importância como tema de pesquisa.

5 TRABALHO PROPOSTO

O presente capítulo tem por finalidade apresentar as informações referentes à plataforma desenvolvida neste trabalho. Serão abordados a visão geral e escopo da plataforma, sua estrutura, bem como seu diagrama ER, além dos requisitos que a plataforma pretende cumprir e seus detalhamentos, buscando apresentar de forma geral as principais características do projeto proposto.

5.1 Visão Geral

Este trabalho propôs desenvolver e implementar uma plataforma de gerenciamento de conteúdo, visando à otimização do processo de desenvolvimento de cadastros, e que seja responsável por criar as telas de CRUD (Create, Read, Update e Delete), além de, permitir opções de validação e customização dessas interfaces.

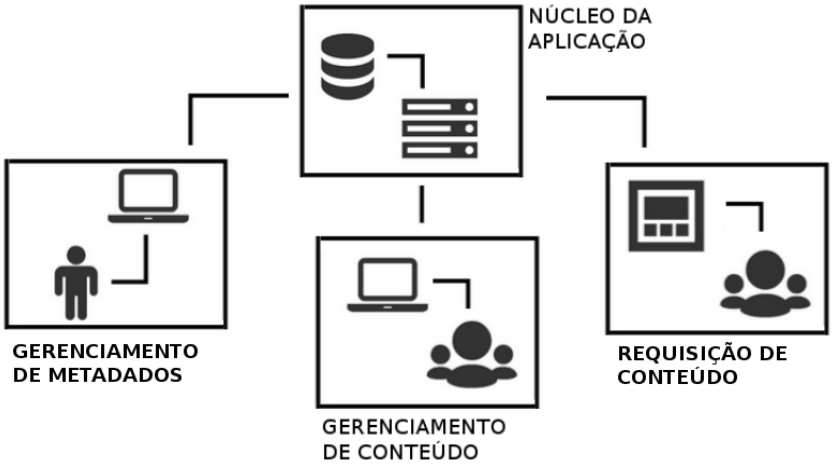
Disponibilizada uma interface, onde o usuário administrativo irá configurar os tipos de dados de que o registro desejado seja composto, bem como as validações e customizações esperadas. Com base nesses dados, a plataforma irá apresentar as interfaces de CRUDs, as quais serão utilizadas pelos usuários responsáveis pela atualização destes dados.

A ferramenta proposta busca abstrair as estruturas de dados, bem como as linguagens SQL, de modo a se utilizar de uma estrutura genérica (metamodelo), que seja capaz de suportar os diversos tipos de dados utilizados em um portal WEB.

Um dos diferenciais deste projeto é a sua comunicação, que será realizada por meio de um conjunto de *Web Services*, onde a aplicação irá realizar as funções de solicitação de formulário, validação e atualização dos dados, sendo estes executados à parte, localmente em diferentes servidores.

A figura 4 descreve o funcionamento da plataforma, que será melhor detalhado na tabela 2:

Figura 4 - Estrutura de funcionamento da plataforma



Fonte: Elaborado pelo autor (2016).

A tabela 2 especifica os quatro núcleos distintos, nos quais a aplicação pode ser apresentada em torno das funcionalidades e dos usuários da plataforma proposta.

Tabela 2 - Detalhamento da estrutura de funcionamento da plataforma

Núcleo da aplicação	Camada responsável pela lógica e estrutura do sistema, e pelo gerenciamento do metamodelo.
Gerenciamento de metadados	Camada que envolve o gerenciamento das estruturas de dados (definição das estruturas) e das permissões de acesso.
Gerenciamento de conteúdo	Camada que envolve o gerenciamento de conteúdo (inserção, atualização, exclusão) de conteúdo dos metamodelos.
Requisição de conteúdo	Camada que permite a visualização do conteúdo dos metamodelos, por intermédio de diferentes interfaces, como páginas WEB e *Web Services*.

Fonte: Elaborado pelo autor (2016).

5.2 Escopo

Pressman (1995) afirma que a compreensão de forma clara e completa dos requisitos de um *software* é fundamental para o desenvolvimento deste com sucesso. Não importando a qualidade do projeto proposto, a incorreta análise do *software* irá gerar um resultado insatisfatório do usuário.

Sommerville (2010) separa os requisitos de software em dois tipos distintos:

- Requisitos Funcionais: especificações a que o *software* deve atender, bem como determinados *inputs* do sistema a que deve reagir; entendem-se também como requisitos funcionais as especificações que o sistema não deve realizar.

- Requisitos Não Funcionais: restrições que o software possui sobre determinadas funções ou serviços. Na maioria dos casos, aplicam-se a um contexto global do software.

Buscando, uma melhor compreensão das funcionalidades da plataforma proposta, a seguir, será apresentada a relação de requisitos funcionais e não funcionais que o sistema deve atender.

5.2.1 Requisitos funcionais

Tabela 3 - Descrição do Requisito Funcional 1

Requisito	RF001
Nome	Permitir estabelecer e gerenciar os usuários do sistema
Descrição	
O sistema deverá permitir o estabelecimento e gerenciamento de usuários que utilizaram o sistema.	

Fonte: Elaborado pelo autor (2016).

Tabela 4 - Descrição do Requisito Funcional 2

Requisito	RF002
Nome	Permitir estabelecer e gerenciar os tipos de dados
Descrição	
O sistema deve permitir que sejam definidos os tipos de dados a serem utilizados, bem como regras e suas restrições. Também devem ser possíveis o seu gerenciamento e atualizações através de um painel administrativo.	

Fonte: Elaborado pelo autor (2016).

Tabela 5 - Descrição do Requisito Funcional 3

Requisito	RF003
Nome	Permitir estabelecer e gerenciar os tipos de estruturas
Descrição	
O sistema deve permitir que sejam definidos os tipos de estruturas a serem utilizados, bem como regras e suas restrições. Também devem ser possíveis o seu gerenciamento e atualizações através de um painel administrativo.	

Fonte: Elaborado pelo autor (2016).

Tabela 6 - Descrição do Requisito Funcional 4

Requisito	RF004
Nome	Permitir o gerenciamento dos conteúdos dos metamodelos
Descrição	
O sistema deve permitir que sejam cadastrados os dados em suas respectivas estruturas, bem como o gerenciamento de CRUD (que deve ser realizado por meio de um painel administrativo).	

Fonte: Elaborado pelo autor (2016).

Tabela 7 - Descrição do Requisito Funcional 5

Requisito	RF005
Nome	Fornecer dados para exibição
Descrição	
O sistema deve fornecer os dados de suas estruturas para posterior visualização por meio de *Web Services*.	

Fonte: Elaborado pelo autor (2016).

5.2.2 Requisitos não funcionais

Tabela 8 - Descrição do Requisito Não Funcional 1

Requisito	RNF001
Nome	Utilizar arquitetura MVC
Descrição	
A ferramenta deve utilizar a arquitetura MVC.	

Fonte: Elaborado pelo autor (2016).

Tabela 9 - Descrição do Requisito Não Funcional 2

Requisito	RNF002
Nome	Utilizar plataforma WEB
Descrição	
O sistema deve ser desenvolvido em uma plataforma WEB.	

Fonte: Elaborado pelo autor (2016).

Tabela 10 - Descrição do Requisito Não Funcional 3

Requisito	RNF003
Nome	Utilizar linguagem de programação PHP
Descrição	
A ferramenta deve utilizar a linguagem de programação PHP.	

Fonte: Elaborado pelo autor (2016).

Tabela 11 - Descrição do Requisito Não Funcional 4

Requisito	RNF004
Nome	Utilizar banco de dados PostgreSQL
Descrição	
Utilizar o banco de dados PostgreSQL para manipular os dados do sistema.	

Fonte: Elaborado pelo autor (2016).

Tabela 12 - Descrição do Requisito Não Funcional 5

Requisito	RNF005
Nome	Utilizar servidor WEB Apache
Descrição	
A ferramenta deve utilizar o servidor WEB Apache.	

Fonte: Elaborado pelo autor (2016).

Tabela 13 - Descrição do Requisito Não Funcional 6

Requisito	RNF006
Nome	Utilizar o framework CodeIgniter
Descrição	
A ferramenta deve ser desenvolvida utilizando-se do framework PHP CodeIgniter.	

Fonte: Elaborado pelo autor (2016).

Tabela 14 - Descrição do Requisito Não Funcional 7

Requisito	RNF007
Nome	Utilizar a biblioteca JavaScript jQuery
Descrição	
A ferramenta utilizará a biblioteca JavScript jQuery na construção de suas interfaces.	

Fonte: Elaborado pelo autor (2016).

Tabela 15 - Descrição do Requisito Não Funcional 8

Requisito	RNF008
Nome	Utilizar *Web Services* para comunicação
Descrição	
A ferramenta irá disponibilizar *Web Services* para se comunicar com aplicações de terceiros.	

Fonte: Elaborado pelo autor (2016).

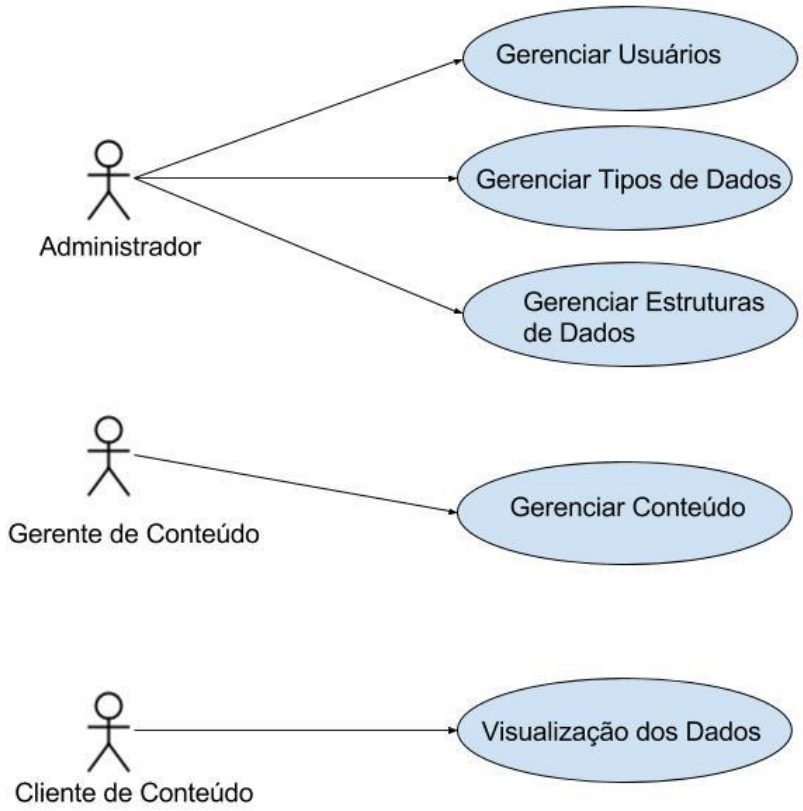
5.3 Casos de Uso

Knewitz (2011) define caso de uso como uma maneira de interpretar e compreender os conceitos de ator e cenário, dando um tratamento mais formal à representação dessas interações. Booch, Rumbaugh e Jacobson (2005) descrevem como agentes o ser humano ou sistemas automatizados, e como ator, um conjunto de papéis os quais os usuários possam assumir em uma interação.

5.3.1 Diagrama de Caso de Uso

Segundo Kewitz (2011, p. 51), “o diagrama de casos de uso é um diagrama que mostra um conjunto de casos de uso, atores e seus relacionamentos”; já Fowler (2005) descreve este tipo de diagrama como uma valiosa ferramenta para auxiliar no entendimento de requisitos funcionais de um sistema. Pretendendo um melhor entendimento dos requisitos da aplicação, foi elaborado um diagrama de caso de uso, o qual é apresentado na figura 5.

