UNIVERSIDADE TECNOLÓGICA FEDERAL DO PARANÁ
CURSO DE TECNOLOGIA EM ANÁLISE E DESENVOLVIMENTO DE SISTEMAS

PAULO HENRIQUE MERLIN

REPOSITÓRIO DE EXEMPLOS DE USO DE RECURSOS FRONT-END PARA DESENVOLVIMENTO WEB

TRABALHO DE CONCLUSÃO DE CURSO

PATO BRANCO
2016

PAULO HENRIQUE MERLIN

REPOSITÓRIO DE EXEMPLOS DE USO DE RECURSOS FRONT-END PARA DESENVOLVIMENTO WEB

Trabalho de Conclusão de Curso de graduação, apresentado à disciplina de Trabalho de Diplomação, do Curso Superior de Tecnologia em Análise e Desenvolvimento de Sistemas, da Universidade Tecnológica Federal do Paraná, Campus Pato Branco, como requisito parcial para obtenção do título de Tecnólogo.

Orientador: Andreia Scariot Beulke

PATO BRANCO
2016

ATA Nº: 279

DEFESA PÚBLICA DO TRABALHO DE DIPLOMAÇÃO DO ALUNO **PAULO HENRIQUE MERLIN**.

Às 17:00 hrs do dia 22 de junho de 2016, Bloco V da UTFPR, Câmpus Pato Branco, reuniu-se a banca avaliadora composta pelos professores Andréia Scariot Beulke (Orientadora), Beatriz Terezinha Borsoi (Convidada) e Soelaine Rodrigues Ascari (Convidada), para avaliar o Trabalho de Diplomação do aluno Paulo Henrique Merlin, matrícula 749222, sob o título **Repositório para Armazenamento e Consulta de Exemplos de Uso de Recursos de HTML 5, CSS3 e JavaScript** ; como requisito final para a conclusão da disciplina Trabalho de Diplomação do Curso Superior de Tecnologia em Análise e Desenvolvimento de Sistemas, COADS. Após a apresentação o candidato foi entrevistado pela banca examinadora, e a palavra foi aberta ao público. Em seguida, a banca reuniu-se para deliberar considerando o trabalho **APROVADO**. Às 17:30 hrs foi encerrada a sessão.

_____
Profa. Andréia Scariot Beulke, Esp.
Orientadora

_____
Profa. Beatriz Terezinha Borsoi, Dr.
Convidada

_____
Profa. Soelaine Rodrigues Ascari, M.Sc.
Convidada

_____
Profa. Eliane Maria de Bortoli Fávero, M.Sc
Coordenadora do Trabalho de Diplomação

_____
Prof. Edilson Pontarolo, Dr.
Coordenador do Curso

À minha mãe Fátima “in memoriam” e à minha Vó Leny – vocês sempre estarão no meu coração e nos meus pensamentos.

AGRADECIMENTOS

Certamente estes parágrafos não irão mencionar todas as pessoas que fizeram parte desta importante fase da minha vida. Portanto, desde já me desculpo àquelas que não estão entre essas palavras, mas tenham certeza que fazem parte dos meus pensamentos e de minha gratidão.

Agradeço à minha Vó Leny, ao incentivo, apoio e estímulo para enfrentar as barreiras da vida, à minha mãe Fátima, que do céu me protege e me guia pelo bom caminho e aos demais integrantes da minha família. À Jacob pela amizade e carinho.

Agradeço aos meus professores, que com muita determinação e companheirismo me ajudaram a chegar até aqui, especialmente a Professora Beatriz Terezinha Borsoi que desde o primeiro dia de aula acreditou no meu potencial. Minha orientadora, Professora Andreia Scariot Beulke pela amizade, paciência e dedicação demonstrada. Professora Soelaine Rodrigues Ascari, pela amizade e incentivo em relação ao intercâmbio que participei. Enfim, a todos que, embora não nominados, contribuíram para minha formação acadêmica.

Agradeço aos meus colegas de classe, especialmente João Guilherme Brasil Pichetti, que não mediu esforços para me ajudar quando tive dúvidas, sempre demonstrando muita amizade e companheirismo. Aos demais colegas, Wiglan Mariani, Lucas Decol e Luan Szady, meus sinceros agradecimentos.

Essa caminhada jamais seria possível sem a amizade, auxílio e companheirismo de todos vocês. Sou muito grato e peço que Deus retribua a todos com muita paz, saúde e alegrias.

RESUMO

MERLIN, Paulo Henrique. Repositório de exemplos de uso de recursos front-end para desenvolvimento web. 2016. 54f. Monografia (Trabalho de Conclusão de Curso) - Curso Superior de Tecnologia em Análise e Desenvolvimento de Sistemas, Universidade Tecnológica Federal do Paraná, Câmpus Pato Branco. Pato Branco, 2016.

As aplicações web têm evoluído muito desde os primeiros sites que permitem interatividade com o usuário e uso de banco de dados. Inicialmente esses sistemas eram baseados em *HyperText Markup Language* (HTML) e formulários com componentes extremante simples. Atualmente há uma categoria, relativamente recente, dessas aplicações que utiliza componentes de interação semelhantes às aplicações *desktop*, com bem mais recursos nesse aspecto de interação, que as caracteriza como ricas. Elas são as *Rich Internet Apllication* (RIA). Embora a caracterização "rica" seja decorrente das funcionalidades de interação, essas aplicações também se caracterizam pela redução de tráfego de dados entre cliente e servidor, processamento assíncrono e outros aspectos técnicos. A definição de interfaces mais ricas, que é a ênfase deste trabalho, deve-se aos recursos das tecnologias utilizadas para desenvolvê-las. Entre esses recursos estão *Cascading Style Sheets* (CSS) que é o CSS3, HTML5, Bootstrap e JavaScript. Esse último, mais voltado para a realização de processamento no lado cliente para a validação e implementação de campos calculados, por exemplo. Considerando a grande expansão de uso dessas tecnologias, este trabalho apresenta uma proposta de repositório para armazenar exemplos de uso de recursos dessas tecnologias. O objetivo é que indivíduos que queiram aprender sobre essas tecnologias tenham uma base de exemplos para consulta. Outras tecnologias podem ser agregadas ao repositório. Essas são as citadas por serem as prioritariamente utilizadas para o desenvolvimento da interface do aplicativo. Além de consultar os exemplos armazenados no repositório, o usuário também pode sugerir (incluir) exemplos, denominados itens, das categorias. Esses exemplos incluídos por usuários são validados pelo administrador do sistema antes de serem disponibilizados para consulta.

Palavras-chave: CSS3. HTML5. Bootstrap. JavaScript. RIA.

ABSTRACT

MERLIN, Paulo Henrique. 2016. Repository of examples for use in front-end features for web development. 54f. Monografia (Trabalho de Conclusão de Curso) - Curso Superior de Tecnologia em Análise e Desenvolvimento de Sistemas, Universidade Tecnológica Federal do Paraná, Pato Branco Campus. Pato Branco. 2016.

Web applications have evolved a lot since the first e-commerce sites (or other applications which provided interactivity between the user and the usage database). Although many were based on HyperText Markup Language (HTML) and formed with extremely simple components, there now exists an extensive category of applications that use similar interaction components as desktop applications. These components create more resources in this aspect of interaction and can likewise be characterized as rich, collectively referred to as the Application Rich Internet (RIA). Although the characterization "rich" is a result of interaction features, these applications are also characterized by data traffic reduction between client and server, asynchronous processing and other technical aspects. The definition of richer interfaces, which is the emphasis of this work, is categorized by the technological resources used to develop these interfaces, which in this research included Cascading Style Sheets 3 (CSS), CSS3, HTML5, Bootstrap and JavaScript. Of these, the latter is friendlier in performing processing tasks on the client side implementation for validation and calculated fields. Considering the increasing expansion of use of these technologies, this paper proposes a repository to store examples of these technological resources. The goal is for people who want to learn about these technologies to have examples for reference. While other technologies can be added to the repository, these are cited as being primarily used for the development of the application interface. In addition to reviewing the examples stored in the repository, you can also suggest attach examples into the repository, such as items or categories. These examples uploaded by users will be validated by the system administrator before being available for consultation.

Key words: CSS3. HTML5. Bootstrap. JavaScript. RIA.

LISTA DE FIGURAS

LISTA DE QUADROS

LISTAGENS DE CÓDIGO

LISTA DE ABREVIATURAS E SIGLAS

API	*Application Programming Interface*
CSS	*Cascading Style Sheets*
DOM	*Document Object Model*
HTML	*HyperText Markup Language*
HTTP	*Hypertext Transfer Protocol*
JPA	*Java Persistence API*
RIA	*Rich Internet Applications*
URL	*Uniform Resources Locator*
UTFPR	Universidade Tecnológica Federal do Paraná
XHTML	*Extensible Hypertext Markup Language*
XML	*Extensible Markup Language*
W3C	*World Wide Web Consortium*

SUMÁRIO

1 INTRODUÇÃO

Este capítulo apresenta as considerações iniciais, os objetivos, a justificativa e a organização do texto que indica os capítulos subsequentes deste trabalho.

1.1 CONSIDERAÇÕES INICIAIS

A evolução das aplicações *web* tradicionais, baseadas em *HyperText Markup Language* (HTML) e *Hypertext Transfer Protocol* (HTTP), conduziu às aplicações *web* denominadas ricas. As *Rich Internet Applications* (RIA) são caracterizadas por serem aplicações *web* com possibilidade de processamento no lado cliente, comunicação assíncrona entre cliente e servidor, alta usabilidade e apresentação de conteúdo mais atrativo ao usuário (DRIVER; VALDES; PHIFER, 2005, STEARN, 2007).

As RIAs oferecem uma experiência interativa e responsiva para o usuário (LAWTON, 2008), com interface gráfica denominada rica pelos componentes e formas de interação que elas provêm e por permitirem respostas rápidas (CAMERON, 2004) pela interação assíncrona entre cliente e servidor. As RIAs remontam aplicações *desktop* que são entregues com funcionalidades baseadas em *web* (DISSANAYAKE; DIAS, 2014). Com a introdução de HTML5 e *Cascading Style Sheets* (CSS 3), as RIAs tem se tornando mais proeminentes em termos de portabilidade e disponibilidade (CORREIA, 2013).

HTML5 é uma linguagem de marcação atualizada que tem o objetivo de melhorar a linguagem HTML fornecendo suporte às mais recentes multimídias, ao mesmo tempo em que se mantém facilmente utilizável por pessoas e consistentemente compreensível por computadores e dispositivos (STRÍBNÝ; SMUTNÝ, 2013).

CSS3 é a versão recente do CSS que permite definir estilos para páginas *web* (W3C, 2016). CSS é uma "folha de estilo" composta por “camadas” e utilizada para definir a apresentação (aparência) de páginas na Internet que foram desenvolvidas em linguagens de marcação como *Extensible Markup Language* (XML), HTML e *Extensible Hypertext Markup Language* (XHTML). O CSS define como serão exibidos os elementos contidos no código de uma página da Internet, fazendo a separação entre o formato e o conteúdo de um documento (TECMUNDO, 2016).

HTML5 tem realizado um papel importante no desenvolvimento de RIAs pelas funcionalidades que fornece e que facilitam a implementação das características das

aplicações denominadas ricas. Por ser uma tecnologia ainda relativamente nova a apresentação de exemplos e modelos dos seus conceitos é relevante para os acadêmicos e profissionais que querem ou precisam aprender a utilizar os recursos de HTML5, associado com CSS3 e JavaScript, por exemplo, para o desenvolvimento da parte cliente de uma aplicação *web* rica. Este trabalho apresenta uma proposta de repositório desses conceitos e exemplos de utilização de recursos dessas tecnologias.

1.2 OBJETIVOS

A seguir são apresentados o objetivo geral e os objetivos específicos do desenvolvimento do trabalho.

1.2.1 Objetivo Geral

Implementar um repositório, apresentado como site *web*, para armazenamento e consulta de exemplos de uso de recursos de HTML5, CSS3 e JavaScript.

1.2.2 Objetivos Específicos

- Desenvolver um exemplo de site *web* utilizando tecnologias como HTML5, CSS e JavaScript.
- Propor um repositório para que alunos de disciplinas vinculadas ao desenvolvimento *web* possam consultar e cadastrar modelos de uso de recursos de HTML5, CSS e JavaScript.
- Possibilitar que usuários do repositório possam incluir sugestões de exemplos nas categorias apresentadas.

1.3 JUSTIFICATIVA

A evolução no desenvolvimento *web* é extremamente notável, principalmente se comparadas as interfaces *web* mais antigas com as atuais. A competição em relação as melhorias no desenvolvimento resultaram em importantes avanços, inclusive nas tecnologias utilizadas. Essas novas tecnologias permitem o desenvolvimento de interfaces mais

profissionais, aprimorando vários aspectos como: a agilidade no desenvolvimento, os códigos das aplicações ficaram mais legíveis, há a separação entre as marcações e estilizações, tudo isso interferindo positivamente durante a manutenção do projeto.