Figura 5 - Diagrama de caso de uso



Fonte: Elaborado pelo autor (2016).

5.3.2 Especificação dos Casos de Uso

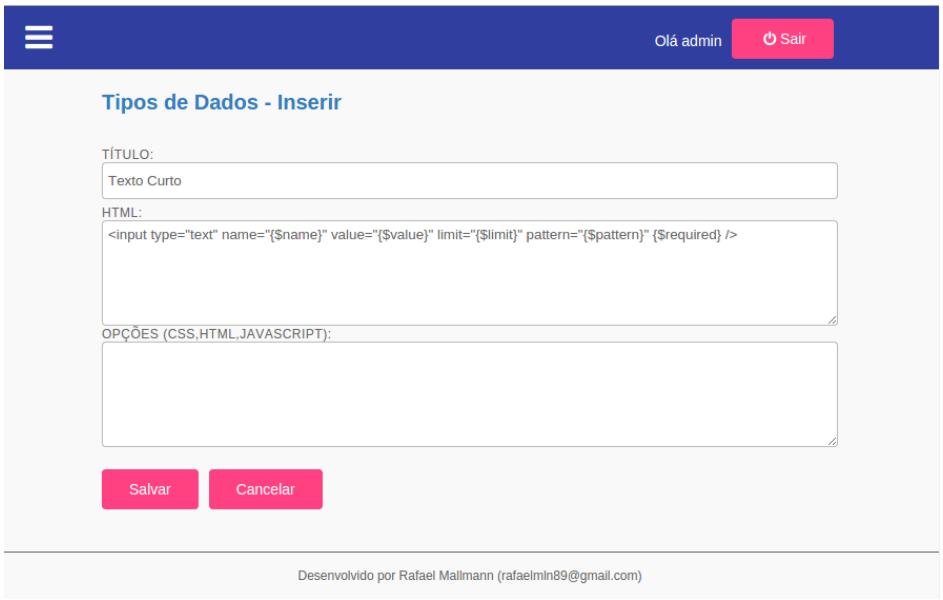
A seguir serão descritos os casos de uso da aplicação, que representam as interações entre os casos de uso e os atores envolvidos, além de ser apresentado um esboço das interfaces abordadas em cada ocasião.

Tabela 16 - Descrição do caso de uso Gerenciar tipos de dados

Caso de Uso	UC01
Nome	Gerenciar tipos de dados
Ator(es)	Administrador
Descrição	
A aplicação deve possibilitar ao administrador gerenciar os diferentes tipos de dados que possam vir a ser utilizados no sistema.	

Fonte: Elaborado pelo autor (2016).

Figura 6 - Interface do caso de uso Gerenciar tipos de dados



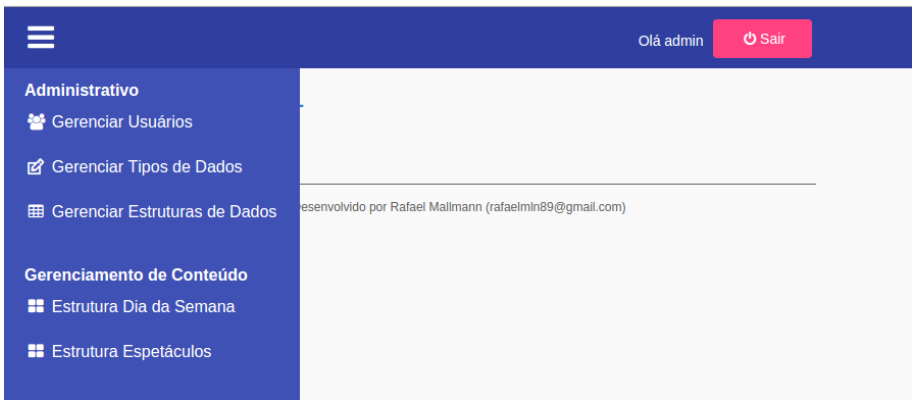
Fonte: Elaborado pelo autor (2016).

Tabela 17 - Descrição do caso de uso Gerenciar permissões

Caso de Uso	UC02
Nome	Gerenciar permissões
Ator(es)	Administrador
Descrição	
O administrador deve gerenciar as permissões a que cada gerente de conteúdo terá acesso na plataforma.	

Fonte: Elaborado pelo autor (2016).

Figura 7 - Interface do menu com opções recorrentes as suas permissões



Fonte: Elaborado pelo autor (2016).

Tabela 18 - Descrição do caso de uso Gerenciar estruturas de dados

Caso de Uso	UC03
Nome	Gerenciar estruturas de dados
Ator(es)	Administrador
Descrição	
A plataforma deve possibilitar o gerenciamento de diversas estruturas de dados, de acordo com as necessidades de cada caso.	

Fonte: Elaborado pelo autor (2016).

Figura 8 - Interface do caso de uso Gerenciar estruturas de dados

Fonte: Elaborado pelo autor (2016).

Tabela 19 - Descrição do caso de uso Gerenciar dados

Caso de Uso	UC04
Nome	Gerenciar dados
Ator(es)	Gerente de Conteúdo
Descrição	
O gerente de conteúdo deve realizar o cadastro e gerenciamento dos dados nas diversas estruturas de dados.	

Fonte: Elaborado pelo autor (2016).

Figura 9 - Interface do caso de uso Gerenciar dados

Fonte: Elaborado pelo autor (2016).

Figura 10 - Interface do caso de uso Gerenciar dados

Fonte: Elaborado pelo autor (2016).

Tabela 20 - Descrição do caso de uso Visualização de conteúdo

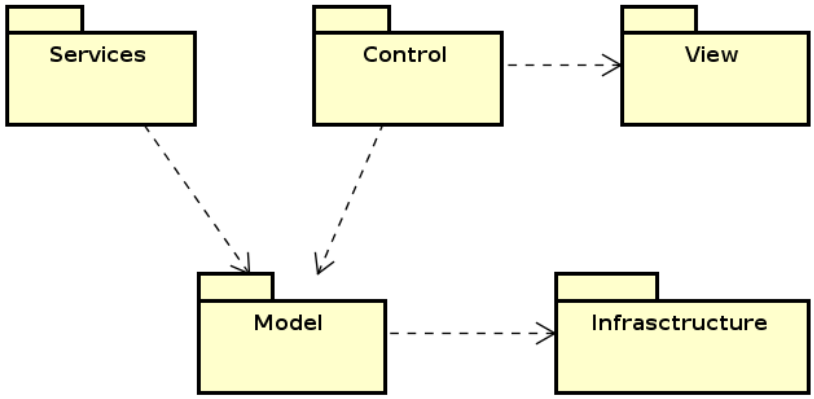
Caso de Uso	UC05
Nome	Visualização de conteúdo
Ator(es)	Cliente de Conteúdo
Descrição	
O cliente de conteúdo terá acesso à visualização dos dados por intermédio de uma interface *website* de exibição dos dados.	

Fonte: Elaborado pelo autor (2016).

5.4 Arquitetura da aplicação

Buscando visualizar melhor a organização da estrutura da aplicação, esta pode ser exemplificada através de módulos, que são ilustrados na figura 11 e melhor detalhados nas tabelas 23 a 27:

Figura 11 - Visão geral da arquitetura



Fonte: Elaborado pelo autor (2016).

Tabela 21 – Especificação do pacote Services

Pacote	Services.
Descrição	O pacote Services contempla toda a camada de comunicação com o sistema por intermédio de *Web Services*.
Conteúdo	WebServiceContent: Classe de consultas de registros; WebServiceAdmin: Classe de gerenciamento de conteúdo por requisições;

Fonte: Elaborado pelo autor (2016).

Tabela 22 - Especificação do pacote Control

Pacote	Control.
Descrição	Contempla a camada de controle das requisições WEB. Neste pacote é realizado o controle de chamada aos demais pacotes model e view, que acabam por processar as requisições de forma correta.

Fonte: Elaborado pelo autor (2016).

Tabela 23 - Especificação do pacote Model

Pacote	Model.
Descrição	Contém as classes que representam as entidades do modelo e implementam a lógica de negócios e comunicação com o BD.
Conteúdo	Formulario_model: apresenta as funções de apoio à tela de *login*; Estruturas_model: apresenta as funções para manipular as estruturas; Tipos_model: apresenta as funções para manipular os tipos; Usuarios_model: apresenta as funções para manipular os usuários; Registros_model: representa um registro de uma estrutura de um metamodelo, e possui métodos para manipular este registro;

Fonte: Elaborado pelo autor (2016).

Tabela 24 - Especificação do pacote View

Pacote	View.
Descrição	Contém as interfaces de visualização e apresentação dos dados ao usuário; nestas são apresentadas as interfaces HTML e JavaScript das páginas.
Conteúdo	footer: contém o HTML da rodapé do sistema; header: contém o HTML da cabeçalho do sistema; listar: listagem padrão dos dados cadastrados no sistema; login: contém o HTML da página de *login*; menu: contém o HTML de opções do menu, de acordo com as permissões; estruturas_editar: contém o HTML da página de edição de estruturas; registros_editar: contém o HTML da página de edição de registros; tipos: contém o listagem de tipos cadastrados; tipos_editar: contém o HTML da página de edição de tipos; usuarios_editar: contém o HTML da página de edição de usuários; welcome_message: contém o HTML da página de abertura do sistema;

Fonte: Elaborado pelo autor (2016).

Tabela 25 - Especificação do pacote Infrasctructure

Pacote	Infrasctructure.
Descrição	Contém pacotes que oferecem recursos básicos de infraestrutura. Contém bibliotecas PHP, com intuito de obter um ganho de produtividade, responsáveis por comunicação com recursos externos (banco de dados e *Web Services*).
Conteúdo	MY_Banco: Desenvolvida para realizar a comunicação com a estrutura de metamodelo, onde esta é tratada, sendo utilizada como um banco de dados relacional. Xmlrpc: Biblioteca de comunicação xml para cliente e servidor.

Fonte: Elaborado pelo autor (2016).

Segundo Dall'Oglio (2012 p. 13), o "*MVC é um padrão de projeto de software que consiste na separação da aplicação em camadas lógicas*", sendo que este padrão de desenvolvimento em camadas, pretende conseguir um melhor gerenciamento de sistemas complexos. Conforme a W3School (2015), as camadas do MVC consistem em:

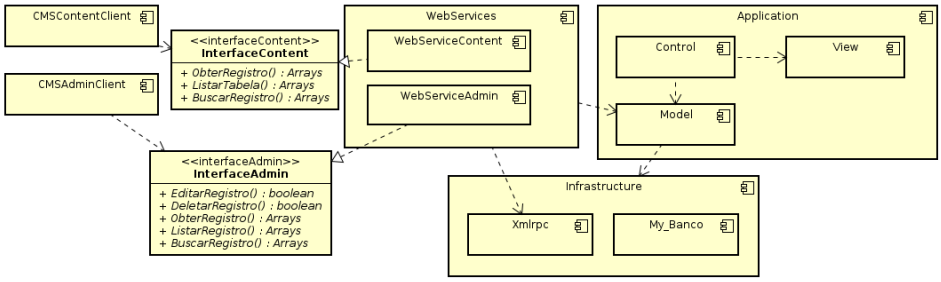
- Model: camada da aplicação, que representa o controle lógico do negócio, realizando as transações de busca e armazenamento junto ao banco de dados;
- View: camada da aplicação, onde são realizados o controle e a exibição dos dados para com os usuários;
- Controller: camada que é responsável pelo controle e gerência entre os dados de entrada e saída, fazendo esta a relação com as chamadas dos dados na camada model e demonstração nas camadas de views.

A aplicação em questão será desenvolvida seguindo os padrões desta arquitetura, buscando assim um desenvolvimento que utilize as melhores práticas de mercado.

5.5 Especificação dos Serviços

A arquitetura dos serviços descreve os componentes que atuam com a aplicação; nesta podemos observar os recursos disponibilizados na plataforma para utilização de outras ferramentas. A figura 12 é o esboço da arquitetura da aplicação, que será melhor apresentada nas tabelas 26 a 31:

Figura 12 - Especificação dos serviços



Fonte: Elaborado pelo autor (2016).

Tabela 26 - Especificação do Serviço CMSContentClient

Componente	CMSContentClient.
Descrição	Contempla as requisições de leitura ao sistema, onde apenas a permissão de leitura é necessária. Representa o solicitante do *Web Service*, aquele que irá solicitar o conteúdo para posterior apresentação dos dados.

Fonte: Elaborado pelo autor (2016).

Tabela 27 - Especificação do Serviço CMSAdminClient

Componente	CMSAdminClient.
Descrição	Contempla todas as requisições de gerenciamento de conteúdo (CRUD), exige as permissões de leitura, escrita e exclusão no sistema. Representa o gerente de conteúdo.

Fonte: Elaborado pelo autor (2016).

Tabela 28 - Especificação do Serviço WebServiceContent

Componente	WebServiceContent.
Descrição	Processamento das requisições *Web Services* ao sistema, contempla as requisições de leitura realizadas pelo pacote CMSContentClient.
Métodos disponíveis	obter(): retorna o registro solicitado; listar(): retorna todos os registros da estrutura em questão; buscar(): retorna os registros que forem compatíveis com os filtros indicados;

Fonte: Elaborado pelo autor (2016).

Tabela 29 - Especificação do Serviço WebServiceAdmin

Componente	WebServiceAdmin.
Descrição	Processamento das requisições *Web Services* ao sistema, contempla além de requisições de leitura, requisições de escrita e exclusão. Estas são realizadas pelo pacote CMSAdminClient.
Métodos disponíveis	editar(): atualizada um registro com os valores informados; inserir(): insere um registro; excluir(): exclui o registro indicado; obter(): retorna o registro solicitado; listar(): retorna todos os registros da estrutura em questão; buscar(): retorna os registros que forem compatíveis com os filtros indicados;

Fonte: Elaborado pelo autor (2016).

Tabela 30 - Especificação do Serviço Aplication

Componente	Aplication.
Descrição	Formado pelos pacotes controls, model e views, que compõem a tradicional arquitetura MVC.

Fonte: Elaborado pelo autor (2016).

Tabela 31 - Especificação do Serviço Infrastructure

Componente	Infrastructure.
Descrição	O serviço contempla as bibliotecas do sistema, que consistem na classe MY_Banco, para comunicação com o metamodelo, e a classe Xmlrpc para comunicação com arquivos XML.

Fonte: Elaborado pelo autor (2016).

5.6 Utilização das Classes

Nesta seção, serão apresentados exemplos de utilização do modelo criado; demostrando assim seu uso na estrutura do sistema e o seu aproveitamento por clientes do sistema.

A Figura 13 demonstra como é realizada a criação de uma nova estrutura no sistema, configurando o título na variável “$estrutura” e as colunas que a estrutura irá contemplar no array “$campos”.

Figura 13 - Classe para inserir uma nova estrutura

```php
<?php
class InserirEstrutura extends CI_Controller
{
    //Carrega classe que representa a estrutura
    $this->load->model('Estruturas_model');

    $estrutura = array();
    $estrutura["titulo"] = "Dia da Semana";

    $campos = array();
    $campos[0] = array(
        ["titulo"] => "Descrição",
        ["limite"] => NULL,
        ["mascara"] => ".*",
        ["aceita_nulo"] => "f",
        ["tipo_id"] => 27 );

    //insere registro na tabela: estrutura
    $estrutura_id = $this->Estruturas_model->salvar($estrutura);
    //insere referência de campos a estrutura
    $this->Estruturas_model->salvar_campos($estrutura_id, $campos);
}
?>
```

Fonte: Elaborado pelo autor (2016).

A Figura 14 descreve a utilização da aplicação para listagem de registros salvos em determinada estrutura.

Figura 14 - Classe para inserir um registro

```php
<?php
class InserirRegistro extends CI_Controller
{
    //Carrega classe que representa os registros das estruturas
    $this->load->model('Registros_model');
    //Carrega classe que representa a estrutura
    $this->load->model('Estruturas_model');

    //obtem dados da estrutura em questão
    $tabela_id = $this->Estruturas_model->obter_id_tabela('diadasemana');
    $campos = $this->Estruturas_model->listar_campos_tabela($tabela_id);

    //Registro a ser inserido na estrutura
    $registro = array( ["descrio"] => "Domingo" );
    $registro_id = $this->Registros_model-
>salvar_registro($registro, $tabela_id, montar_assoc($campos, 'descricao', 'id' ));
}
?>
```

Fonte: Elaborado pelo autor (2016).

A Figura 15 descreve a utilização da aplicação para listagem de registros salvos em determinada estrutura, podendo os retornos dos registros ao final deste código ser acessados através da variável “$dados”.

Figura 15 - Classe para listagem de registros

```php
<?php
class ListarRegistros extends CI_Controller
{
    //Carrega classe que representa a estrutura
    $this->load->model('Estruturas_model');

    $tabela_id = $this->Estruturas_model->obter_id_tabela('diadasemana');
    $dados = $this->Estruturas_model->listar_tabela( $tabela_id );
}
?>
```

Fonte: Elaborado pelo autor (2016).

5.7 Modelo de Classes

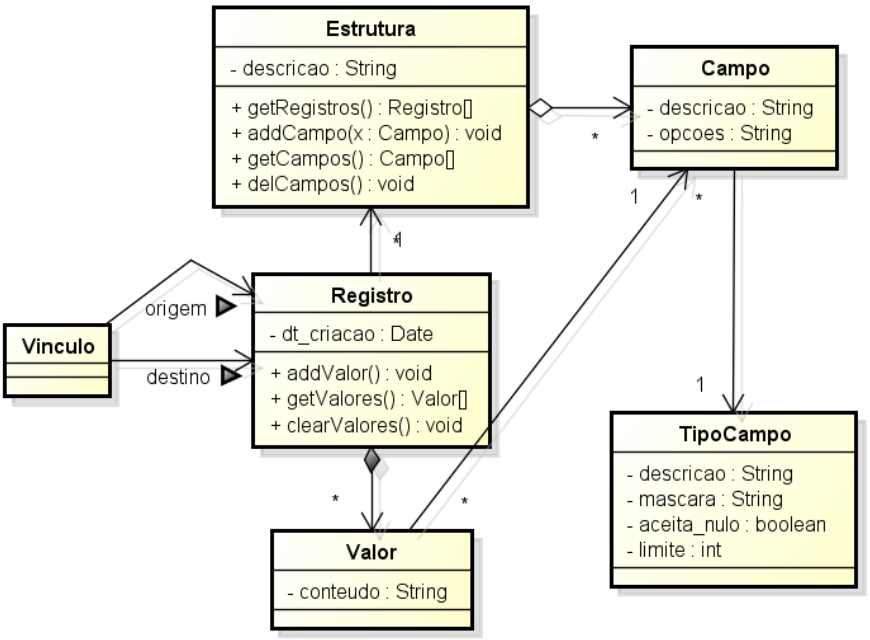
O modelo de classes procura representar a estrutura de classes, métodos e atributos da UML Knewitz (2011, p. 70) declara que:

> No paradigma da orientação a objetos, um sistema de software é modelado como uma coleção de objetos interconectados que colaboram entre si executando tarefas específicas ou solicitando a execução de tarefas por outros objetos, de forma que as funcionalidades do sistema sejam cumpridas. Classe é a abstração de objetos descritos pelos mesmos dados e que possuem o mesmo comportamento.

5.7.1 Diagrama de Classes

A figura 16 apresenta o diagrama de classes, e as estruturas e comportamentos de cada classe em questão; sendo que a estrutura é tida pelos seus atributos, e as operações é que definem o comportamento (KNEWITZ, 2011).

Figura 16 - Diagrama de classes



Fonte: Elaborado pelo autor (2016).

5.7.2 Detalhamento das Classes

As tabelas abaixo descrevem as relações de classes, das quais o sistema é composto; apresenta também um detalhamento de atributos e operações.

Tabela 32 - Descrição da classe Estrutura

Classe	Estrutura
Descrição	
Classe utilizada para representar uma entidade no metamodelo.	
Atributos	
Nome	Descrição
Descrição	Título da entidade
Operações	
Nome	Descrição

getRegistros()	Retorna a lista de registros da entidade em questão
addCampo()	Insere um campo na entidade
getCampos()	Retorna os campos da entidade em questão
delCampos()	Exclui campo(s) da entidade

Fonte: Elaborado pelo autor (2016).

Tabela 33 - Descrição da classe Campo

Classe	Campo
Descrição	
Classe utilizada para representar os variados campos.	
Atributos	
Nome	Descrição
Descrição	Título do campo
Opções	Representação das opções que este campo possui

Fonte: Elaborado pelo autor (2016).

Tabela 34 - Descrição da classe TipoCampo

Classe	TipoCampo
Descrição	
Classe utilizada para representar restrições de tipos e regras dos campos.	
Atributos	
Nome	Descrição
Descrição	Identificação do tipo do campo
Mascara	Expressão regular que representa a validação do campo
aceita_nulo	Identificação de campo *not null*
Limite	Tamanho máximo de caracteres que o campo aceita

Fonte: Elaborado pelo autor (2016).

35Tabela 36 - Descrição da classe Registro

Classe	Registro
Descrição	
Classe utilizada para representar o índice aos registros das entidades.	
Atributos	
Nome	Descrição
dt_criacao	Data na qual o registro foi atualizado.
Operações	
Nome	Descrição
addValor()	Insere um registro na estrutura
getValores()	Retorna os valores do referido registro
clearValores()	Exclui os valores do referido registro

Fonte: Elaborado pelo autor (2016).

37Tabela 38 - Descrição da classe Vinculo

Classe	Vinculo
Descrição	
Classe utilizada para representar referência estrangeira entre registros.	

Fonte: Elaborado pelo autor (2016).

39Tabela 40 - Descrição da classe Valor

Classe	Valor
Descrição	
Classe utilizada para representar os valores destinados a cada registro.	
Atributos	
Nome	Descrição
Conteúdo	Representa o valor do campo

Fonte: Elaborado pelo autor (2016).

Os métodos básicos de persistência (incluir, excluir, …) não serão especificados, pois os mesmos já estarão implementados pela arquitetura selecionada para este trabalho, e, conforme Fowler (2005), é dispensável a especificação dos métodos de persistência, implementados na arquitetura.