A aplicação desenvolvida, como resultado deste trabalho visa propor um repositório de exemplos de recursos de tecnologias utilizadas para desenvolvimento *web* utilizando tecnologias *front-end*. O sistema permite cadastrar conceitos, exemplos de código e imagens que ilustram os códigos.

O usuário logado no sistema (administrador) tem permissões para fazer a inclusão de recursos, categorizados por grupos. Os demais usuários (sem *login)* poderão cadastrar itens de uma categoria previamente cadastrada pelo administrador, que são exemplos de uso dos recursos contidos no repositório. Esses itens deverão ser validados por um usuário com *login* para serem incluídas no sistema e apresentados juntamente com o recurso.

1.4 ESTRUTURA DO TRABALHO

Este texto está organizado em capítulos. O Capítulo 2 apresenta o referencial teórico que discorre sobre as principais tecnologias para desenvolvimento web *front-end*. O Capítulo 3 apresenta os materiais e o método utilizado no desenvolvimento deste trabalho. No Capítulo 4 são apresentados os resultados obtidos com a realização do trabalho. O Capítulo 5 apresenta a conclusão. Por fim, estão as referências bibliográficas utilizadas na composição deste texto.

2 REFERENCIAL TEÓRICO

Este capítulo apresenta o referencial teórico deste trabalho que se refere às aplicações Internet ricas, à linguagem de marcação de hipertexto e às folhas de estilo para formatação de páginas *web*.

2.1 RICH INTERNET APPLICATION

Recentemente um novo tipo de aplicações *web* ganhou popularidade: as chamadas *Rich Internet Application* (RIA), traduzido como aplicações Internet ricas. Essas aplicações trouxeram alguns benefícios (BENJAMIN et al., 2010, p. 404):

a) computação (ou processamento) significativa pode ser realizada no lado cliente;
b) o estado da aplicação no lado cliente pode ser modificado sem comunicação com o servidor;
c) tecnologias como Ajax tornam possível enviar requisições assíncronas ao servidor e manipular a resposta em background, sem recarregar a página no cliente.

As RIAs podem ser conceituadas como aplicações *web* que tendem a ter características similares e funcionalidades de interação como as aplicações *desktop* (PEINTNER; KOSCH; HEUER, 2009).

Entre as tecnologias disponíveis no lado cliente e para a construção da interface da aplicação que a caracterizam como RIA estão a linguagem de marcação de hipertexto em sua versão 5 e as folhas de estilo em sua versão 3.

2.2 HYPERTEXT MARKUP LANGUAGE

Hyper Text Markup Langauge, HTML, foi a primeira linguagem a usar *tags* de marcação para descrever e formatar a apresentação do conteúdo de páginas *web* nos anos 90. HTML é uma linguagem em evolução, passando pelas versões 2.0, 3.0, 3.2, 4.0 e 5. A primeira versão definitiva foi a 2.0 e a versão 3.0 fornece suporte para tabelas, imagens, títulos e outros elementos. HTML 4.0 integra funcionalidades de CSS e JavaScript, levando o conceito de páginas *web* estáticas para páginas dinâmicas, fornecendo uma grande plataforma para o desenvolvimento de aplicações *web*. HTML 5 introduziu métodos de armazenamento de dados que podem permitir que as aplicações *web* possam oferecer melhorias significativas

de desempenho, segurança e privacidade, sem o uso de linguagens suplementares (NASEEM; MAJEED, 2013).

A marca inicial de representação livremente acessível para HTML foram as chamadas *tags* HTML, publicadas inicialmente na Internet por Berns-Lee em 1991 (FREITAG, 1998). A HTML 3.2 incorporou tabelas, *applets*, junção de conteúdo e imagens. Em 1997 a *World Wide Web Consortium* (W3C) adotou oficialmente a HTML 3.2 como padrão industrial de HTML. A versão 3.2 foi melhorada por instituições visando incorporar os navegadores *web* mais utilizados no mercado, como o Netscape e Microsoft (NASEEM; MAJEED, 2013). HTML 4.0 trouxe o uso de novos atributos em folhas de estilo e elementos de *script*. Usando JavaScript os programadores podem desenvolver aplicações *web* múltiplas (GONZALEZ; MARCELIN-JIMENEZ, 2011).

2.2.1 HTML5

HTML5 é a evolução da linguagem HTML. Está sendo desenvolvida desde 2006 pela W3C em parceria com a WHATWG. Essa versão da linguagem une o que há de melhor no HTML versão 4, XHTML, e HTML DOM (*Document Object Model*) e adiciona um conjunto de novos elementos, atributos e *Application Programming Interface (*API) (TERUEL, 2011).

A HTML 5 oferece armazenando seguro de dados em duas formas: de seção e local (ou *web*). No armazenamento de seção, os dados são armazenados em janelas, exceto nos navegadores que armazenam os dados. No armazenamento local, os dados são armazenados no lado do cliente (NASEEM; MAJEED, 2013).

HTML5 trouxe novas funcionalidades e APIs que participam de uma concepção e desenvolvimento de serviços visando atender as exigências dos usuários. Entre essas APIs estão as que permitem o armazenamento local, sem dependências de *plugins* ou bibliotecas adicionais (JEMEL; SERHROUCHNI, 2014). Essas APIs provaram a sua capacidade para melhorar o desempenho de aplicações *web* e a qualidade da experiência dos usuários. No entanto, como ressaltam esses autores, além do desempenho, o usuário precisa ter certeza sobre a confidencialidade e disponibilidade dos seus dados.

A HTML5 está em contínua evolução. Várias bibliotecas, como AJAX e JQuery, foram escritas para oferecer ao programador a capacidade de tirar vantagens das novas características apesar das diferenças na capacidade de cada navegador (FRANKSTON, 2014).

Os navegadores atuais são ricos ambientes de programação, com textos, vídeos, e capacidade de apresentação de áudio, vetores e gráficos baseados em *pixels*. A aplicação

também pode usar armazenamento local em plataformas, de modo que possa funcionar como uma aplicação *stand-alone* (FRANKSTON, 2014).

HTML5 é um grande avanço para o desenvolvimento de padrões *web*, pois possui recursos para a padronização de conteúdo *web* e de funcionalidades interativas (JIANPING; JIE, 2010).

A HTML5 criou novos elementos com forte valor semântico com o propósito de melhorar a acessibilidade por meio da marcação correta para cada conteúdo.

2.2.1.1 Novos recursos e vantagens do HTML5

A HTML5 fornece um conjunto de novos elementos para marcar os conteúdos a serem disponibilizados na *web*. Esses novos elementos tenham por objetivo fornecer um valor semântico para possibilitar a acessibilidade desses conteúdos. Por exemplo, em versões anteriores, para criar um bloco de conteúdo utiliza-se a *tag* que não possui nenhum valor semântico. Com a marcação adequada dos conteúdos, os leitores de tela interpretam os elementos, possibilitando acessibilidade para, por exemplo, acessar via teclado os conteúdos disponibilizados na *web*.

HTML5 adiciona diversas características sintáticas que incluem elementos para <video>, <audio> e <canvas> e a integração de vetores gráficos escaláveis e MathML para fórmulas matemáticas (STRÍBNÝ; SMUTNÝ, 2013).

Há ainda os elementos especificamente concebidos para identificar a estrutura de leiaute da página como: <header>, <section>, <nav>, <article>, <footer>, elementos de semântica, como <aside>, <figure>, <dialog>, elementos em linha, como, <time>, <meter> e <progress> e elementos iterativos, como, <details>, <datagrid> e <command> (JIANPING; JIE, 2010).

O *canvas label type* é um recurso novo e interessante porque permite aos desenvolvedores usar o rótulo diretamente na página para fazer gráficos em 2D ou por meio de linguagem e JavaScript, chamar o OpenGL ou DirectX e fazer a renderização do gráfico para 3D com efeitos e interatividade (JIANPING; JIE, 2010, STRÍBNÝ; SMUTNÝ, 2013).

Dentre as vantagens e novidades da HTML 5 estão (TERUEL, 2011): redução da necessidade de *plugins* externos; disponibilização de elementos para substituir diversos *scripts* utilizados atualmente; renderização de elementos semelhantes em diversos tipos de dispositivos e navegadores; novas APIs para manipulação de conteúdo *off-line*, acesso ao

banco de dados, validação de formulários, criação de gráficos e desenhos, etc.; novos tipos de campos para formulários, como: *calendar, time, date, email, url, search*, etc.

Além de novos elementos a atributos, HTML5 não traz diversos atributos de estilização e formatação que são melhores utilizados com a linguagem CSS. Além disso, não traz elementos que comprometem a usabilidade e a acessibilidade (TERUEL, 2011).

Outra vantagem no uso da HTML5, juntamente com o CSS3 é a respeito do design responsivo, que consiste em adaptar o site ao *browser*, sem a necessidade de definir várias folhas de estilos específicas para cada resolução, ou seja, é um tipo de design onde o leiaute varia de acordo com a resolução do equipamento utilizado pelo usuário. Esse tipo de design tem origem em 2010, e tem como objetivo melhorar a experiência de navegação dos usuários de páginas web (DEVMEDIA, 2016).

Um recurso muito importante, quando se trabalha com responsividade, ou design responsivo de um aplicativo *web*, são as *media queries*, pois definem condições de um CSS específico, ou seja, se o aplicativo se adequar a todas as condições, o CSS será aplicado. As *medias queries* possuem operadores lógicos que poderão ser usados de acordo com a necessidade que a aplicação requer. As *medias features* permitem diferenciar um dispositivo do outro de acordo com as características dos mesmos, como altura, largura, orientação e resolução do dispositivo (DEVMEDIA, 2016). As medias mostram algumas, das muitas, vantagens e recursos que a HTML5 pode oferecer aos usuários.

2.3 CSS

CSS é a abreviação para o termo *Cascading Style Sheet*, CSS, que traduzido ao português significa folha de estilos em cascata. Uma definição, para esse termo, pode ser encontrada no site da W3C na página relacionada à CSS: folha de estilo em cascata é um mecanismo simples para adicionar estilos como fontes, cores e espaçamentos aos documentos *web* (SILVA, 2012).

A primeira versão das especificações do W3C para as folhas de estilos em cascata foi para as CSS nível 1, lançada em 17 de dezembro de 1996, um único documento, que está dividido em nove seções e cinco apêndices. Em 12 de maio de 1998, foi lançada a versão para as CSS nível 2, também em documento único, contendo dezenove seções e oito apêndices. Em 2 de agosto de 2002, foi proposta a revisão da especificação, o que deu origem a versão das CSS nível 2.1, essa versão passou para o *status* de recomendação do W3C em 7 de junho de 2011 (SILVA, 2012).

O conjunto de funcionalidades de níveis anteriores é, sempre, um subconjunto das funcionalidades da versão atual, portanto por ela plenamente suportado. Isso significa que um agente de usuário que suporte as funcionalidades das CSS de nível atual, suportará também todas as funcionalidades das CSS de níveis anteriores (SILVA, 2012).

2.3.1 CSS3

O modelo de desenvolvimento das especificações para as CSS3 alterou o modelo adotado para as versões anteriores. Enquanto essas foram desenvolvidas em um documento único, as CSS3 estão sendo desenvolvidas em módulos (SILVA, 2012, p. 43).

Na versão 3 do CSS as especificações existentes foram organizadas em assuntos e para cada assunto foi criada uma especificação independente. As especificações para seletores CSS3 constituem um módulo, as bordas e fundo, outro módulo, as cores, outro e assim por diante. Cada módulo é desenvolvido de forma independente e segue um cronograma próprio (SILVA, 2012).

A grande vantagem da modularização está no fato de que o desenvolvimento das funcionalidades de um módulo não mais depende do andamento dos outros módulos, como ocorreu com o desenvolvimento das versões anteriores das CSS. Essa independência possibilita que fabricantes comecem a implementar, funcionalidades previstas em módulos adiantados no seu desenvolvimento (SILVA, 2012).

Cazenave, Quint e Roisin (2011) ressaltam que com a utilização dos recursos do HTML5 e do CSS3, os programadores podem desenvolver as aplicações *web* com conteúdo mais claramente visível, pois o HTML5 fornece *tags* específicas para a criação de determinados conteúdos. Por exemplo, para criar uma barra de navegação pode ser utilizada a *tag* <nav> ao invés de utilizar a *tag* genérica . Assim, a estilização desses conteúdos torna-se mais eficientes devido à declaração de elementos únicos e bem definidos.

3 MATERIAIS E MÉTODO

Este capítulo apresenta os materiais e o método utilizados para a modelagem e o desenvolvimento de um repositório de exemplos de uso de recursos *front-end* para desenvolvimento *web*.

3.1 MATERIAIS

No Quadro 1 estão as ferramentas e as tecnologias utilizadas para o desenvolvimento da aplicação que resultado da realização deste trabalho.

Ferramenta / Tecnologia	Versão	Referência	Finalidade
Java	1.8.0_73	https://www.oracle.com/java/index.html	Linguagem de programação.
SpringBoot	1.3.3. RELEASE	https://projects.spring.io/spring-framework/	Framework para suporte a injeção de dependência e gerenciamento de transações.
HTML	5	https://www.w3.org/TR/html5/	Linguagem de marcação para definição da interface.
CSS	3	https://www.w3.org/TR/CSS/	Estilização da interface.
Bootstrap	3.3.6	http://getbootstrap.com/	Desenvolvimento da interface.
Thymeleaf	2.1.4	http://www.thymeleaf.org	Serve para manipular e organizar conteúdo HTML de forma dinâmica no lado servidor.
Astah Community	6.8.0	http://astah.net/editions/community	Diagrama de classes
MySql	5.7	https://www.mysql.com/	Banco de dados.
MySql Workbench	6.0.7	https://www.mysql.com/products/workbench/	Diagrama de entidade e relacionamento.

Quadro 1 – Ferramentas e tecnologias de modelagem e implementação

3.2 MÉTODO

O sistema proposto neste trabalho foi desenvolvido de acordo com as seguintes etapas:

a) Levantamento de requisitos: definidos após reunião com alguns professores do Curso de Tecnologia em Análise e Desenvolvimento de Sistemas da Universidade Tecnológica Federal do Paraná, Câmpus Pato Branco. Esses professores comentaram a necessidade de desenvolver um repositório que pudesse ser utilizado para apresentar exemplos de recursos de tecnologias como: HTML5, CSS3 e JavaScript. Exemplos de

recursos de quaisquer tecnologias podem ser incluídos no repositório. Essas foram as inicialmente pensadas por serem as utilizadas no desenvolvimento da interface da aplicação.

b) Análise e projeto: a modelagem foi realizada visando representar os recursos e funcionalidades que do aplicativo.

c) Implementação: realizada utilizando as tecnologias apresentadas no Quadro1.

d) Testes: realizados informalmente e com o objetivo de verificar os erros de código e se os requisitos pré-estabelecidos haviam sido atendidos. Uma versão final do aplicativo será instalada em um servidor do próprio Departamento Acadêmico de Informática da UTFPR (Universidade Tecnológica Federal do Paraná) para que os usuários possam testá-lo.

4 RESULTADOS

Este Capítulo apresenta o resultado deste trabalho que é a modelagem de um sistema *web*, cuja função é ser um repositório de armazenamento e consulta de exemplos de uso de recursos de HTML5, CSS3, Bootstrap, entre outros. No capítulo também constam códigos que visam exemplificar como a implementação foi realizada com o objetivo de estudo das tecnologias. A implementação será efetivada como trabalho de conclusão de curso.

4.1 ESCOPO DO SISTEMA

A aplicação *web*, resultado deste trabalho, permite cadastrar exemplos da utilização de tecnologias (conceitos, códigos e *views*) visando disponibilizar um repositório para que o usuário possa consultar esses exemplos para fins de estudos. No sistema o administrador cadastrará categorias e itens dessas categorias. Esses itens se referem aos exemplos de códigos que poderão ser acessadas via *web* para auxiliar os indivíduos que desejam aprender essas tecnologias. O usuário poderá registrar exemplos de tecnologias (conceito, código e *view*) e esses exemplos serão validados pelo administrador antes de serem disponibilizados para consulta.

4.2 MODELAGEM DO SISTEMA

O Quadro 2 apresenta a listagem dos requisitos funcionais identificados para o sistema.

Identificação	Nome	Descrição
RF01	Manter categoria	Incluir, editar, consultar e excluir as categorias que serão exibidas no sistema.
RF02	Manter itens	Incluir, editar, consultar e excluir os itens de cada categoria.
RF03	Sugerir inclusões	As sugestões de inclusão serão enviadas por usuários sem *login*, e ficarão aguardando aprovação do administrador.
RF04	Validar inclusões	Após fazer o *login* no sistema, o administrador será redirecionado para uma página que mostrará a lista com todas as sugestões incluídas por usuários sem *login*, para efetuar a aprovação ou não das mesmas.

Quadro 2 – Requisitos funcionais

O Quadro 3 apresenta o requisito não-funcional identificado para o sistema, também denominado de requisito suplementar. Esses requisitos, explicam as regras de negócios, restrições ao sistema de acesso, requisitos de qualidade, desempenho, segurança e outros.

Identificação	Nome	Descrição
RNF01	Acesso ao sistema	O acesso ao sistema será realizado por meio de *login* e senha e apenas o administrador terá acesso.

Quadro 3 – Requisito não funcional

O diagrama de casos de uso apresentado na Figura 1 contém as funcionalidades essenciais do sistema realizados pelos seus atores que são: administrador e usuário. O administrador é responsável por manter as categorias, itens, usuários e validar as inclusões feitas pelos usuários. Os usuários, por sua vez, poderão consultar as categorias e itens, previamente cadastrados pelo administrador, e sugerir inclusões de itens nas categorias.

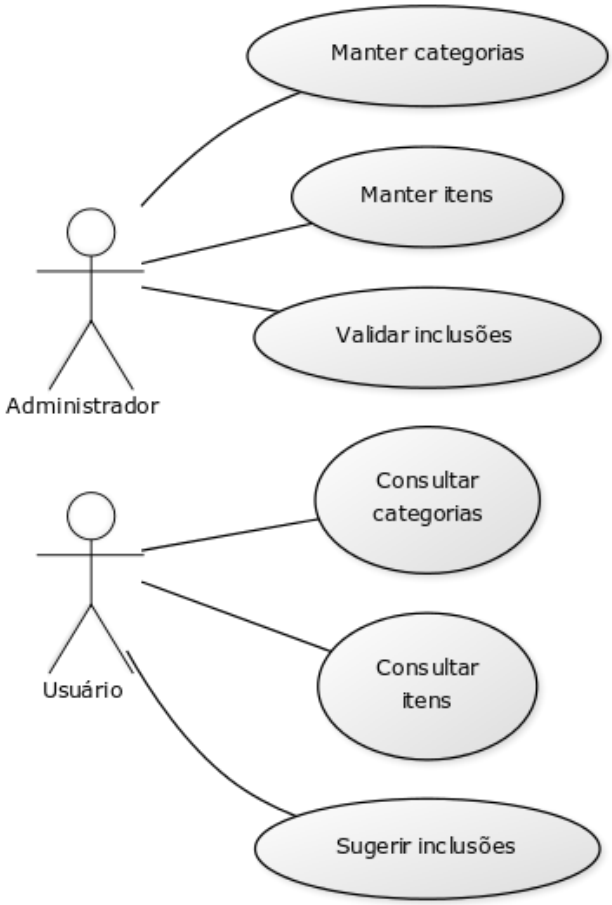

Figura 1 – Diagrama de casos de uso

O caso de uso denominado “Manter categorias”, que é realizado pelo administrador é apresentado no Quadro 4.

<table>
<tr><td colspan="2">Caso de uso:
Manter categorias.
Descrição:
Realização do cadastro de categorias no sistema.
Evento Iniciador:
Necessidade de incluir, excluir, consultar ou alterar categorias do sistema.
Atores:
Administrador.
Pré-condição:
Existência dos dados essenciais para realizar o cadastro.
Sequência de Eventos:
1. Ator seleciona formulário no qual deseja realizar a operação.
2. Sistema apresenta o formulário.
3. Ator realiza a operação desejada: incluir, excluir, consultar e alterar usuários.
4. Sistema verifica se os dados para a operação estão consistentes e realiza a operação.
Pós-Condição:
Operação de inclusão, exclusão, consulta e alteração de usuários do sistema.</td></tr>
<tr><td>Nome do fluxo alternativo (extensão)</td><td>Descrição</td></tr>
<tr><td>Linha 4: Dados não são válidos.</td><td rowspan="2">4.1 No momento de salvar, o sistema faz a verificação e constata que há dados inválidos.
4.2 É emitida mensagem informando que os dados são inválidos.
4.3 Retorna para o formulário de avaliação em estado de edição – passo 3 (Linha 3).</td></tr>
<tr><td></td></tr>
</table>

Quadro 4 – Caso de uso manter categorias

A descrição do caso de uso manter itens está apresentada no Quadro 5.

<table>
<tr><td colspan="2">Caso de uso:
Manter itens.
Descrição:
Realização do cadastro de itens em uma categoria do sistema.
Evento Iniciador:
Necessidade de incluir, excluir, consultar ou alterar itens do sistema.
Atores:
Administrador.
Pré-condição:
Existência dos dados essenciais para realizar o cadastro.
Sequência de Eventos:
1. Ator seleciona formulário no qual deseja realizar a operação.
2. Sistema apresenta o formulário.
3. Ator realiza a operação desejada: incluir, excluir, consultar e alterar usuários.
4. Sistema verifica se os dados para a operação estão consistentes e realiza a operação.
Pós-Condição:
Operação de inclusão, exclusão, consulta e alteração de usuários do sistema.</td></tr>
<tr><td>Nome do fluxo alternativo (extensão)</td><td>Descrição</td></tr>
</table>

Linha 4: Dados não são válidos.	4.1 No momento de salvar, o sistema faz a verificação e constata que há dados inválidos. 4.2 É emitida mensagem informando que os dados são inválidos. 4.3 Retorna para o formulário de avaliação em estado de edição – passo 3 (Linha 3).

Quadro 5 – Caso de uso manter itens

O caso de uso "Validar inclusões" é apresentado no Quadro 6.

Caso de uso:
Validar inclusões.
Descrição:
Realização da validação das inclusões feitas pelos usuários.
Evento Iniciador:
Necessidade de incluir itens no sistema.
Atores:
Administrador.
Pré-condição:
Existência dos dados essenciais para realizar o cadastro.
Sequência de Eventos:
1. Ator seleciona formulário no qual deseja realizar a operação.
2. Sistema apresenta o formulário.
3. Ator realiza a operação desejada: incluir, excluir, consultar e alterar usuários.
4. Sistema verifica se os dados para a operação estão consistentes e realiza a operação.
Pós-Condição:
Operação de inclusão, exclusão, consulta e alteração de usuários do sistema.

Nome do fluxo alternativo (extensão)	Descrição
Linha 4: Dados não são válidos.	4.1 No momento de salvar, o sistema faz a verificação e constata que há dados inválidos. 4.2 É emitida mensagem informando que os dados são inválidos. 4.3 Retorna para o formulário de avaliação em estado de edição – passo 3 (Linha 3).

Quadro 6 – Caso de uso validar inclusões

O caso de uso "Consultar categorias", que é realizado pelo usuário é apresentado no Quadro 7.

Caso de uso:
Consultar categorias.
Descrição:
Realização a consulta das categorias cadastradas pelo administrador.
Evento Iniciador: Consultar categorias cadastradas no sistema.
Atores:
Usuário.
Pré-condição:
Existência dos dados essenciais para realizar o cadastro.
Sequência de Eventos:
1. Ator seleciona formulário no qual deseja realizar a operação.

2. Sistema apresenta o formulário. 3. Ator realiza a operação desejada: incluir, excluir, consultar e alterar usuários. 4. Sistema verifica se os dados para a operação estão consistentes e realiza a operação. **Pós-Condição**: Operação de inclusão, exclusão, consulta e alteração de usuários do sistema.

Nome do fluxo alternativo (extensão)	Descrição
Linha 4: Dados não são válidos.	4.1 No momento de salvar, o sistema faz a verificação e constata que há dados inválidos. 4.2 É emitida mensagem informando que os dados são inválidos. 4.3 Retorna para o formulário de avaliação em estado de edição – passo 3 (Linha 3).

Quadro 7 - Caso de uso consultar categorias

O Quadro 8 apresenta o caso de uso "Consultar itens".

Caso de uso: Consultar itens. **Descrição**: Realização a consulta dos itens das categorias cadastradas pelo administrador. **Evento Iniciador**: Interesse em consultar itens cadastrados. **Atores**: Usuário. **Pré-condição**: Existência dos dados essenciais para realizar o cadastro. **Sequência de Eventos**: 1. Ator seleciona formulário no qual deseja realizar a operação. 2. Sistema apresenta o formulário. 3. Ator realiza a operação desejada: incluir, excluir, consultar e alterar usuários. 4. Sistema verifica se os dados para a operação estão consistentes e realiza a operação. **Pós-Condição**: Operação de inclusão, exclusão, consulta e alteração de usuários do sistema.

Nome do fluxo alternativo (extensão)	Descrição
Linha 4: Dados não são válidos.	4.1 No momento de salvar, o sistema faz a verificação e constata que há dados inválidos. 4.2 É emitida mensagem informando que os dados são inválidos. 4.3 Retorna para o formulário de avaliação em estado de edição – passo 3 (Linha 3).

Quadro 8 – Caso de uso consultar itens

A descrição do caso de uso "Sugerir inclusões" é apresentada no Quadro 9.

Caso de uso: Sugerir inclusão. **Descrição**: Sugerir a inclusão de itens em determinada categoria cadastrada pelo administrador. **Evento Iniciador**: Sugestão de inclusão de itens para cadastrar no sistema. **Atores**: Usuário.