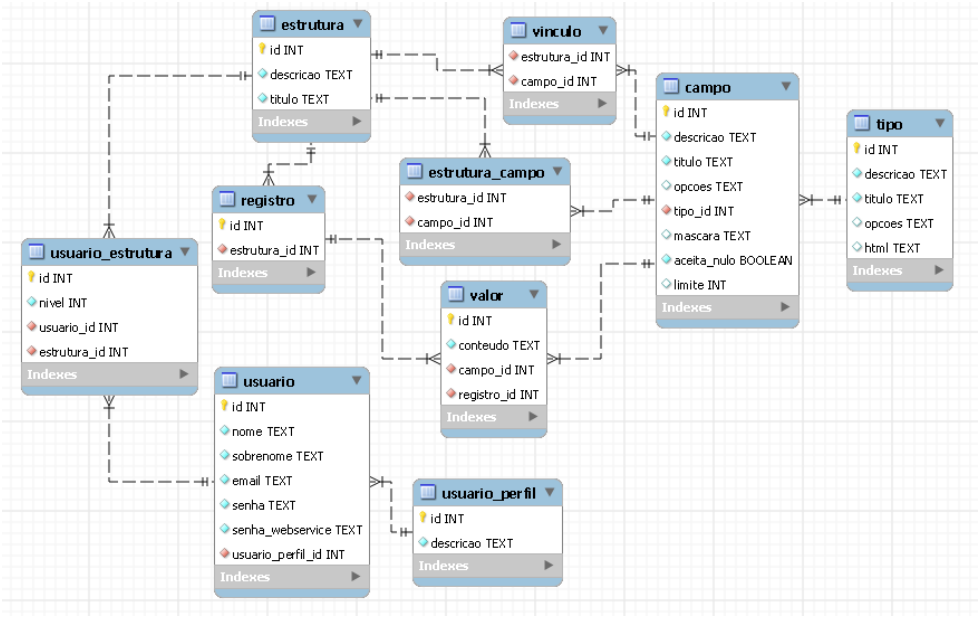
5.8 Modelo ER

Pressman (1995) descreve que a modelagem de dados é utilizada em aplicações de banco de dados. Segundo Pressman (1995, p. 342), "Ela proporciona ao analista e ao projetista de banco de dados o necessário esclarecimento sobre os dados e as relações que regem esses dados".

5.8.1 Modelo Relacional

Buscando um melhor entendimento da arquitetura utilizada, foi elaborado um diagrama de modelo relacional apresentado na figura 17, pois, segundo Pressman (2009, p. 349), "usando o diagrama entidade-relacionamento como notação fundamental, a modelagem de dados concentra-se na definição de objetos de dados (objetos que não encapsulam processamento) e na maneira pela qual eles se relacionam entre si".

Figura 17 - Diagrama ER da aplicação



Fonte: Elaborado pelo autor (2016).

5.8.2 Detalhamento das Tabelas

A tabela 38, é composta pelas relações das tabelas que integram o diagrama ER da aplicação, bem como suas descrições específicas:

Tabela 41 - Detalhamento das tabelas

Nome da Tabela	Descrição
Estrutura	Tabela que busca implementar a identificação das entidades no metamodelo.
estrutura_campo	Tabela responsável por representar a relação entre a entidade/tabela e os campos pertencentes a ela.
Campo	Tabela que identifica os variados campos que compõem as tabelas.
Tipo	Tabela que identifica as restrições de tipos e regras dos campos.
Valor	Tabela que recebe os valores destinados a cada registro, sendo diretamente relacionada com a tabela campo.
Registro	Tabela que fornece o índice aos registros das tabelas.
Vinculo	Tabela que exerce o papel de relação de chave estrangeira entre os diferentes registros.
Usuário	Tabela que representa os usuários com acesso permitido a plataforma.
usuario_perfil	Tabela com a relação dos diferentes perfis de acesso dos usuários.
usuario_estrutura	Tabela que exerce o papel de relação entre as diferentes estruturas e usuários, bem como a permissão em cada entidade.

Fonte: Elaborado pelo autor (2016).

5.9 Utilização dos Serviços

Buscando uma maior diversidade de aplicação do sistema de gerenciamento de conteúdo, desenvolveu-se uma interface de comunicação via requisições *Web Services*, e por meio destas, torna-se possível uma utilização e gerenciamento de dados com segurança, mesmo sem acesso à interface da aplicação.

A estrutura do *Web Service* criado, exige a estrutura de consulta em que os dois primeiros parâmetros sejam a identificação do usuário solicitante (usuário e senha); como terceiro parâmetro obrigatório é informada a função, que determina o tipo de solicitação realizada, a relação de funções disponibilizadas pode ser observada nas tabelas 28 e 29; o quarto parâmetro é a estrutura onde a ação é realizada, sendo este parâmetro opcional, dependendo da função que se deseja executar. Estes dados podem ser passados através de parâmetros da requisição GET ou POST.

O sistema, uma vez que utiliza o Framework CodeIgniter, disponibiliza a biblioteca de comunicação XML entre servidor e cliente XML-RPC, usada para desenvolvimento dos serviços de requisições *Web Services*.

As imagens 18 e 19 ilustram a utilização da classe Xmlrpc, demonstrando o funcionamento de um modelo cliente servidor, para o processamento de uma requisição ao sistema.

Figura 18 - Utilização da classe xmlrpc como cliente

```php
<?php
class Cms_content_client extends CI_Controller
{
    public function index()
    {
        //carrega bibliotecas
        $this->load->library('xmlrpc');
        $this->load->helper('url');
        //define url e porta do server
        $server_url = site_url('Web_service_content');
        $this->xmlrpc->server($server_url, 80);
        //definicao da funcao
        $this->xmlrpc->method('obter_registro');
        //usuario, senha do webservice, tabela, parametro(s)
        $request = array('admin', 'e10adc3949ba59abbe56e057f20f883e', 'login', 17);

        //chamada ao server
        $this->xmlrpc->request($request);
        if ( ! $this->xmlrpc->send_request())
        {
            echo $this->xmlrpc->display_error();
        }
        else
        {
            $retorno = $this->xmlrpc->display_response();
            echo $retorno[0];
        }
    }
}
?>
```

Fonte: Elaborado pelo autor (2016).

Figura 19 - Utilização da classe xmlrpc como servidor

```php
<?php
class Web_service_content extends CI_Controller
{
    function __construct()
    {
        parent::__construct();
        //carrega bibliotecas
        $this->load->library('xmlrpc');
        $this->load->library('xmlrpcs');
    }

    public function index()
    {
        //inicializacao do servico
        $config['functions']
['obter_registro'] = array('function' => 'Web_service_content.obter_registro');
        $this->xmlrpcs->initialize($config);
        $this->xmlrpcs->serve();
    }

    function obter_registro($request)
    {
        //leitura dos parametros
        $parameters = $request->output_parameters();
        //obtem os registros solicitados no banco de dados
        $this->load->model('Registros_model');
        $dados = $this->Registros_model-
>obter_registro( $parameters[2], $parameters[3] );
        //retorno da solicitacao
        $response = array( array(json_encode($dados)), 'struct');
        return $this->xmlrpc->send_response($response);
    }
}
?>
```

Fonte: Elaborado pelo autor (2016).

Procurando uma melhor ilustração do funcionamento e utilização das classes do sistema, a figura 20 representa uma solicitação de consulta *Web Service* via código PHP, em que se observa a definição dos dados da consulta, onde os parâmetros de acesso de usuário e senha são definidos, bem como a função a ser solicitada, a tabela e os parâmetros para essa consulta. Em seguida, a url de requisição é elaborada juntamente com os parâmetros em formato GET. Por fim é realizada a solicitação via função file_get_contents() do PHP, onde a requisição é feita de fato.

Figura 20 - Exemplo de requisição Web Service via PHP

```php
<?php
//parametros da requisicao
$function = 'obter_registro';
$user = 'admin';
$pass = 'e10adc3949ba59abbe56e057f20f883e';
$table = 'login';
$param = '17';

$params_get = 'function='.$function.'&user='.$user.'&pass='.$pass.'&table='.$table
.'&param='.$param;
$url_webservice = 'http://localhost/tcc/aplicacao/index.php/cms_content_client';
$url_webservice = $url_webservice.'?'.$params_get;
//requisicao
$espetaculos = json_decode( file_get_contents( $url_webservice ) );
?>
```

Fonte: Elaborado pelo autor (2016).

5.10 Tecnologias Utilizadas

Como linguagem de programação para a aplicação, foi utilizado o PHP, tradicional linguagem de *scripting*, amplamente utilizada para o desenvolvimento WEB, podendo esta ser diretamente incorporada ao HTML (PHP, 2015). O autor deste trabalho possui conhecimento prévio desta linguagem, bem como suas características e funcionalidades, o que contribuiu para a escolha do PHP neste trabalho.

Pretendendo uma otimização de desenvolvimento dentro dos padrões MVC, foi usado o Framework PHP CodeIgniter, cuja estrutura de desenvolvimento atende à estrutura desejada, além de fornecer classes e bibliotecas que venham a favorecer o correto desenvolvimento nos padrões desejados (CODEIGNITER, 2015).

Como banco de dados, a aplicação fez uso do PostgreSQL, facilmente adaptável ao desenvolvimento WEB com uso de linguagem PHP, além de ser considerada uma solução robusta e confiável para o armazenamento de dados estruturais, cumprindo assim os requisitos do projeto proposto (MILANI, 2008).

5.11 Cenário de Testes

Para realizar uma avaliação prévia, buscou-se simular uma situação real, onde a plataforma proposta teria sua utilização testada e validada. Nesta etapa, foi utilizoda a plataforma em ambiente local, tendo sido empregados os recursos PHP, banco de dados PostgreSQL e Apache como servidor de aplicação; neste ambiente foi instalada a aplicação juntamente com sua base de dados, ficando assim acessível via *localhost*;

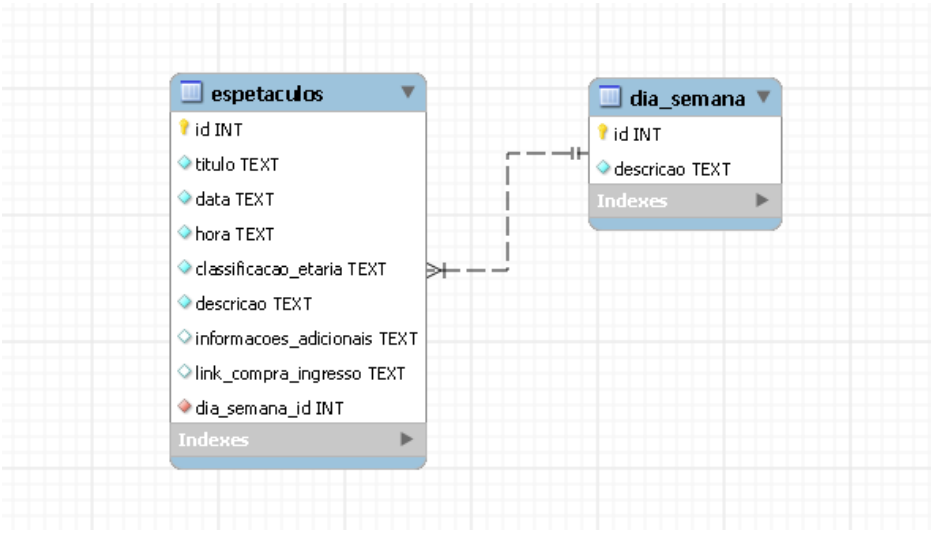
Após análise de *websites* relacionados a espaços culturais e teatros, identificou-se que todos estes apresentam uma relação das próximas atrações a serem realizadas; como estes eventos ocorrem geralmente em uma data prevista, essa relação deve ser atualizada periodicamente por funcionários do estabelecimento, constituindo um caso real de utilização de sistemas de gerenciamento de conteúdo.

5.11.1 Modelo do Cenário

Buscando cumprir os requisitos para uma boa apresentação das informações destes espetáculos em um *website*, torna-se necessário informar uma apresentação do espetáculo, os horários e a classificação etária. A estrutura em questão contempla duas tabelas, que são apresentadas na figura 21. Os registros contemplam os seguintes campos:

- Nome do espetáculo;
- Classificação etária;
- Data;
- Hora;
- Dia da semana em que ocorre;
- Descrição do espetáculo;
- Informações adicionais.