Pré-condição: Existência dos dados essenciais para realizar o cadastro. **Sequência de Eventos**: 1. Ator seleciona formulário no qual deseja realizar a operação. 2. Sistema apresenta o formulário. 3. Ator realiza a operação desejada: incluir, excluir, consultar e alterar usuários. 4. Sistema verifica se os dados para a operação estão consistentes e realiza a operação. **Pós-Condição**: Operação de inclusão, exclusão, consulta e alteração de usuários do sistema.

Nome do fluxo alternativo (extensão)	Descrição
Linha 4: Dados não são válidos.	4.1 No momento de salvar, o sistema faz a verificação e constata que há dados inválidos. 4.2 É emitida mensagem informando que os dados são inválidos. 4.3 Retorna para o formulário de avaliação em estado de edição – passo 3 (Linha 3).

Quadro 9 – Caso de uso sugerir inclusões

A Figura 2 apresenta o diagrama de classes.

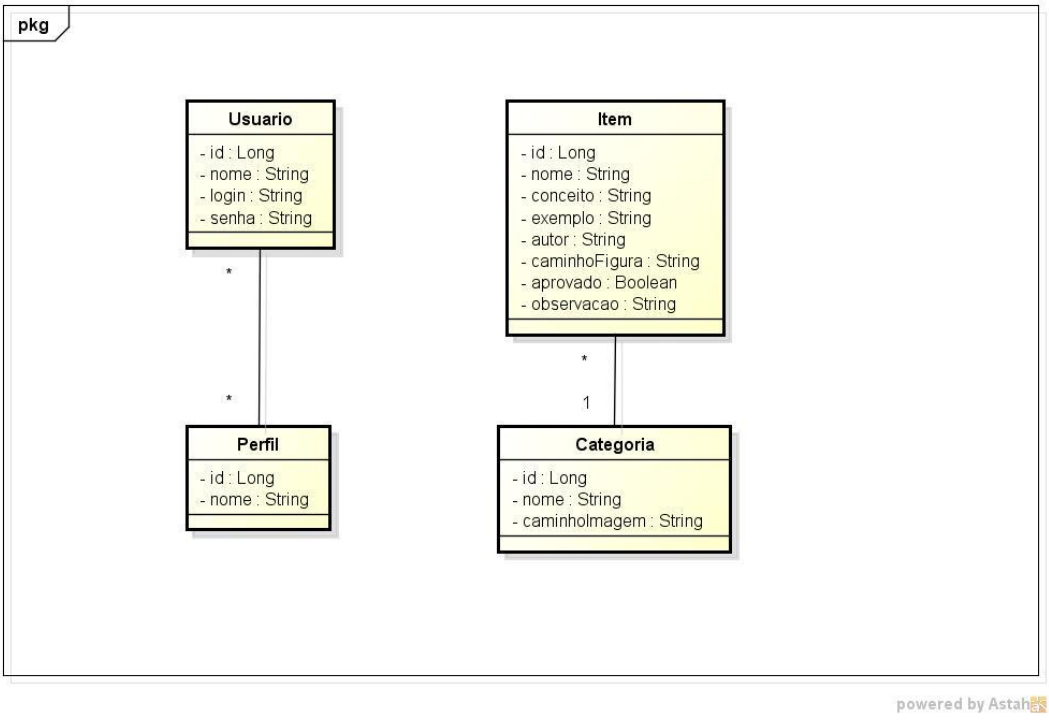

Figura 2 - Diagrama de classes

As classes apresentadas no diagrama da Figura 2 estão documentadas a seguir. Os Quadros 10 a 14 apresentam a descrição das classes de modelo.

Identificação:	Categoria
Descrição:	Principais categorias que serão mostradas no site.
Atributos:	id (número): mapear a classe com a anotação "@Entity" ("@Id") para id da classe para o JPA. nome (string): nome da categoria. descricao (string): breve descrição da categoria.

	caminhoImagem (string): url da imagem selecionada que representará a categoria.

Quadro 10 – Descrição da classe Categoria

Identificação:	Item
Descrição:	Principais itens de cada categoria.
Atributos:	id (número): mapear a classe com a anotação "@Entity" ("@Id") para id da classe para o JPA. nome (string): nome o item. conceito (string): conceito geral sobre o item da categoria. exemplo (string): exemplo de código referente ao item. autor (string): autor do item cadastrado. caminhoFigura (string): url do exemplo a ser mostrado para os usuários. aprovado (*boolean*): aprovação para a publicação.

Quadro 11 – Descrição da classe Item

Identificação:	Usuário
Descrição:	Usuários com acesso ao sistema.
Atributos:	id (número): mapear a classe com a anotação "@Entity" ("@Id") para id da classe para o JPA. nome (string): nome do aluno. *login* (string): identificação de acesso ao sistema. senha (string): para acesso ao sistema.

Quadro 12 – Descrição da classe Usuário

Identificação:	Usuário Perfis
Descrição:	Tipo do perfil do usuário que acessa o sistema.
Atributos:	usuário_id (número): mapear a classe com a anotação "@Entity" ("@Id") para id da classe para o JPA. perfis_id (número): mapear a classe com a anotação "@Entity" ("@Id") para id da classe para o JPA.

Quadro 13 - Descrição da classe usuário perfil

Identificação:	Perfil
Descrição:	Tipo do perfil do usuário que acessa o sistema.
Atributos:	id (número): mapear a classe com a anotação "@Entity" ("@Id") para id da classe para o JPA. nome (string): nome.

Quadro 14 - Descrição da classe perfil

A Figura 3 apresenta o diagrama das entidades e relacionamentos que apresentam o banco de dados da aplicação.

Figura 3 - Diagrama de entidades e relacionamentos

No Quadro 15 estão os campos da tabela de categorias.

Campo	Tipo	Nulo	Chave primária	Chave estrangeira	Observações
id	Long	Não	Sim	Não	
nome	Texto	Não	Não	Não	
descricao	Texto	Não	Não	Não	
caminhoImagem	Texto	Sim	Não	Não	

Quadro 15 - Descrição da tabela categoria

O Quadro 16 apresenta os campos da tabela item.

Campo	Tipo	Nulo	Chave primária	Chave estrangeira	Observações
id	Long	Não	Sim	Não	
nome	Texto	Não	Não	Não	
conceito	Texto	Não	Não	Não	
exemplo	Texto	Não	Não	Não	
autor	Texto	Não	Não	Não	
observacao	Texto	Não	Não	Não	
caminhoFigura	Texto	Não	Não	Não	
aprovado	Boolean	Não	Não	Não	

Quadro 16 - Descrição da tabela item

O Quadro 17 apresenta os campos da tabela usuário.

Campo	Tipo	Nulo	Chave primária	Chave estrangeira	Observações
id	Long	Não	Sim	Não	
nome	Texto	Não	Não	Não	
login	Texto	Não	Não	Não	
senha	Texto	Não	Não	Não	

Quadro 17 - Descrição da tabela usuário

A tabela perfil é apresentada no Quadro 18.

Campo	Tipo	Nulo	Chave primária	Chave estrangeira	Observações
id	Long	Não	Sim	Não	
nome	Texto	Não	Não	Não	

Quadro 18 - Descrição da tabela perfil

4.3 CONFIGURAÇÕES DE AMBIENTE

Este projeto foi desenvolvido com a linguagem de programação Java, usando o Maven, que é uma ferramenta responsável por gerenciar as dependências, controlar versões de artefatos, manter o nível de qualidade do código, entre outras coisas.

O gerenciamento dessas dependências ocorrem em um arquivo chamado pom.xml, Nesse arquivo ficam todas as dependências que serão utilizadas no projeto. Ao configurá-lo, o Maven pesquisa as dependências que o projeto necessita no seu repositório central e realiza o *download* das mesmas para o projeto.

O Spring é um *framework* desenvolvido com a intenção de simplificar a programação de aplicações em linguagem Java, baseando-se no padrão de inversão de controle e injeção de dependências, ou seja, ele instancia objetos apenas quando for necessário. Esse *framework* possui diversos módulos, que são utilizados conforme a necessidade do projeto. O SpringBoot que foi utilizado nesse trabalho para envolver todos os módulos do Spring, reaproveitando, assim, os módulos existentes e aumentando a produtividade.

Uma forma fácil de iniciar o projeto foi gerando-o atrás do site do próprio Spring. A Figura 4 mostra como criar um projeto do tipo Maven, usando o SpringBoot e ao lado direito, no campo “Dependencies” como facilmente adicionar as dependências, que irão variar, de acordo com o que o projeto irá utilizar.

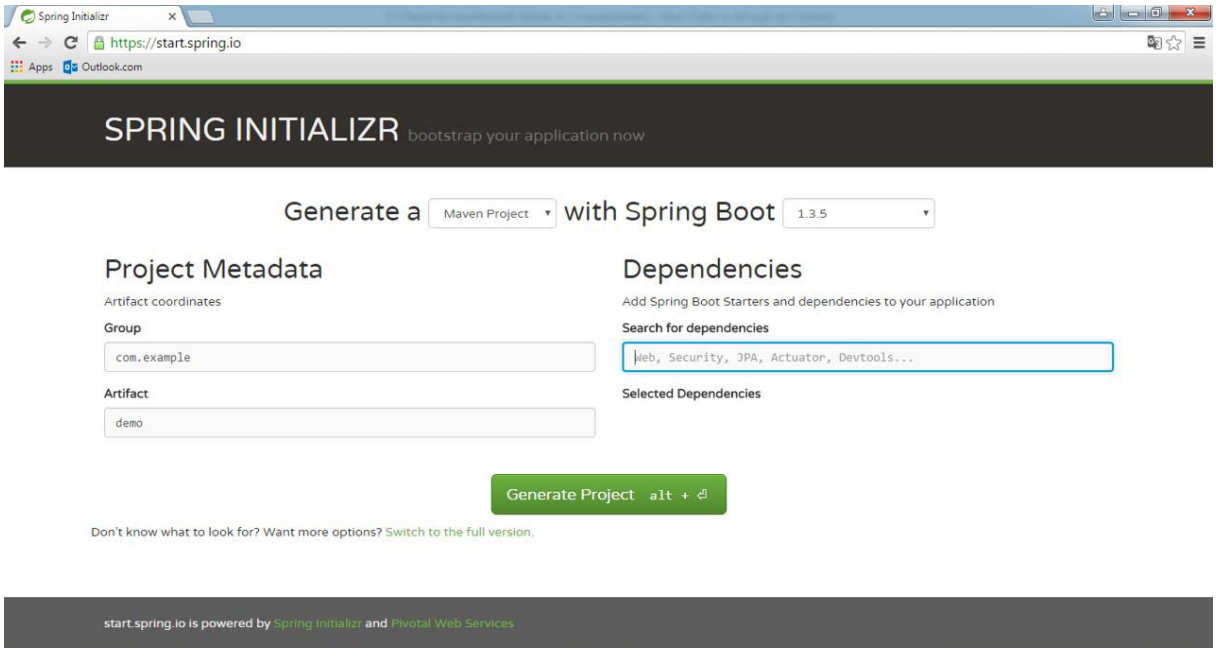

Figura 4 – Configurações para criar um projeto Maven

Após gerar o projeto com as dependências desejadas, foi necessário importá-lo no ambiente de desenvolvimento.

4.4 APRESENTAÇÃO DO SISTEMA

O leiaute do sistema é composto por seções superior e central. A seção superior contém o logo e os *links* de *home* e *login* e a seção central contém as imagens das categorias cadastradas pelo administrador. As seções são estáticas responsivas, ou seja, a visualização se adapta conforme o dispositivo utilizado no acesso do sistema (celular, *tablet*, *notebook*).

A Figura 5 ilustra a estruturação da página de acordo com cada seção.

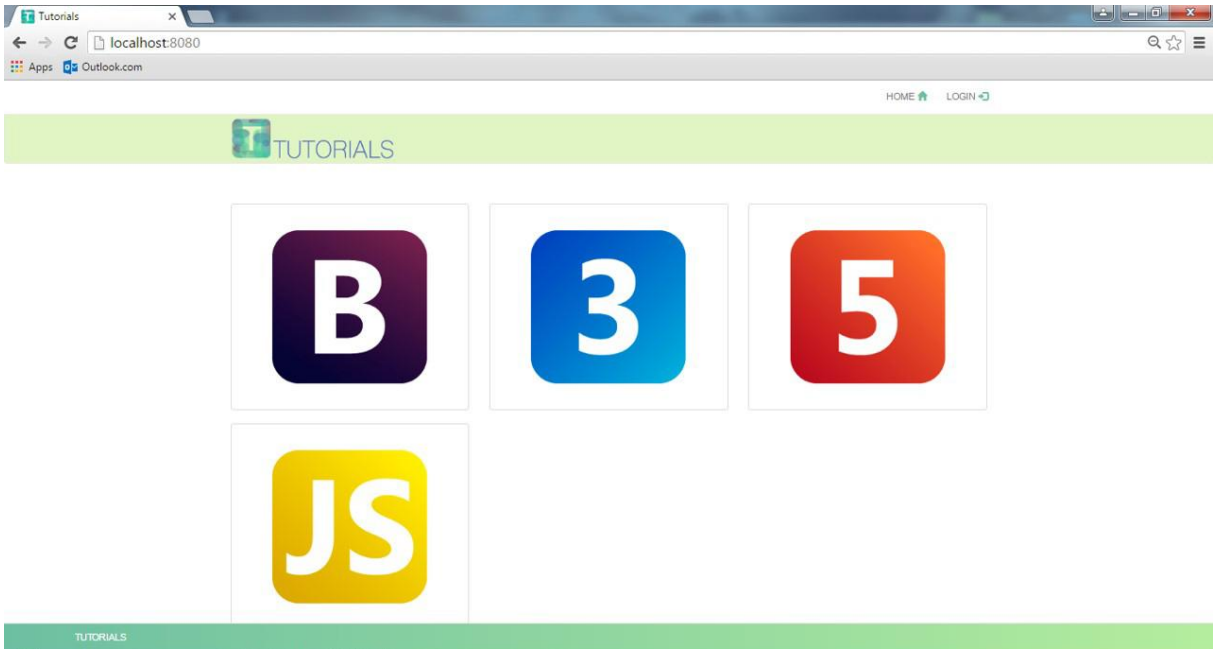

Figura 5 – Página inicial

Ao clicar em qualquer categoria o usuário é direcionado para a página interna que contém a seção superior e as seções que apresenta a lateral esquerda que permite o acesso das categorias e itens e a seção central que apresenta o conceito, código e *view* do item selecionado em uma determinada categoria. A Figura 6 ilustra a tela de categorias, itens e conteúdo.

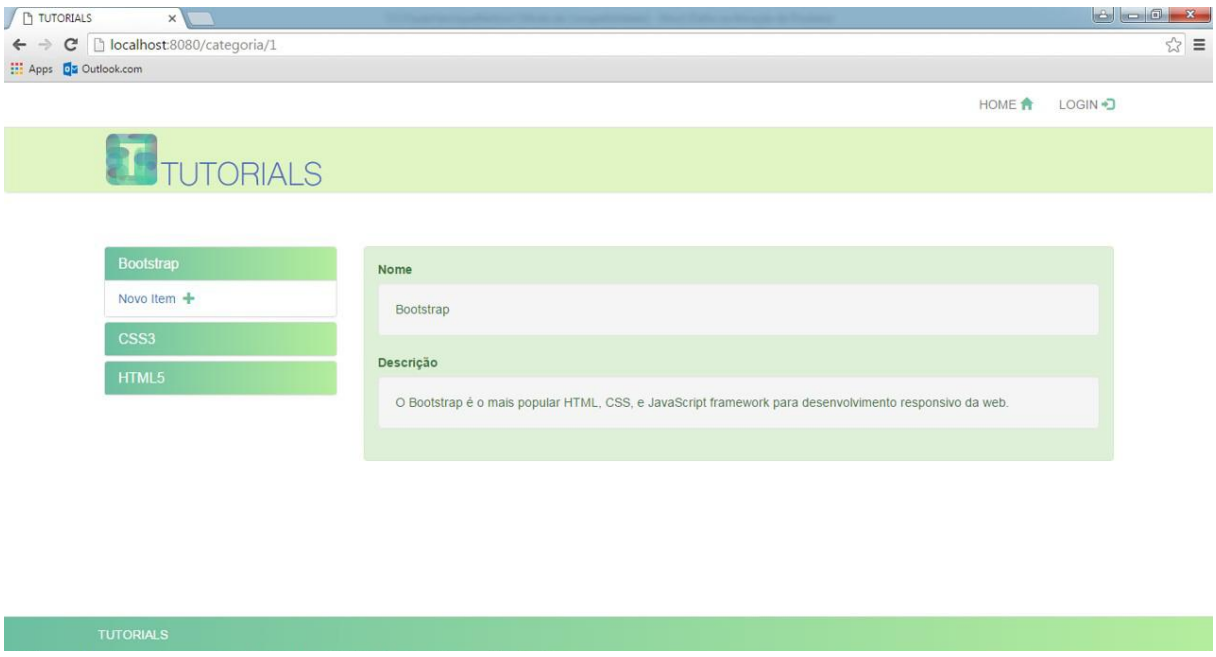

Figura 6 – Tela de categorias, itens e conteúdos

Ao clicar em qualquer item das categorias são apresentadas as informações que cada item possui, como, por exemplo, nome, conceito do item, exemplo de código e visualização gráfica do exemplo. A Figura 7 apresenta a tela com as informações cadastradas em um item.

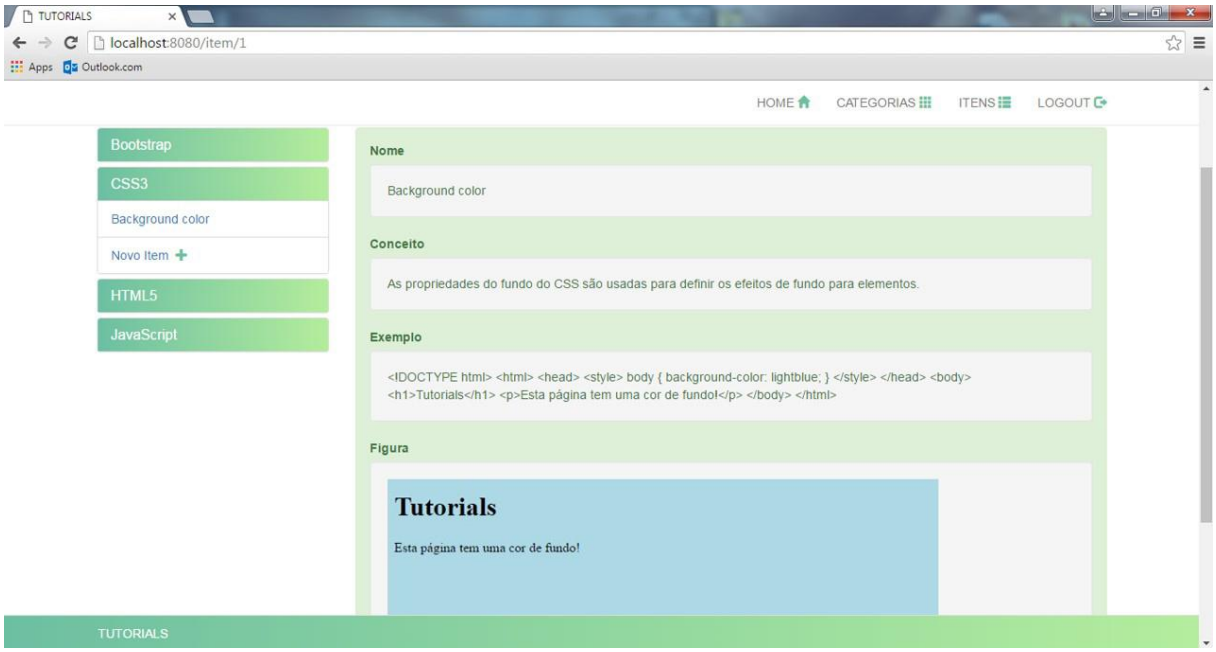

Figura 7 - Itens da categoria

A Figura 8 mostra a tela de *login* do sistema, contendo um alerta, informando que a área é restrita ao administrador do sistema.

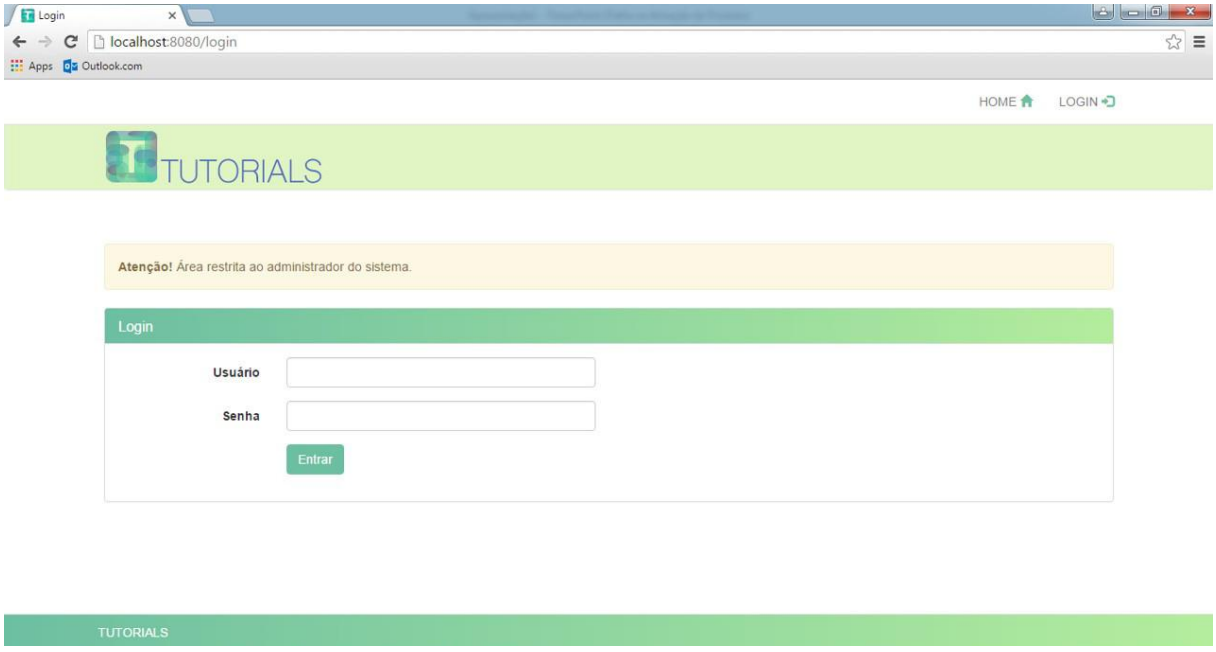

Figura 8 - Tela de login

Após realizar o *login* no sistema, o administrador tem acesso à tela que apresenta as aprovações pendentes que são as sugestões ou itens dos usuários que não necessitam efetuar *login*. No entanto, essas sugestões serão aprovadas pelo administrador do sistema para serem exibidas. A Figura 9 mostra a tela com aprovações pendentes para serem validadas pelo administrador.

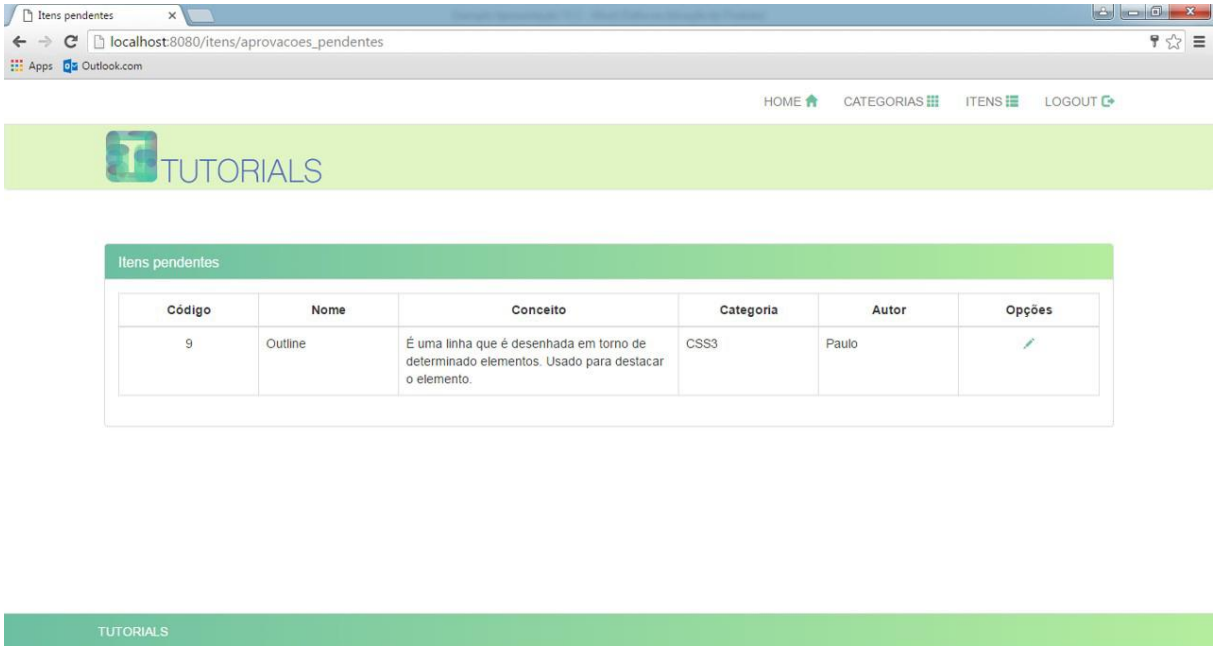

Figura 9 - Tela com itens pendentes

Ao clicar no *link* categoria, uma lista de categorias será exibida e um *link* chamado "Nova categoria" estará disponível no canto superior direito da seção central para que o usuário possa registrar uma nova categoria. Nessa tela, também é apresentada uma opção para filtrar essas categorias, conforme ilustra a Figura 10. As mesmas opções estarão disponíveis se o administrador clicar no *link* itens.