Figura 21 - Diagrama ER para cadastro de espetáculos



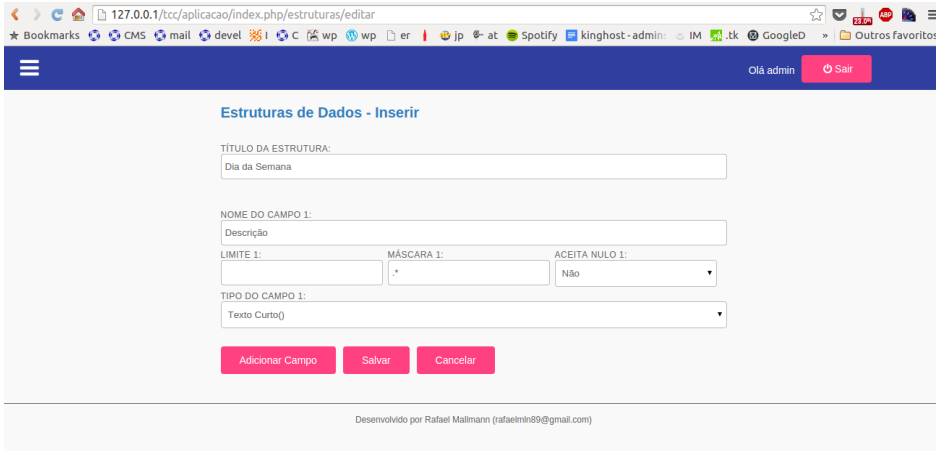
Fonte: Elaborado pelo autor (2016).

5.11.2 Definição da Estrutura

O sistema oferece ao usuário com permissões de administrador, a opção de “gerenciar estruturas”, onde selecionando a opção “inserir” pode realizar o cadastro de uma nova estrutura na plataforma.

A figura 22 apresenta a criação da estrutura “Dia da Semana” no sistema, a qual representa a tabela dia_semana do diagrama ER apresentado na figura 21. Nesta se pode observar a definição do título como sendo Dia da Semana, e o campo de texto curto, denominado Descrição. O campo id representado na figura 21 não foi cadastrado, pois o sistema já contempla um identificador único para todas as estruturas criadas.

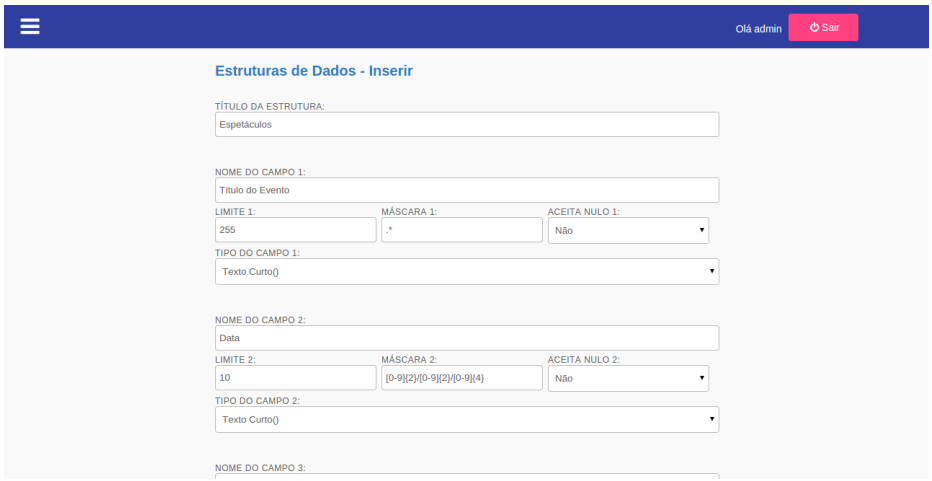
Figura 22 - Definição da estrutura “Dia da Semana”



Fonte: Elaborado pelo autor (2016).

A figura 23 apresenta um exemplo de cadastro da estrutura “Espetáculos”, representada na figura 21. Os campos cadastrados obedecem às regras impostas pelo diagrama ER, onde são definidas restrições de campo nulo, limite de caracteres, o tipo de registro, e as máscaras, ou regras de validação de cada campo.

Figura 23 -Definição da estrutura “Espetáculos”



Fonte: Elaborado pelo autor (2016).

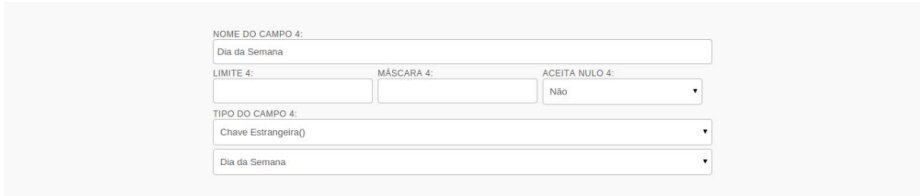
A figura 24 demonstra o cadastro do campo “Classificação etária”, onde o tipo de campo selecionado exige que as opções deste sejam adicionadas.

Figura 24 - Detalhe do cadastro do campo classificação etária

Fonte: Elaborado pelo autor (2016).

A figura 25 ilustra a forma de cadastro do campo “Dia da semana” na estrutura “Espetáculos”, exemplificando o cadastro do tipo chave estrangeira, onde é necessário selecionar a estrutura à qual o registro deva ser vinculado.

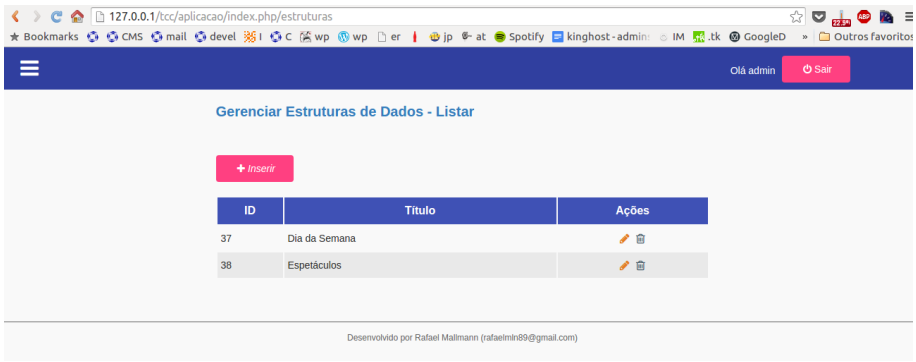
Figura 25 - Detalhe do cadastro do campo dia da semana



Fonte: Elaborado pelo autor (2016).

Após a realização do cadastro das duas estruturas, as relações das estruturas cadastradas podem ser observadas nos seus dados básicos na tela de listagem de estruturas. A figura 26 ilustra o funcionamento desta tela.

Figura 26 - Relação de estruturas cadastradas



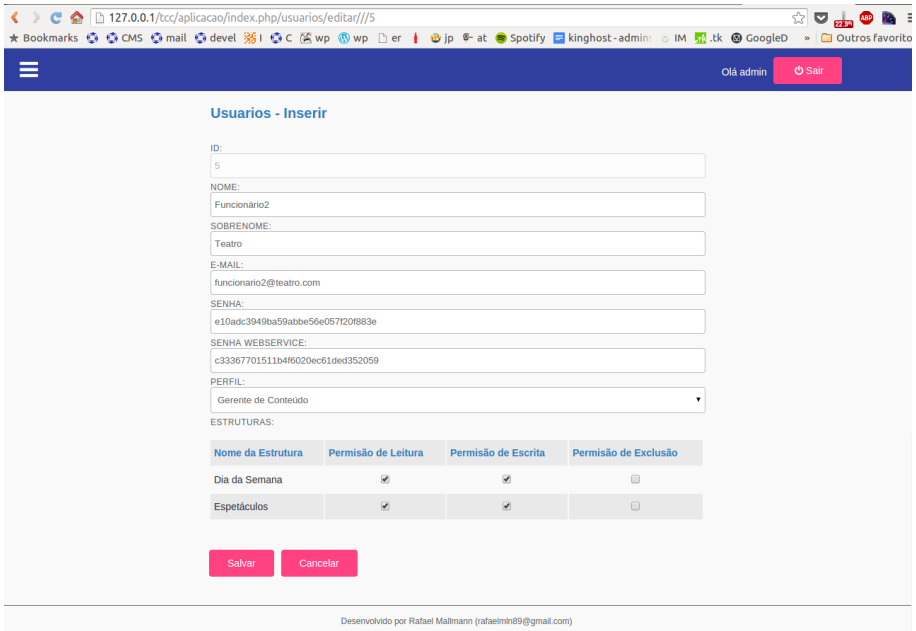
Fonte: Elaborado pelo autor (2016).

5.11.3 Gerenciamento de Conteúdo

Para a realização da tarefa de gerenciamento de conteúdo, sem fornecer acesso privilegiado ao sistema, foi realizado o cadastro de dois usuários fictícios, que representam funcionários do teatro em questão. Estes recebem o perfil de Gerenciadores de Conteúdo, com permissões de leitura e escrita de dados nas referidas estruturas.

A figura 27 ilustra o cadastro do usuário “Funcionário2”, onde são descritos seus dados pessoais, com a definição de senha de acesso ao sistema, a senha de requisição de serviço via *Web Service* e, por fim, o perfil e permissões específicas em cada estrutura do sistema.

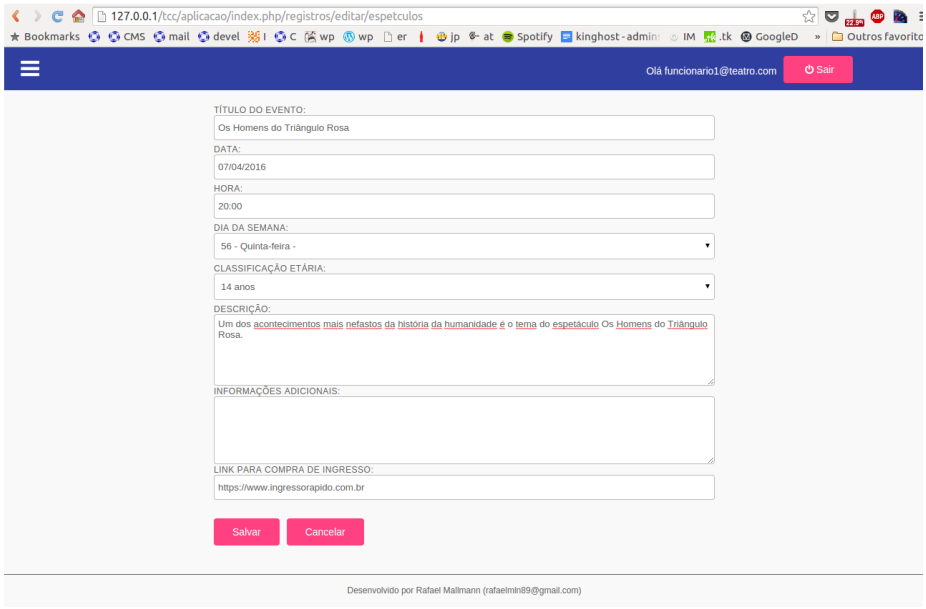
Figura 27 - Cadastro de usuário



Fonte: Elaborado pelo autor (2016).

O sistema apresenta um menu de opções dinâmico, de acordo com as permissões concedidas para cada usuário. Com o cadastro devidamente realizado das estruturas e dos usuários, o usuário “Funcionario2” pode realizar o *login* na plataforma e ter acesso ao cadastro de espetáculos, onde está disponível o gerenciamento desses registros. A figura 28 ilustra a utilização do sistema para cadastro de um espetáculo na plataforma.

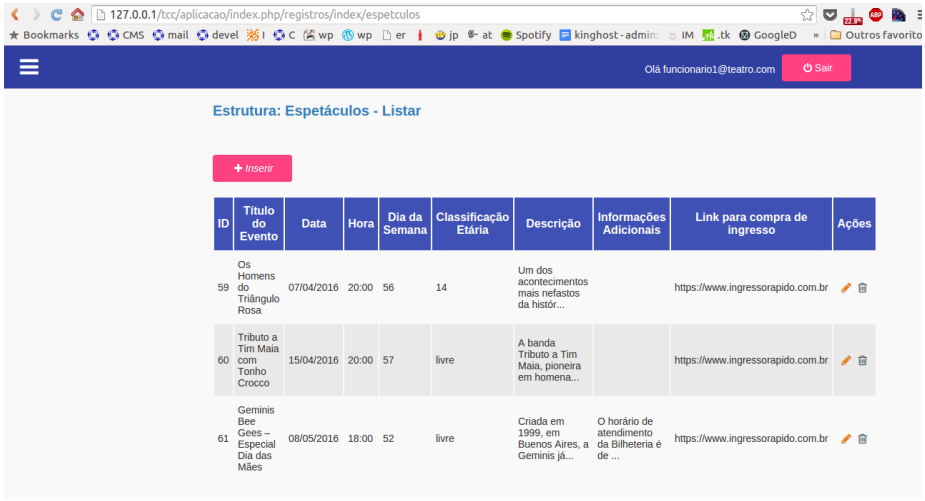
Figura 28 - Cadastro de espetáculo



Fonte: Elaborado pelo autor (2016).

A plataforma oferece uma listagem de cadastros de cada estrutura, com informações resumidas. A figura 29 mostra a relação de registros inserida na estrutura Espetáculos.

Figura 29 - Relação dos espetáculos cadastrados



Fonte: Elaborado pelo autor (2016).

5.11.4 Requisição de Conteúdo

Para a realização de consulta dos dados cadastrados no sistema, a plataforma disponibiliza *Web Services* de consultas e atualizações, podendo assim ser gerenciada sem o acesso direto à plataforma.

O exemplo de código apresentado na figura 30, ilustra um exemplo de solicitação *Web Service* para uma requisição de conteúdo, onde os parâmetros definidos solicitam a listagem dos registros na estrutura "espetculos". A requisição é realizada através da função PHP file_get_contents().

Figura 30 - Exemplo de código de requisição Web Service

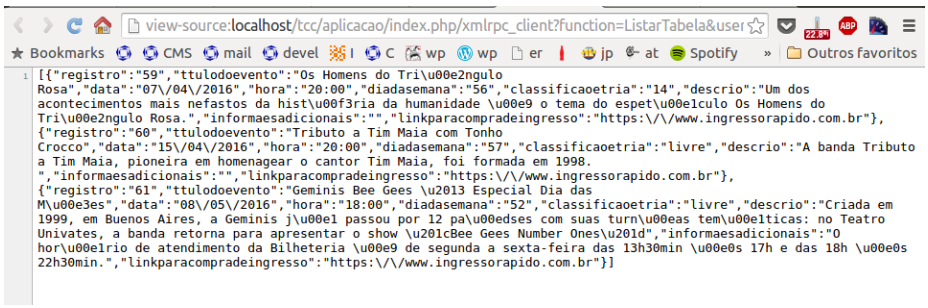
```php
<?php
//parametros da requisicao
$function = 'listar_tabela';
$user = 'admin';
$pass = 'e10adc3949ba59abbe56e057f20f883e';
$table = 'espetculos';

$params_get = 'function='.$function.'&user='.$user.'&pass='.$pass.'&table='.$table;
$url_webservice = 'http://localhost/tcc/aplicacao/index.php/cms_content_client';
$url_webservice = $url_webservice.'?'.$params_get;
//requisicao
$espetaculos = json_decode( file_get_contents( $url_webservice ) );
?>
```

Fonte: Elaborado pelo autor (2016).

A figura 31 ilustra um exemplo de retorno de chamada *Web Service* apresentado na figura 30, a qual realiza a solicitação dos registros cadastrados na estrutura Espetáculos. Esses retornos são apresentados em formato JSON.

Figura 31 - Retorno de requisição em JSON

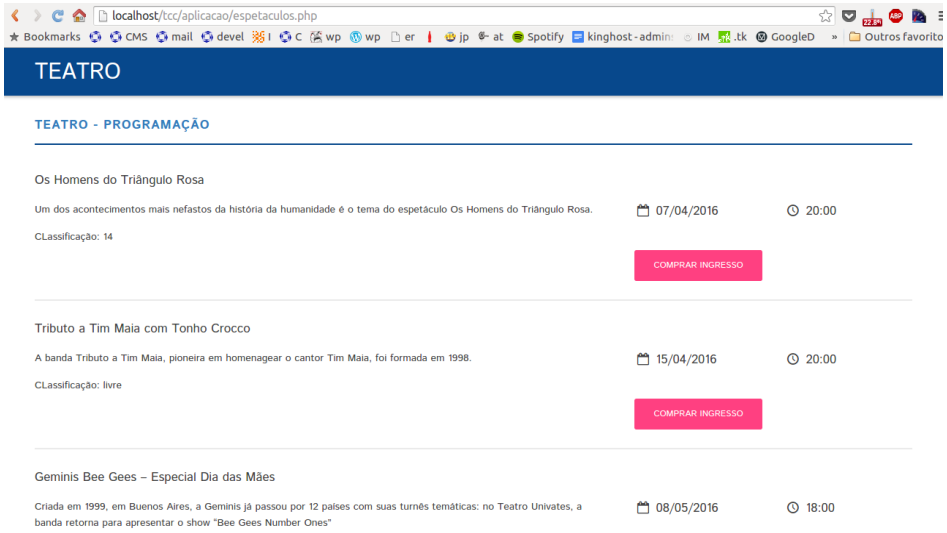


Fonte: Elaborado pelo autor (2016).

Como forma de ilustrar o funcionamento da aplicação, uma tela de exemplo de utilização de comunicação *Web Service* foi implementada, expondo assim a apresentação em um cenário real de listagem dinâmica de uma agenda de espetáculos. Para este exemplo, os dados foram tratados em um layout elaborado pelo autor, onde se utilizam marcadores de

texto HTML e estilos CSS para uma apresentação mais elaborada, apresentando assim um cenário mais próximo do utilizado em *websites* reais. O resultado é apresentado na figura 32.

Figura 32 - Página HTML de apresentação de dados obtidos via requisição Web Service



Fonte: Elaborado pelo autor (2016).

Os testes em laboratório transcorreram sem apresentar problemas na ferramenta proposta, o que qualificou e permitiu que a plataforma fosse levada à avaliação de profissionais de mercado, conforme apontado na metodologia. Para esta nova etapa de avaliação da ferramenta, foi proposto um novo cenário que será descrito no próximo capítulo, juntamente com a descrição dos avaliadores, método de avaliação e os resultados obtidos.

5.12 Comparação de ferramentas de mercado e plataforma proposta

Após a modelagem, desenvolvimento e testes da ferramenta proposta, se torna possível realizar a comparação entre ela, e as demais ferramentas de gestão de conteúdo apresentadas no capítulo 4. Para tal serão utilizados os mesmos critérios já apresentados na seção 4.10. A tabela 39, busca apresentar um comparativo da solução desenvolvida, frente às demais soluções de mercado existentes atualmente.

Tabela 42 - Comparação de ferramentas de mercado e plataforma proposta x funcionalidades

	C1	C2	C3	C4	C5	C6	C7	C8	C9	C10
BrowserCMS	Ruby	S	N	S	N	S	N	S	N	S
Django CMS	Phyton	S	N	S	N	S	N	S	N	N
Drupal	PHP	S	N	S	S	S	S	S	S	S
Joomla	PHP	N	N	S	N	S	S	S	S	N
KeystoneJS	JavaScript	S	S	N	S	N	S	S	S	S
Mezzanine	Phyton	S	N	S	N	N	N	S	S	N
PencilBlue	JavaScript	S	N	N	S	S	N	S	N	S
Refinerycms	Ruby	N	S	N	N	N	N	S	N	S
WordPress	PHP	N	N	N	S	N	S	S	S	N
Plataforma Proposta	PHP	S	S	N	S	N	S	S	N	S

Fonte: Elaborado pelo autor (2016).

Como pode ser observado na tabela 39, a plataforma proposta cumpri os objetivos propostos, dentro do escopo do trabalho. Não é possível afirmar que está é melhor que as demais, apenas que apresenta as funcionalidades apontadas pelos critérios de comparação escolhidos por este trabalho.

Os testes em laboratório transcorreram sem apresentar problemas na ferramenta proposta. Assim, a partir desta qualificação, foi possível disponibilizar a ferramenta para a avaliação de profissionais de mercado, conforme apontado na metodologia. Para esta nova etapa de avaliação da ferramenta, foi proposto um novo cenário que será descrito no próximo capítulo, juntamente com a descrição dos avaliadores, método de avaliação e os resultados obtidos.

6 AVALIAÇÃO DA PLATAFORMA

Para uma melhor avaliação da ferramenta proposta, foi proposto um cenário de testes, sendo este submetido a três avaliadores distintos. Seu objetivo foi avaliar cada área de atuação da plataforma (administração de estruturas de dados e o gerenciamento de permissões, gerenciamento de conteúdo e requisições de dados à plataforma), buscando assim uma avaliação mais completa de todos os objetivos abordados neste trabalho.

6.1 Cenário Escolhido

O cenário escolhido foi um cadastro de exposições artísticas, que compreende exposições de obras, fotografias e outras artes afins. Após análise de *websites* relacionados a espaços culturais, identificou-se que todos estes apresentam uma relação das atrações que estão sendo expostas, bem como um cronograma das exposições futuras, com seus respectivos locais. Constatou-se que as exposições por serem atemporais, são constantemente atualizadas por funcionários responsáveis, sendo assim um caso real de utilização de sistemas de gerenciamento de conteúdo.

Buscando cumprir os requisitos mínimos para uma apresentação das informações de exposições em um *website*, o teste buscou englobar o cadastro e administração de tipos de exposição, cadastro e administração de prédios e salas onde acontecerão as exposições, bem como cadastro e administração dos períodos e descrição das obras em questão.

Prezando pelo desenvolvimento de um cenário funcional e por uma correta avaliação da plataforma, os requisitos contemplados neste cenário foram:

- Cadastro e administração de prédios/locais das exposições;
- Cadastro e administração das salas/espaços que recebam as atrações artísticas;
- Cadastro e administração dos tipos de exposições;
- Cadastro e administração das exposições, contemplando o local, tipo e período destas;
- Disponibilização dos dados cadastrados por consultas externas ao sistema via requisições *Web Services*.

6.2 Modelo do Cenário Proposto

A estrutura proposta é constituída pelas estruturas “predio”, “sala”, “tipo” e “exposicao”, que serão apresentadas na tabela 40.

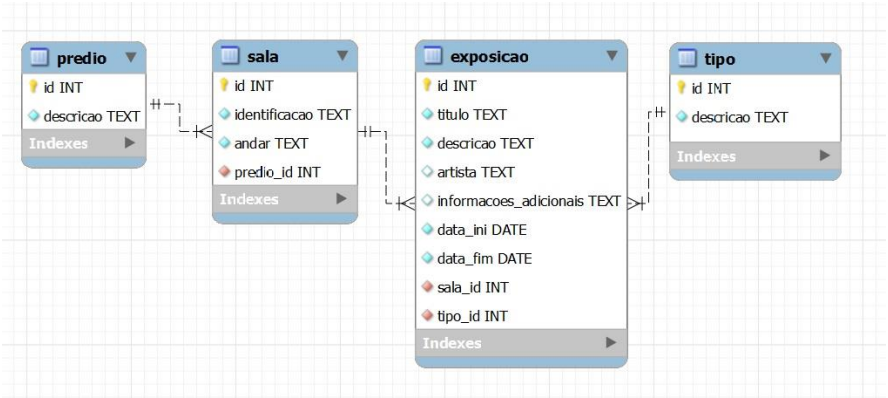
Tabela 43 - Detalhamento das tabelas

Nome da tabela	Descrição
predio	Tabela que busca representar os distintos prédios que recebem exposições.
sala	Tabela que identifica a sala ou espaço físico dentro de um prédio.
tipo	Tabela que identifica os variados tipos de exposições.
exposicao	Tabela que representa a descrição e informações relacionadas a cada exposição.