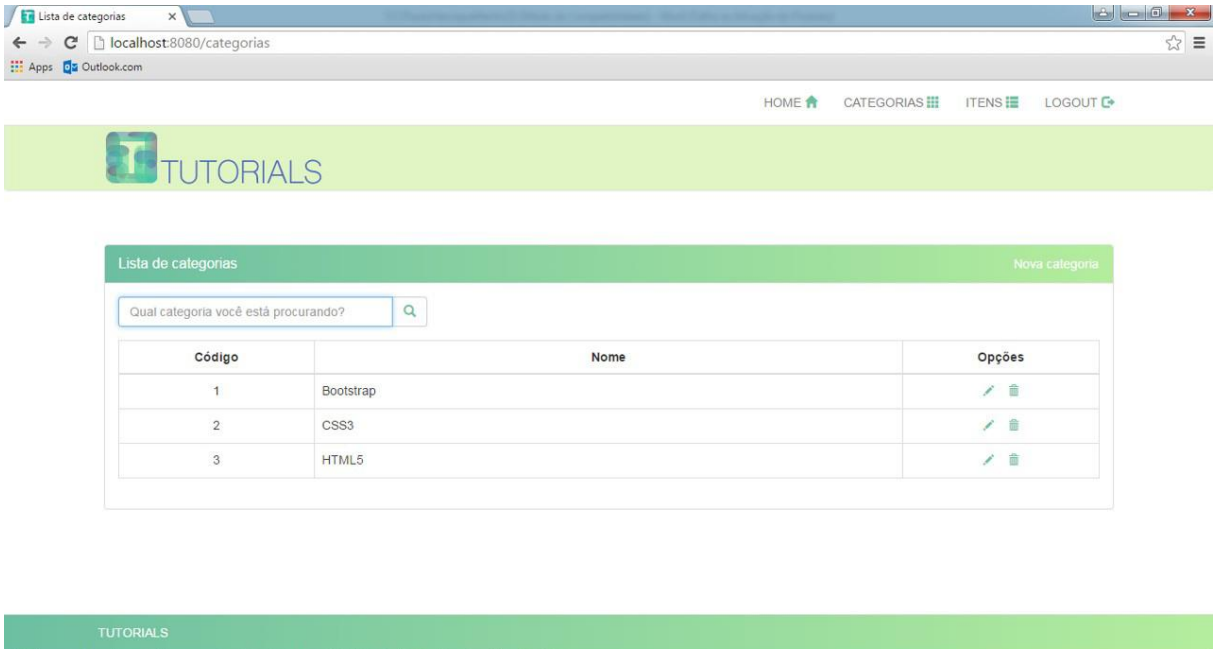

Figura 10 - Lista de categorias

Ao clicar na opção “Nova categoria” ou “Novo item”, um formulário será exibido para que o usuário possa cadastrar uma nova categoria ou item. Se o administrador clicar na opção “Salvar” sem informar os campos obrigatórios, o sistema fará uma validação e apresentará uma mensagem que informa que determinados campos são necessários para o registro ser efetuado corretamente. A Figura 11 exemplifica a inserção de uma nova categoria com campos validados.

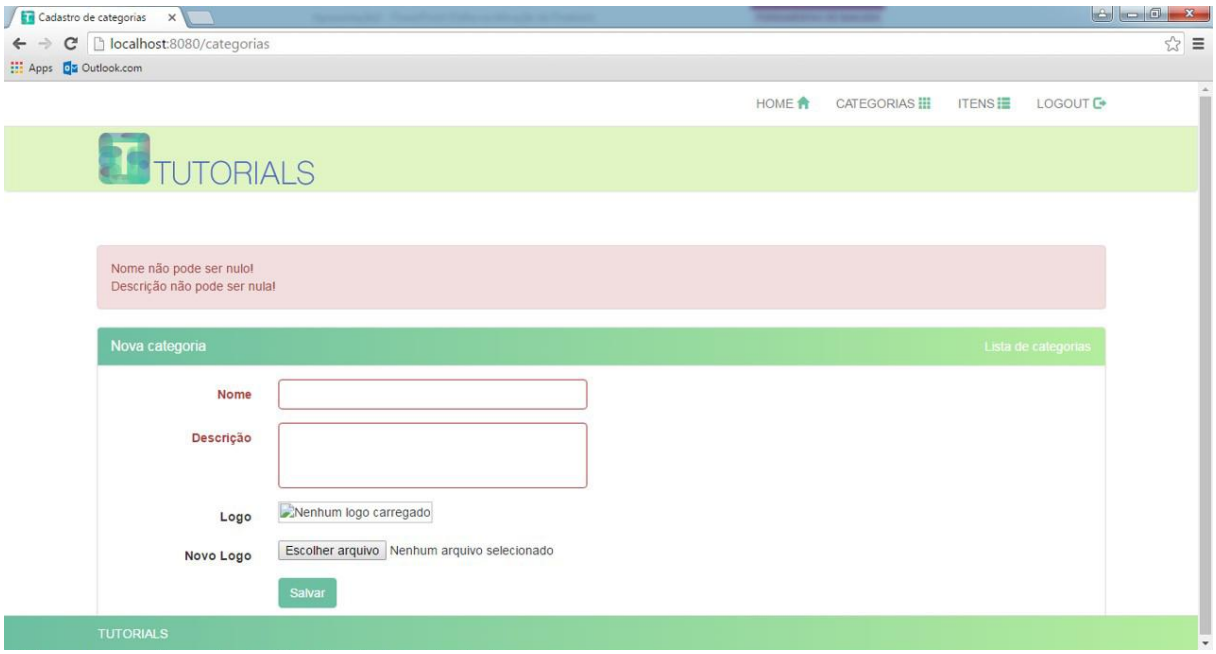

Figura 11 - Exemplo de validação de campos

Se os campos informados forem válidos, o sistema gravará a categoria ou item e exibirá uma mensagem informando que o registro foi efetuado com sucesso e, também, limpará os campos do formulário para permitir a inclusão de novas categorias ou itens, conforme ilustra a Figura 12.

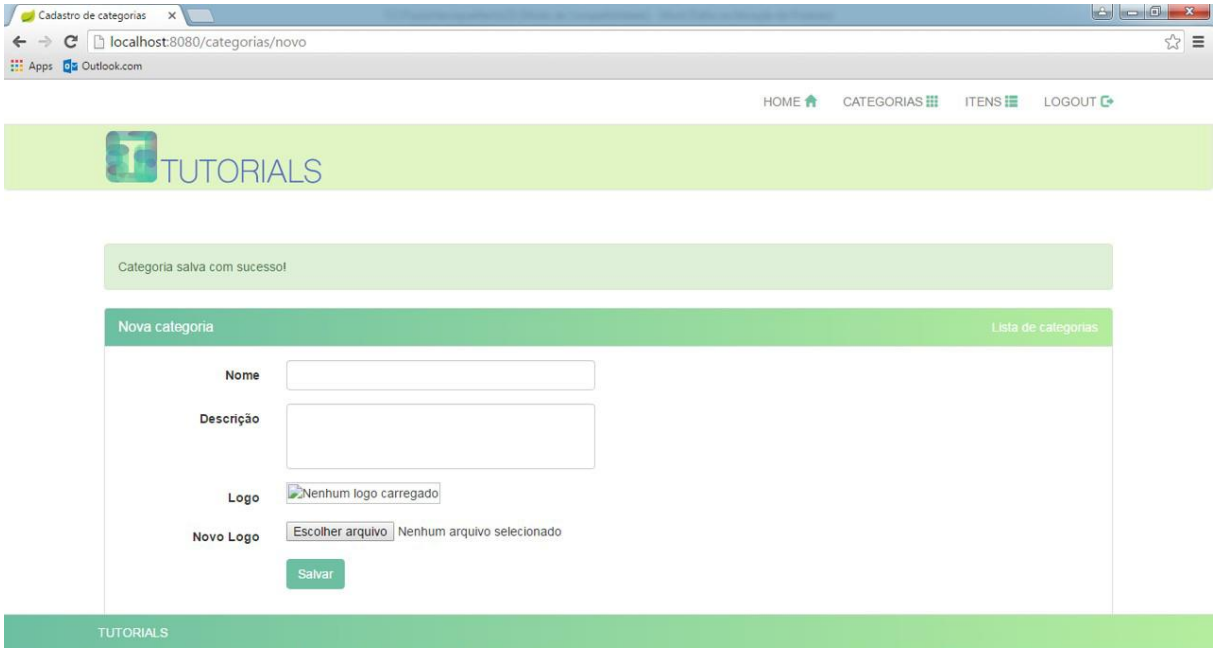

Figura 12 – Exemplo de validação ao salvar nova categoria

Na Figura 13, é possível visualizar a operação de edição, que é semelhante à de inclusão, sendo que para realizar a mesma, o administrador precisa clicar no *link* com o ícone representado graficamente por um lápis, na coluna de opções (Figura 10). Desta forma, o sistema realizará o mesmo procedimento utilizado na inclusão. No entanto apresentará formulário preenchido com os dados da categoria ou item selecionado. O processo de validação também exige que os campos obrigatórios sejam preenchidos corretamente.

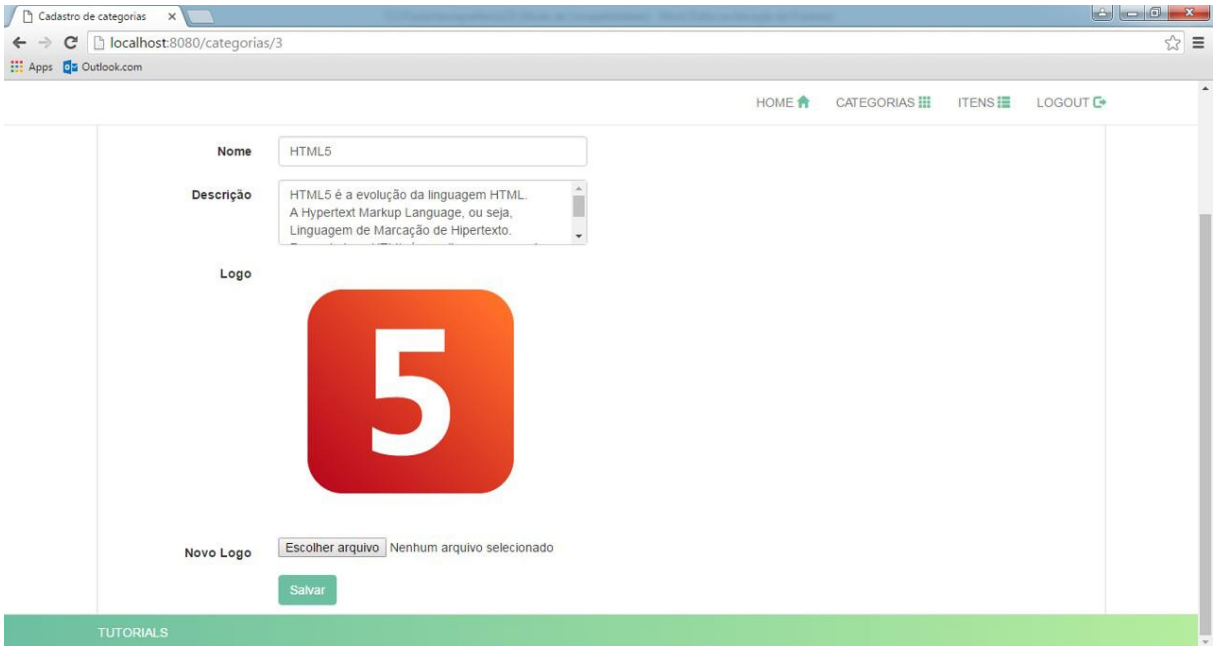

Figura 13 – Formulário para edição

Para a operação de exclusão, o usuário deve clicar no ícone representado graficamente por uma lixeira, localizado na coluna de opções (Figura 10), referente à categoria ou item de que deseja excluir. O sistema apresentará uma mensagem de confirmação, informando ao usuário que determinada categoria ou item será excluído, conforme mostra a Figura 14.

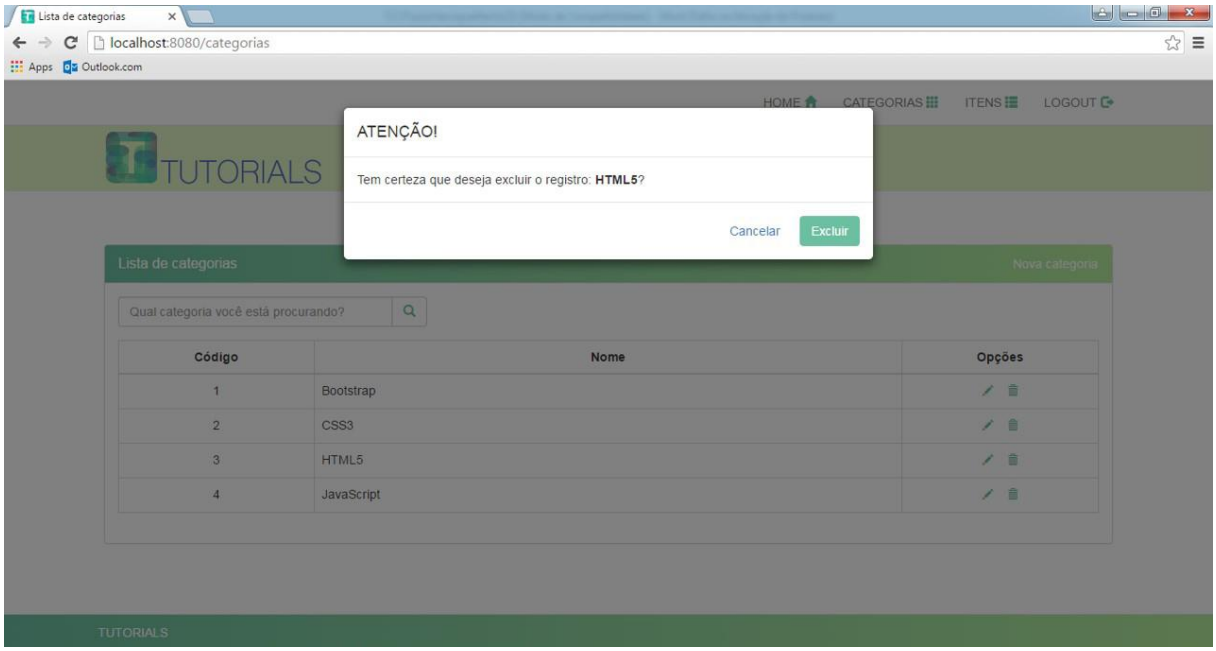

Figura 14 – Mensagem de exclusão

Se o administrador confirmar a exclusão de um registro, o sistema removerá a categoria ou item desejado, atualizará o formulário correspondente e apresentará uma mensagem que o registro foi removido com sucesso, como mostra a Figura 15.

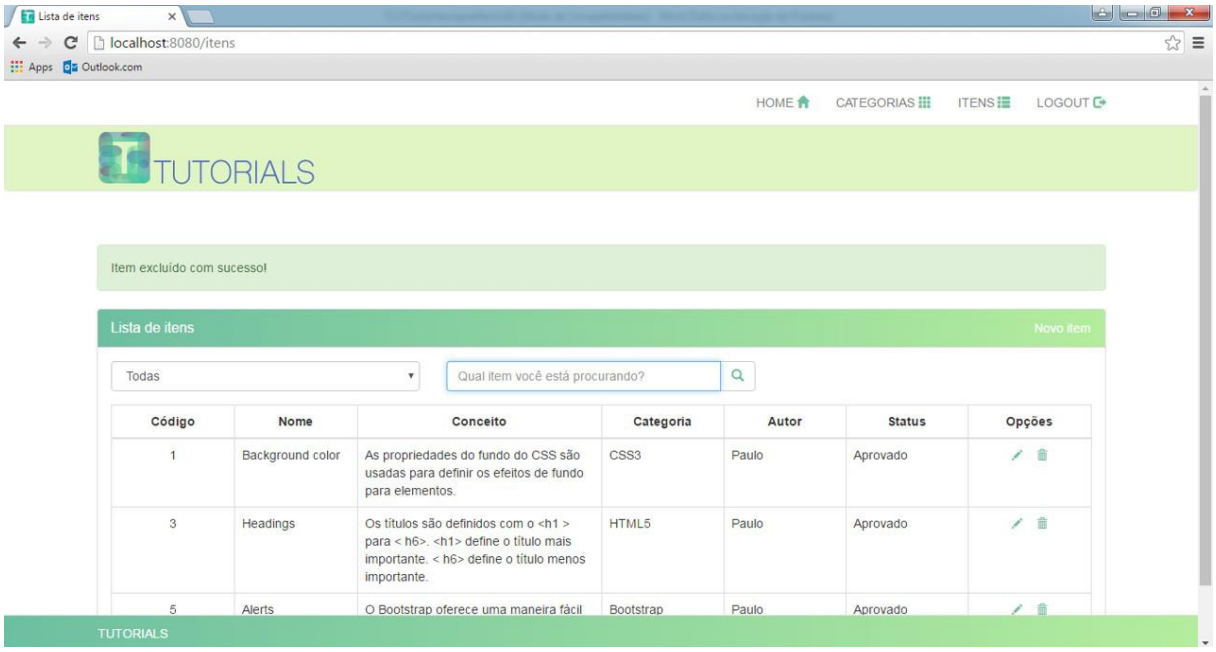

Figura 15 – Confirmação da exclusão

A Figura 16 mostra como o sistema se comporta (responsividade) quando acessado em um *tablet*.

Figura 16 – Tela inicial da aplicação visualizada em um *tablet*

A Figura 17 mostra como é a tela de apresentação de item acessado por uma determinada categoria, quando visualizado em um *tablet*.

Figura 17 – Tela de categorias e itens visualizada em um *tablet*

A responsividade é, também, observada caso o sistema seja acessado por meio de um *Smartphone* (Figura 18).

Figura 18 - Tela inicial da aplicação visualizada em um smartphone

A Figura 19 mostra como é a tela de apresentação de item acessado por uma determinada categoria, quando visualizado em um *smartphone*.

Figura 19 - Tela de categorias e itens visualizada em um smartphone

Assim, é possível observar que a visualização depende da tela que está sendo acessada. No entanto, como o sistema foi desenvolvido com recursos de responsividade fornecidos pelo Bootstrap, a visualização não fica comprometida quando acessada por diferentes dispositivos e, assim, o usuário pode acessar todos os recursos do sistema sem comprometer, também, a interatividade.

4.5 IMPLEMENTAÇÃO DO SISTEMA

Para facilitar o desenvolvimento e minimizar a repetição de códigos, utilizou-se um leiaute padrão, chamado “LayoutPadrao.html” (Listagem 1). Essa página contém o leiaute básico do cabeçalho do sistema, que será utilizado por todas as demais páginas.

```
<!DOCTYPE html>
<html xmlns="http://www.w3.org/1999/xhtml"
      xmlns:th="http://www.thymeleaf.org"
      xmlns:layout="http://www.ultraq.net.nz/thymeleaf/layout">

<head>
      <meta charset="UTF-8"/>
      <title>TUTORIALS</title>
      <link rel="stylesheet" type="text/css" href="/css/bootstrap.css"/>
```

```
      <link rel="stylesheet" type="text/css" href="/css/style.css"/>
      <link rel="icon" href="images/favicon.ico"/>
</head>
<body>
      <div class="navbar navbar-default navbar-fixed-top" role="navigation">
      <div class="container">

            <div class="navbar-header">

            <button type="button" class="navbar-toggle" data-toggle="collapse"
data-target=".navbar-collapse">
            <span class="sr-only">Toggle navigation</span>
            <span class="icon-bar"></span>
            <span class="icon-bar"></span>
            <span class="icon-bar"></span>
            </button>
            </div>

      <div class="collapse navbar-collapse navHeaderCollapse">

            <ul class="nav navbar-nav navbar-right">
                        <li>
                              <a href="/">HOME
                        <span class="glyphicon glyphicon-home"></span></a>
                        </li>
                        <li sec:authorize="hasRole('ROLE_USER')">
                              <a href="/categorias">CATEGORIAS
                        <span class="glyphicon glyphicon-th"></span></a>
                        </li>
                        <li sec:authorize="hasRole('ROLE_USER')">
                              <a href="/itens">ITENS
                        <span class="glyphicon glyphicon-th-list"></span></a>
                        </li>
                        <li sec:authorize="isAnonymous()">
                              <a href="/login">LOGIN
                        <span class="glyphicon glyphicon-log-in"></span></a>
                        </li>
                        <li sec:authorize="isAuthenticated()">
                              <a href="/logout">LOGOUT
                        <span class="glyphicon glyphicon-log-out"></span></a>
                        </li>
                  </ul>
            </div>
      </div>
</div>

<nav class="navbar navbar-default navbar-lower" role="navigation">
        <div class="container">
              <div class="col-md-12">
                <a class="navbar-brand"  href="/"> <img class="img-responsive"
                    alt="Tutorials" src="/images/tutorials.png" /></a>
              </div>
        </div>
</nav>

    <section layout:fragment="conteudo">
    <p>Conteúdo principal</p>
    </section>
```

```
    <div id="footerPai">
          <div class="panel-footer">TUTORIALS</div>
    </div>

      <script src="/js/jquery-2.2.3.min.js"></script>
      <script src="/js/bootstrap.js"></script>
      <script src="/js/confirma-exclusao.js"></script>
</body>
</html>
```

Listagem 1 – *Template* do leiaute padrão

A Listagem 1 apresenta o leiaute padrão de todas as páginas do sistema. No código é apresentado também o uso Bootstrap, um *framework* utilizado para desenvolver aplicações responsivas. A classe *navbar-toggle* "comprime" as opções do menu, caso o sistema seja utilizado ou acessado em dispositivos pequenos, como, por exemplo, celulares ou *tablets*. A seção "conteudo" contém as outras partes das páginas do sistema.

O menu principal do sistema foi desenvolvido em uma página chamada "index.html" (Listagem 2). Nela é possível observar a utilização da página "LayoutPadrao.html" exibida na Listagem 1. Para utilizá-la, é necessário acrescentar na *tag* html o seguinte código: layout:decorator="LayoutPadrao", e determinar qual será o conteúdo da página na seção "conteudo".

```
<!DOCTYPE html>
<html xmlns="http://www.w3.org/1999/xhtml"
      xmlns:th="http://www.thymeleaf.org"
      xmlns:layout="http://www.ultraq.net.nz/thymeleaf/layout"
      layout:decorator="LayoutPadrao">
<head>
      <title>Tutorials</title>
</head>
<section layout:fragment="conteudo">

<div class="container">
      <div class="row">
            <div class="col-xs-6 col-md-4" th:each="categoria: ${categorias}">
            <a th:href="@{/categoria/{id}(id=${categoria.id})}"
class="thumbnail">
                  <img th:src="@{/categorias/logo/{id}(id=${categoria.id})}"
                  th:alt="${categoria.nome}" th:title="${categoria.nome}" />
                  </a>
            </div>
      </div>
</div>
</section>
</html>
```

Listagem 2 – Template do index

Na página “index.html” foi utilizado a classe *thumbnail* do Bootstrap quetem por objetivo mostrar imagens, vídeos ou textos com o mínimo de marcações necessárias. Também foram utilizadas algumas expressões do *Thymeleaf*, caracterizadas e observadas pela marcação “th”.

A Listagem 3 mostra como a classe Categoria foi desenvolvida. A anotação @Entity usa o nome da classe Java para associar com a tabela no banco de dados, por padrão todos os atributos da classe representam colunas da tabela.

```
package br.edu.utfpr.tutorials.model;
import java.io.Serializable;
import java.util.List;
import javax.persistence.Column;
import javax.persistence.Entity;
import javax.persistence.GeneratedValue;
import javax.persistence.Id;
import javax.persistence.OneToMany;
import javax.validation.constraints.NotNull;
import org.hibernate.validator.constraints.NotEmpty;

@Entity
public class Categoria implements Serializable{
	private static final long serialVersionUID = -2611161425737626530L;
	@Id
	@GeneratedValue
	private Long id;
	@Column(length=60,nullable=false)
	@NotEmpty(message="Nome não pode ser nulo!")
	@NotNull(message="Nome não pode ser nulo!")
	private String nome;
	@Column(nullable=false)
	@NotEmpty(message="Descrição não pode ser nula!")
	@NotNull(message="Descrição não pode ser nula!")
	private String descricao;
	private String caminhoImagem;
	@OneToMany(mappedBy="categoria")
	private List<Item> itens;

	//Método construtor
	public Categoria (){
	}
	public Long getId() {
		return id;
	}
	public void setId(Long id) {
		this.id = id;
	}
	public String getNome() {
		return nome;
	}
	public void setNome(String nome) {
		this.nome = nome;
	}
	public String getDescricao() {
```

```
		return descricao;
	}
	public void setDescricao(String descricao) {
		this.descricao = descricao;
	}
	public String getCaminhoImagem() {
		return caminhoImagem;
	}
	public void setCaminhoImagem(String caminhoImagem) {
		this.caminhoImagem = caminhoImagem;
	}
	public List<Item> getItens() {
		return itens;
	}
	public void setItens(List<Item> itens) {
		this.itens = itens;
	}
}
```