Fonte: Elaborado pelo autor (2016).

Buscando um melhor entendimento da arquitetura utilizada para o cenário de testes, foi elaborado um diagrama de modelo relacional apresentado na figura 33.

Figura 33 - Diagrama ER do cenário de testes



Fonte: Elaborado pelo autor (2016).

6.3 Método de Avaliação

Buscando uma maior efetividade na avaliação da plataforma, o processo foi dividido em três etapas distintas, onde num primeiro momento buscou-se avaliar a criação e administração de estruturas de dados e o gerenciamento de usuários e permissões; num segundo momento foi proposta a avaliação do gerenciamento de conteúdo; e num terceiro e último momento, as requisições de dados à plataforma por intermédio de *Web Services*.

Com o intuito de inserir os avaliadores ao contexto geral da plataforma, realizou-se uma prévia apresentação do sistema como um todo, contemplando assim a motivação do trabalho, estrutura proposta, bem como as dificuldades a serem supridas e funcionalidades que a plataforma proposta buscou implementar.

A avaliação de administração de estruturas de dados consistiu em disponibilizar o diagrama ER da figura 33, e solicitar o desenvolvimento desta dentro da plataforma. Para essa tarefa foi disponibilizado um *login* e senha com permissões administrativas ao sistema, dando acesso assim a todos os recursos disponíveis. Alguns tipos de dados foram pré-cadastrados, tais como campos de chave estrangeira, texto curto, texto longo e seleção única.

Para a avaliação das funcionalidades de gerenciamento de conteúdo, foi disponibilizado o acesso mediante *login* e senha de um usuário teste com permissão de gerenciamento (leitura, escrita e exclusão) das estruturas descritas na tabela 39.

Para a avaliação de requisição de dados por *Web Services*, foi solicitada a elaboração de um esboço de *website* utilizando os *Web Services* disponíveis pela plataforma. Neste teste, foram disponibilizados os exemplos de códigos de requisição *Web Service*s apresentados nas figuras 20 e 30, bem como a relação das funções disponíveis na tabela 28. Para a autenticação das requisições, foram disponibilizados os dados de acesso de um usuário e senha com permissão aos *Web Services*.

6.4 Caracterização dos Avaliadores

A avaliação da plataforma proposta foi realizada por três avaliadores com perfis distintos, buscando uma melhor análise de cada segmento da plataforma, contando-se com avaliadores com formação e experiência específica nos itens avaliados.

O avaliador A é um homem entre 24 e 28 anos, graduado em Sistemas da Informação pela UNIVATES, possui mais de 6 anos de experiência em levantamento de requisitos e desenvolvimento WEB, atualmente atua como analista de sistemas na empresa UNIVATES.

A avaliadora B é uma mulher entre 32 e 36 anos, pós-graduada em Marketing pela UNIVATES. Já atuou como coordenadora do portal da empresa UNIVATES, e atualmente coordena a equipe de imprensa desta instituição. Possui 13 anos de experiência com gestão de conteúdo, e na sua carreira já trabalhou com diferentes plataformas de gerenciamento de conteúdo, sendo elas: Joomla, Fred, WordPress e demais sistemas terceiros.

O avaliador C é um homem entre 24 e 28 anos, graduado em Análises de Sistemas pela UNIVATES, possui mais de 11 anos de experiência em desenvolvimento de sistemas, 9 deles dedicados ao desenvolvimento WEB. Já trabalhou com desenvolvimento de *websites* nas plataformas Joomla, WordPress, Fred, além de plataformas terceiras.

6.5 Questionário de Avaliação

Para a validação da ferramenta, e de acordo com a metodologia descrita no capítulo 3, foi elaborada uma pesquisa qualitativa buscando a opinião de especialistas no assunto para obter um parecer técnico sobre a solução proposta. Este questionário é composto por doze perguntas: três são questões específicas para avaliação de administração de estruturas de dados e gerenciamento de usuários e permissões; uma questão busca avaliar o gerenciamento de conteúdo; uma questão, as requisições de dados a plataforma; sete questões avaliam a plataforma como um todo.

Este questionário foi aplicado pessoalmente, logo após a utilização da plataforma, aos avaliadores apresentados no capítulo 6.4. O questionário aplicado aos avaliadores encontra-se no apêndice A. Os resultados obtidos, bem como a análise das respostas textuais, serão apresentados na seção 6.6.

6.6 Resultados da Avaliação

O primeiro aspecto do protótipo a ser avaliado, foi verificar quais as melhorias que o software pode agregar no dia a dia. Dentre os entrevistados, destacou-se a facilidade de criação de cadastros sem necessidade de interação direta com o banco de dados e a praticidade de utilização dos dados por chamadas *Web Services*. O entrevistado A ressaltou a elaboração dos cadastros sem a necessidade prévia de conhecimento SQL; a entrevistada B enalteceu a autonomia e agilidade para manutenção de informações; já o entrevistado C destacou a flexibilidade e adaptação dos requisitos dos cadastros e a vantagem de trabalhar com os dados por chamadas *Web Services*.

Como segundo e terceiro aspectos avaliados, buscou-se validar a navegabilidade e interface na ferramenta. De modo geral, os três avaliadores tiveram um parecer positivo quanto à navegação e interface na plataforma, não tendo maiores dificuldades para sua utilização. O avaliador A ressaltou a clareza das informações, por estarem dispostas de forma limpa, e sua navegação amigável em poucos cliques; já o avaliador C, sugeriu manter o menu aberto, permitir a personalização das cores, mensagens de confirmação e maiores informações, e ou, descrição sobre os campos. Já a avaliadora B não teve maiores sugestões sobre estes aspectos.

O quarto quesito avaliado foram os termos utilizados no software. Estes foram avaliados como adequados pelos avaliadores A e B e parcialmente adequados pelo avaliador C. Como sugestão, o avaliador C propôs utilizar os termos “Tipos de Cadastros” para “Gerenciamento de Estruturas”, por estes termos já serem bem difundidos em outras plataformas.

O quinto aspecto da avaliação deu abertura a sugestões e melhorias a serem acrescentadas a plataforma. De modo geral foram sugeridas melhorias de interfaces e possibilidade de personalizações. O avaliador A sugeriu que a plataforma disponibilizasse relatórios dos dados cadastrados. A avaliadora B sugeriu exemplos já preenchidos nos campos de cadastros, ou seja, o recurso *placeholder* do HTML; também propôs o recurso de pré-visualização, porém como os temas são trabalhados de forma independente na plataforma, neste modelo o recurso não poderia ser implementado. O avaliador C deu como sugestão a customização de cores e de logomarcas, deixando a plataforma com um caráter mais

comercial. Recursos como editores HTML e calendários foram sugeridos, sendo que já são suportados pela plataforma, na qual, onde junto ao cadastro de tipos, podem ser introduzidas instruções JavaScript e CSS para utilização e implementação de recursos semelhantes.

Como sexto aspecto avaliado, buscou-se analisar se a plataforma estaria madura para ser utilizada em ambiente real, tendo sido positiva a opinião dos três avaliadores neste aspecto. Apenas o avaliador C ressaltou que dificilmente uma ferramenta genérica atenderia a totalidade dos casos; enfatizou, porém, que a plataforma proposta atenderia sim à grande parte das necessidades impostas pelo mercado.

O sétimo e oitavo itens da avaliação englobaram analisar o quanto o software atende à necessidade de automatizar o desenvolvimento de cadastros para diferentes estruturas de dados, e o quanto o software atende à necessidade de criação e administração de estruturas de dados. Nos dois itens, os três avaliadores definiram como adequadas as afirmações. Foi ressaltada a facilidade imposta pela utilização de metamodelos e a funcionalidade de gerenciamento de cadastros oferecido. O avaliador A, porém, fez uma ressalva, em sistemas onde fosse necessário atender regras de negócios específicas, a plataforma proposta encontraria dificuldades de atendimento.

O nono item avaliado foi o funcionamento de gerenciamento de usuários e permissões na plataforma. A avaliação nos três casos foi dada como adequada, e todos os avaliadores comentaram a praticidade de poder definir individualmente as permissões por estrutura cadastrada.

Como décimo item avaliado, buscou-se verificar o quanto o software atende à necessidade de gerenciamento de conteúdo. Nenhum dos avaliadores apresentou problemas ao realizar o gerenciamento de conteúdo; os avaliadores A e B avaliaram como adequado, já o avaliador C, como parcialmente adequado, deixando sugestão relacionada quanto à interface de cadastros, propondo disponibilizar descrições dos campos, informações que auxiliariam os usuários na tarefa de realizar os cadastros.

O décimo primeiro item da avaliação consistiu em verificar a comunicação de dados por requisições externas ao sistema. Os três avaliadores apontaram como adequada, não tendo maiores sugestões. O avaliado C ressaltou o retorno dos dados em JSON, possibilitando a

integração direta com linguagens JavaScript, apontando que a estrutura de comunicação proposta possibilitaria a utilização de diferentes frameworks e CMS.

Ao final do questionário, abriu-se espaço para comentários e sugestões gerais sobre a plataforma. Segundo o avaliador A, a plataforma se mostra muito eficiente e com um retorno muito ágil para desenvolvimento de cadastros sob demanda. A avaliadora B ressaltou a fácil utilização e o fato de o software ser intuitivo; também teceu comentários sobre a flexibilidade da plataforma quanto a adicionar novos campos às estruturas, problema recorrente em outros CMS já utilizados. Já o avaliador C deu um retorno positivo sobre a plataforma como um todo, porém não fez nenhuma sugestão diferente das realizadas até o momento.

De maneira geral, todos os entrevistados acreditam que a ferramenta cumpriu seus objetivos propostos, proporcionando facilidades de criação e gerenciamento de estruturas de cadastros, e a camada de comunicação por *Web Services* se mostrou muito funcional. Apesar de não ter sido o foco do trabalho proposto, a interface de cadastro se mostrou muito simples e nenhum dos avaliadores apresentou problemas na sua utilização.

6.7 Trabalhos Futuros

O sistema desenvolvido neste trabalho buscou a elaboração de uma plataforma que realiza a administração e gerenciamento de dados para diferentes estruturas de dados. Como esta plataforma foi desenvolvida em caráter experimental, existe a possibilidade de se desenvolver diversos trabalhos que complementem a ferramenta, sendo alguns explicados a seguir.

Sobre a plataforma construída, poderá ocorrer o desenvolvimento de APIs de integração com softwares terceiros, tais como sistemas ERPs, intranets, e demais softwares empresariais, ampliando assim a diversidade e as funcionalidades já apresentadas neste trabalho. Estas API's poderiam, por exemplo, permitir a criação de formulários com dados auxiliares a partir de outros sistemas.

Outra possibilidade de trabalho futuro é tornar o software uma solução comercial, disponibilizando diferentes temas para facilitar o desenvolvimento de *websites* de forma mais eficiente e elegante, bem como a criação de recursos visuais como funcionalidades *drag and*

drop, e atualização de conteúdo em *frontend*, tornando a ferramenta mais atrativa para o público leigo.

Além das ideias já propostas, seria possível desenvolver *plugins* agregando novas funcionalidades à plataforma, buscando facilitadores para recursos como integrações com redes sociais, incorporação de vídeos, geração de relatórios, entre outras funcionalidades. Estes *plugins* trariam novos recursos e possibilidades à plataforma, dando-lhe um aspecto mais semelhante a outras soluções de mercado avaliadas neste trabalho, apresentadas no capítulo 4.

7 CONSIDERAÇÕES FINAIS

O presente trabalho teve por finalidade o desenvolvimento de uma plataforma de gerenciamento de conteúdo com foco em desenvolvimento e administração de estruturas dinâmicas, com uma camada de comunicação de dados isolada por intermédio de *Web Services*. Para seu desenvolvimento, foram realizados estudos apresentados no referencial teórico, bem como um estudo das principais ferramentas de CMS do mercado, para que estes proporcionassem uma maior compreensão das aplicações de gerenciamento de conteúdo. Baseado nestes estudos, desenvolveu-se um modelo proposto para a plataforma, que foi seguido no desenvolvimento da aplicação. Sua qualidade foi avaliada por especialistas na área. Não foram encontrados maiores problemas no desenvolvimento da plataforma proposta, sendo possível afirmar que o presente trabalho cumpriu todos os objetivos estabelecidos na metodologia.

Após a realização da pesquisa bibliográfica e de soluções de mercado, verificou-se a existência de lacunas para o desenvolvimento de cadastros dinâmicos sem a necessidade de conhecimento em instruções SQL, e uma comunicação de dados de forma isolada da plataforma por requisições terceiras. Objetivando o desenvolvimento utilizando boas práticas de engenharia de *software*, buscou-se primeiramente os requisitos, o desenvolvimento da modelagem, projeto da arquitetura e então, o desenvolvimento da aplicação.

Ao final do desenvolvimento da aplicação, e das avaliações realizadas pelos especialistas, pode-se concluir que a ferramenta proposta possui um grande potencial, e que sua estrutura possibilita um grande ganho de agilidade e praticidade no gerenciamento de

conteúdo, bem como sua flexibilidade na utilização de dados por requisições externas deixa a ferramenta mais atrativa para trabalhos com diferentes plataformas.

Como continuidade deste trabalho, serão propostas melhorias no sistema de gerenciamento de conteúdo da empresa UNIVATES, utilizando-se das estruturas desenvolvidas neste trabalho. Novas funcionalidades já mencionadas na seção sobre trabalhos futuros serão desenvolvidas, buscando assim um aspecto mais comercial para a plataforma, para sua utilização em projetos futuros de *websites*. No intuito de divulgar os resultados obtidos com este trabalho para com a comunidade acadêmica, buscar-se-á publicação destes resultados em meios científicos.

REFERÊNCIAS

ATKINSON, C., and KÜHNE, T., **Model-Driven Development**: A Metamodeling Foundation, IEEE Software, vol. 20, nº 5, (2003a), pp. 36-41.

BARROS, Aidil Jesus da Silveira; LEHFELD, Neide Aparecida de Souza. **Fundamentos de Metodologia Científica.** 2a ed. ampliada. São Paulo: Pearson Makron Books, 2000.

BOOTH, Grady; RUMBAUGH, James; JACOBSON, Ivar. **UML: Guia do Usuário**. 2ª Edição. Rio de Janeiro: Elsevier, 2005.

BROWSERCMS. Disponível em: <http://www.browsercms.org/>. Acesso em: 15 set. 2015.

CGI, Comitê Gestor da Internet no Brasil. **TIC Governo Eletrônico 2013**. São Paulo 2014 Disponível em: <http://cetic.br/media/docs/publicacoes/2/TIC_eGOV_2013_LIVRO_ELETRONICO.pdf>. Acesso em: 10 out. 2015.

DALL'OGLIO, Pablo. **Adianti Framework para PHP**. Lajeado: Edição do autor, 2012.

DJANGO CMS. Disponível em: <http://www.django-cms.org/en/>. Acesso em: 10 set. 2015

DRUPAL. Disponível em: <https://www.drupal.org/>. Acesso em: 22 set. 2015

FONSECA, J. J. S. **Metodologia da pesquisa científica.** Fortaleza: UEC, 2002.

FOWLER, Martin. **UML Essencial.** 3ª Edição. Porto Alegre: Bookman, 2005.

GUEDES, Gilleanes Thorwald Araujo - **Um Metamodelo UML para a Modelagem de Requisitos em Projetos de Sistemas MultiAgentes,** Universidade Federal do Rio grande do Sul Instituto de Informática Programa de Pós-Graduação em Computação, 2012

GERHARDT, Tatiana Engel; SILVEIRA, Denise Tolfo. **Métodos de Pesquisa.** 1A Edição. Porto Alegre: UFRGS, 2009.

GIL, Antonio Carlos. **Como elaborar projetos de pesquisa.** 5a Edição. São Paulo: Atlas, 2010.

GOLDENBERG, M. **A arte de pesquisar.** Rio de Janeiro: Record, 1997.

GOLDENBERG, Mirian. **A arte de pesquisar: como fazer pesquisas qualitativas em Ciências Sociais.** 4aed. Rio de Janeiro: Record, 2000.

GOMES, Daniel Adorno - **Web services SOAP em Java** Editora Novatec, 2009

IBGE. Disponível em: <http://www.ibge.gov.br />. Acesso em: 15 mai. 2016

JOOMLA. Disponível em: <https://www.joomla.org/>. Acesso em: 22 set. 2015

KAMPFFMEYER, Ulrich. **Content Management**, ECM Enterprise, 2006. Disponível em: < http://www.projectconsult.net/Files/ECM_White%20Paper_kff_2006.pdf>. Acesso em: 25 set. 2015.

KEYSTONEJS. Disponível em: <http://keystonejs.com/>. Acesso em: 22 set. 2015.

KNEWITZ, Marcos André. **Introdução ao desenvolvimento de software com UML**. São Leopoldo, RS: Editora UNISINOS, 2011.

KÜHNE, T., **Matters of (Meta-) Modelling**, In Journal on Software and Systems Modeling, Volume 5, Number 4, (2006), pp. 369-385.

MARCONI, Marina de Andrade; LAKATOS, Eva Maria. **Fundamentos de Metodologia Científica.** 5a Edição. São Paulo: Editora Atlas, 2003.

MAZZANINE. Disponível em: <http://mezzanine.jupo.org/>. Acesso em: 25 set. 2015.

MILANI, André, **PostgreSQL**: guia do programador. Editora Novatec, 2008.

NETO, João Souza; NETO, Arthur Nunes Ferreira - **METAMODELO DO FRAMEWORK COBIT DE GOVERNANÇA DE TI** - JISTEM - Journal of Information Systems and Technology Management Revista de Gestão da Tecnologia e Sistemas de Informação Vol. 10, No. 3, Sept/Dec., 2013 pp.521-540

O'BRIEN, L.; MERSON, P.; BASS, L. Quality Attributes for Service-Oriented Architectures. In: International Workshop on Systems Development in SOA Environments, IEEE Computer Society Washington, DC, USA, 7 p., 2007. Disponível em: <http://ieeexplore.ieee.org/xpl/freeabs_all.jsp?arnumber=4273291> Acesso em: 19 de set. 2015.

OBRIEN, James A.; MOREIRA, Cid Knipel. **Sistemas de informação e as decisões gerenciais na era da internet**. São Paulo: Saraiva, 2001;

OLIVEIRA, Jayr Figueiredo de. **Metodologia para Desenvolvimento de Projetos de Sistemas**. São Paulo, Érica 2001

PENCILBLUE. Disponível em: <https://pencilblue.org/>. Acesso em: 22 set. 2015

PHP. **PHP Web Site**. The PHP Group. Disponível em: <httm://www.php.net/>. Acesso em: 30 de set. 2015.

PRESSMAN, Roger S. **Engenharia de Software**. São Paulo: Makron Books, 1995.

PRESSMAN, Roger S. **Software Engineering: A practitioner's approach**. 7ª Edição, McGrawHill, 2009.

PFLEEGER, Shari Lawrence; Franklin, Dino. **Engenharia de Software: teoria e prática**. São Paulo: Prentice Hall, 2004.

REFINERYCMS. Disponível em: <http://www.refinerycms.com/>. Acesso em: 22 set. 2015

SANTOS, Marcelo Luis B.; Franco, Carlos Eduardo; Terra, José Cláudio C. - **Gestão de Conteúdo 360**, Editora Saraiva 2009.

SANTOS, Antonio Raimundo dos. **Metodologia científica: a construção do conhecimento.** 2a ed. Rio de Janeiro: DP&A editora, 1999.

SAUDATE, Alexandre - **SOA aplicado:** integrando com web services e além - Casa do Código 2012.

SOMMERVILLE, **Ian. Software Engineering**. 9ª Edição. Addison Wesley, 2010.

STRAHRINGER, S. (1996). Metamodellierungals Instrument des Methodenvergleichs, Shaker Verlag, Aachen.

TANAPANOFF, Kira - **Análise da Informação para tomada de decisões desafios e soluções**, Editora Intersaberes 2015

W3SCHOOLS. **The MVC Programming Model**. Disponível em: <http://www.w3schools.com/aspnet/mvc_intro.asp>. Acesso em: 30 set. 2015.

WAINER, Jacques. **Métodos de pesquisa quantitativa e qualitativa para a Ciência da Computação.** In: KOWALTOWSKI, Tomasz; BREITMAN, Karin; organizadores. Atualizações em Informática 2007. Rio de Janeiro: Ed. PUC-Rio; Porto Alegre: Sociedade Brasileira de Computação, 2007.

WORDPRESS. Disponível em: <https://wordpress.org/>. Acesso em: 24 set. 2015.

YORAM (Jerry) Wild, Vijay Mahajan, Robert E. Gunter. **Marketing de Convergência** (Estrategias para consquistar o novo consumidor), Pearson Education ABDR, 2003.

APÊNDICES

APÊNDICE A – Questionário disponibilizado aos avaliadores da plataforma.

Questionário de avaliação da plataforma desenvolvida para o TCCII

1 - Que tipo de melhoria o software poderia agregar no dia a dia?

_____

_____

2 - Como você avalia a navegabilidade deste software?

_____

_____

3 - Como você avalia a interface deste software?

_____

_____

4 - Como você avalia os termos/vocabulários que foram utilizados no software?

( ) Inadequados

( ) Parcialmente inadequados

( ) Indiferente

( ) Parcialmente adequados

( ) Adequados

5 - O que você considera que poderia ser melhorado ou acrescentado no software?

_____

_____

6 - Você considera que o software está maduro para ser utilizado em um ambiente real?

_____

_____

7 - O quanto o software atende a necessidade de automatizar o desenvolvimento de cadastros para diferentes estruturas de dados:

( ) Não atende
( ) Não atende parcialmente
( ) Indiferente
( ) Atende parcialmente
( ) Atende

8 - O quanto o software atende a necessidade de criação e administração de estruturas de dados:

( ) Não atende
( ) Não atende parcialmente
( ) Indiferente
( ) Atende parcialmente
( ) Atende

9 - O quanto o software atende a necessidade de gerenciamento de usuários e permissões:

( ) Não atende
( ) Não atende parcialmente
( ) Indiferente
( ) Atende parcialmente
( ) Atende

10 - O quanto o software atende a necessidade de gerenciamento de conteúdo:

( ) Não atende
( ) Não atende parcialmente
( ) Indiferente

( ) Atende parcialmente

( ) Atende

11 - O quanto o software atende a necessidade de comunicação com os dados por requisições externas ao sistema:

( ) Não atende

( ) Não atende parcialmente

( ) Indiferente

( ) Atende parcialmente

( ) Atende

12 - Comentários/sugestões:

_____

_____