Listagem 3 – Classe categoria

Na Listagem 4 é possível observar como a classe Item foi desenvolvida. Assim como na Listagem 3, essa classe também recebeu a anotação @Entity. Em ambas as classes, categoria ou itens, há atributos que não podem se vazios ou nulos, por isso, foram utilizadas as anotações @NotEmpty e @NotNull, respectivamente.

```
package br.edu.utfpr.tutorials.model;
import java.io.Serializable;
import javax.persistence.Column;
import javax.persistence.Entity;
import javax.persistence.GeneratedValue;
import javax.persistence.Id;
import javax.persistence.ManyToOne;
import javax.validation.constraints.NotNull;
import org.hibernate.validator.constraints.NotEmpty;

@Entity
public class Item implements Serializable{
	private static final long serialVersionUID = 3982141942216451555L;
	@Id
	@GeneratedValue
	private Long id;

	@Column(length=60,nullable=false) //customiza os valores do campo no banco
de dados
	@NotEmpty(message="Nome não pode ser nulo!")
	@NotNull(message="Nome não pode ser nulo!")
	private String nome;

	@NotEmpty(message="O conceito não pode ser nulo!")
	@Column(nullable=false)
	private String conceito;

	@NotEmpty(message="O exemplo não pode ser nulo!")
	@Column(nullable=false)
	private String exemplo;
```

```
	private String autor;

	private String caminhoFigura;
	private Boolean aprovado;

	private String observacao;

	//relacionamento de muitos para um, muitos itens para uma categoria.
	//optinal=false não salva sem ter uma categoria.
	@ManyToOne(optional=false)
	private Categoria categoria;

	public Item(){
	}
	public Long getId() {
		return id;
	}
	public void setId(Long id) {
		this.id = id;
	}
	public String getNome() {
		return nome;
	}
	public void setNome(String nome) {
		this.nome = nome;
	}
	public String getConceito() {
		return conceito;
	}
	public void setConceito(String conceito) {
		this.conceito = conceito;
	}
	public String getExemplo() {
		return exemplo;
	}
	public void setExemplo(String exemplo) {
		this.exemplo = exemplo;
	}
	public String getCaminhoFigura() {
		return caminhoFigura;
	}
	public void setCaminhoFigura(String caminhoFigura) {
		this.caminhoFigura = caminhoFigura;
	}
	public Boolean getAprovado() {
		return aprovado;
	}
	public void setAprovado(Boolean aprovado) {
		this.aprovado = aprovado;
	}
	public Categoria getCategoria() {
		return categoria;
	}
	public void setCategoria(Categoria categoria) {
		this.categoria = categoria;
	}
	public String getAutor() {
		return autor;
```

```
	}
	public void setAutor(String autor) {
		this.autor = autor;
	}
	public String getObservacao() {
		return observacao;
	}
	public void setObservacao(String observacao) {
		this.observacao = observacao;
	}
}
```

Listagem 4 – Classe item

A Listagem 5 mostra o código CategoriaController que é responsável por fazer o controle de todo o fluxo de informações do sistema, executar as regras de negócio e repassar as informações para a visualização. É essa camada que estabelece "se", "o que", "quando" e "onde" deve ocorrer alguma ação. Também é possível definir quais informações devem ser geradas quais regras devem ser acionadas e para onde as informações processadas devem ser direcionadas.

```
package br.edu.utfpr.tutorials.controller;
import java.io.File;
@Controller
@RequestMapping("/categorias")
public class CategoriaController {

	private static final String CADASTRO_VIEW = "CadastroCategoria";
	@Autowired
	private CategoriaRepository categoriaRepository;

	@RequestMapping("/novo")
	public ModelAndView novo(){
		ModelAndView mv = new ModelAndView(CADASTRO_VIEW);
		mv.addObject(new Categoria());
		return mv;
	}

	@RequestMapping(method=RequestMethod.POST)
	public String salvar(@Validated Categoria categoria, Errors errors,
RedirectAttributes attributes,
			@RequestParam("logo") MultipartFile logo) {
		if(errors.hasErrors()){
			return CADASTRO_VIEW;
		}
		// salvar categoria para obter o ID
		categoriaRepository.save(categoria);

		// verifica se arquivo nao esta vazio
		if (!logo.isEmpty()) {
			String imgFolderCategorias = TutorialsApplication.IMAGE_FILES +
"/categorias";
			FileUtils.criarPastaSeNaoExistir(imgFolderCategorias);
```

```
                    String caminhoImagem = imgFolderCategorias + "/" +
categoria.getId() + ".jpg";
                    try {
                          // cria arquivo a ser gravado
                          File file = new File(caminhoImagem);
                          // grava o arquivo
                          FileUtils.gravarArquivo(logo, file);
                    } catch (Exception e) {
                          attributes.addFlashAttribute("message", e.getMessage());
                    }
                    // categoria obtem o caminho da imagem salva
                    categoria.setCaminhoImagem(caminhoImagem);
                    // atualiza os dados da categoria
                    categoriaRepository.save(categoria);
              } else {
                    attributes.addFlashAttribute("message", "Selecione o arquivo");
              }
              attributes.addFlashAttribute("mensagem",    "Categoria    salva    com
sucesso!");
              return "redirect:/categorias/novo";
       }

       @RequestMapping
       public ModelAndView pesquisar(@RequestParam(defaultValue = "%")String nome){

       List<Categoria>todasCategorias=categoriaRepository.findByNomeContaining(nome
);
              ModelAndView mv = new ModelAndView("PesquisaCategorias");
              mv.addObject("categorias", todasCategorias);
              mv.addObject("urlBase", "/categorias");
              if (!"%".equals(nome)) {
                    mv.addObject("nome", nome);
              }
              return mv;
       }

       @RequestMapping("{id}")
       public ModelAndView editar(@PathVariable("id")Categoria categoria){

              ModelAndView mv = new ModelAndView(CADASTRO_VIEW);
              mv.addObject(categoria);
              return mv;
       }

       @RequestMapping(value="{id}", method = RequestMethod.DELETE)
       public String excluir(@PathVariable Long id, RedirectAttributes attributes){
              categoriaRepository.delete(id);
              attributes.addFlashAttribute("mensagem",   "Categoria   excluída   com
sucesso!");
              return "redirect:/categorias";
       }

       @RequestMapping("/logo/{id}")
       // utilizamos o response body para nao retornar uma pagina html, mas sim um
array de bytes
       @ResponseBody public byte[] getLogo(@PathVariable Long id) {
              // carrega os dados da categoria
              Categoria categoria = categoriaRepository.findOne(id);
              // cria array de bytes
```

```
            byte[] fileBytes = new byte[0];
            // verifica se a categoria possui uma imagem
            if (categoria.getCaminhoImagem() != null &&
                        !"".equals(categoria.getCaminhoImagem().trim())) {
                  // le o arquivo
                  File logo = new File(categoria.getCaminhoImagem());
            try {
                  // transforma o arquivo em array de bytes
                        fileBytes = Files.readAllBytes(logo.toPath());
                  } catch (IOException e) {
                        e.printStackTrace();
                  }
            }
        return fileBytes;
      }
}
```

Listagem 5 – CategoriaController

Nos métodos do controlador (apresentados na Listagem 5), foi utilizado a classe ModelAndView do Spring, que é um *framework* para suporte de injeção de dependências e gerenciamento de transações. Essa classe gerencia as camadas *Model* e *View* para tornar possível ao controlador retornar um *model* e uma *view* em um único valor de retorno.

O ItemController tem características semelhantes ao CategoriaController. Na Listagem 6 é possível observar como o mesmo foi desenvolvido.

```
package br.edu.utfpr.tutorials.controller;
import java.io.File;
import java.io.IOException;
import java.nio.file.Files;
import java.util.List;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.security.core.annotation.AuthenticationPrincipal;
import org.springframework.stereotype.Controller;
import org.springframework.validation.Errors;
import org.springframework.validation.annotation.Validated;
import org.springframework.web.bind.annotation.PathVariable;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RequestMethod;
import org.springframework.web.bind.annotation.RequestParam;
import org.springframework.web.bind.annotation.ResponseBody;
import org.springframework.web.multipart.MultipartFile;
import org.springframework.web.servlet.ModelAndView;
import org.springframework.web.servlet.mvc.support.RedirectAttributes;

import br.edu.utfpr.tutorials.TutorialsApplication;
import br.edu.utfpr.tutorials.model.Categoria;
import br.edu.utfpr.tutorials.model.Item;
import br.edu.utfpr.tutorials.model.Usuario;
import br.edu.utfpr.tutorials.repository.CategoriaRepository;
import br.edu.utfpr.tutorials.repository.ItemRepository;
import br.edu.utfpr.tutorials.utils.FileUtils;
```

```
@Controller
@RequestMapping("/itens")
public class ItemController {

private static final String CADASTRO_VIEW = "CadastroItem";

	@Autowired
	private ItemRepository itemRepository;
	@Autowired
	private CategoriaRepository categoriaRepository;

	@RequestMapping("/novo")
	public ModelAndView novo(@RequestParam(name="categoriaId",required=false)
Long categoriaId){
		ModelAndView mv = new ModelAndView(CADASTRO_VIEW);
		Item item = new Item();
		if (categoriaId != null) {
			Categoria categoria = categoriaRepository.findOne(categoriaId);
			if (categoria != null) {
				item.setCategoria(categoria);
			}
		}
		mv.addObject(item);
		carregarCategorias(mv);
		return mv;
	}

	@RequestMapping(method=RequestMethod.POST)
	public ModelAndView salvar(@Validated Item item, Errors errors,
RedirectAttributes attributes,
			@AuthenticationPrincipal Usuario usuario,
@RequestParam("figura") MultipartFile figura) {

		if(errors.hasErrors()){
			//return CADASTRO_VIEW;
			return editar(item);
		}

		// salva o item para obter o ID
		itemRepository.save(item);

		// verifica se arquivo nao esta vazio
		if (!figura.isEmpty()) {
			String imgFolderItens = TutorialsApplication.IMAGE_FILES +
"/itens";
			FileUtils.criarPastaSeNaoExistir(imgFolderItens);
			String caminhoFigura = imgFolderItens + "/" + item.getId() +
".jpg";
			try {
				// cria arquivo a ser gravado
				File file = new File(caminhoFigura);
				// grava o arquivo
				FileUtils.gravarArquivo(figura, file);
			} catch (Exception e) {
				attributes.addFlashAttribute("mensagem",
e.getMessage());
			}
			// item obtem o caminho da imagem salva
			item.setCaminhoFigura(caminhoFigura);
```

```
                  // atualiza os dados do item
                  itemRepository.save(item);
            } else {
                  attributes.addFlashAttribute("mensagem",      "Selecione      o
arquivo");
            }
            if (usuario == null) { // nao esta logado
                  attributes.addFlashAttribute("mensagem", "O Item foi salvo, mas
aguarda aprovação do administrador");
            } else { // esta logado
                  if (item.getAprovado() == null) {
                        item.setAprovado(false);
                  }
                  attributes.addFlashAttribute("mensagem",   "Item   salvo   com
sucesso!");
            }
            ModelAndView mv = new ModelAndView("redirect:/itens/novo");
            return mv;
      }

      @RequestMapping
      public ModelAndView pesquisar(@RequestParam(defaultValue = "%")String nome,
                  @RequestParam(required=false) Long categoriaId){
            List<Item> todosItens = null;
            if (categoriaId == null || categoriaId <= 0) {
                  todosItens=itemRepository.findByNomeContaining(nome);
            } else {

      todosItens=itemRepository.findByNomeContainingAndCategoriaId(nome,
categoriaId);
            }

            ModelAndView mv = new ModelAndView("PesquisaItens");
            mv.addObject("itens", todosItens);
            mv.addObject("urlBase", "/itens");
            carregarCategorias(mv);
            if (!"%".equals(nome)) {
                  mv.addObject("nome", nome);
            }
            return mv;
      }

      @RequestMapping("{id}")
      public ModelAndView editar(@PathVariable("id") Item item){
            ModelAndView mv = new ModelAndView(CADASTRO_VIEW);
            mv.addObject(item);
            carregarCategorias(mv);
            return mv;
      }

      private void carregarCategorias(ModelAndView mv) {
            List<Categoria> categorias = categoriaRepository.findAll();
            mv.addObject("categorias", categorias);
      }

      @RequestMapping(value="{id}", method = RequestMethod.DELETE)
      public String excluir(@PathVariable Long id, RedirectAttributes attributes){
            itemRepository.delete(id);
```

```
            attributes.addFlashAttribute("mensagem", "Item excluído com
sucesso!");
            return "redirect:/itens";
      }

      @RequestMapping("/figura/{id}")
      // utilizamos o response body para nao retornar uma pagina html, mas sim um
array de bytes
      @ResponseBody public byte[] getFigura(@PathVariable Long id) {
            // carrega os dados do item
            Item item = itemRepository.findOne(id);
            // cria array de bytes
            byte[] fileBytes = new byte[0];
            // verifica se o item possui uma imagem
            if (item.getCaminhoFigura() != null &&
                        !"".equals(item.getCaminhoFigura().trim())) {
                  // le o arquivo
                  File logo = new File(item.getCaminhoFigura());
            try {
                  // transforma o arquivo em array de bytes
                        fileBytes = Files.readAllBytes(logo.toPath());
                  } catch (IOException e) {
                        e.printStackTrace();
                  }
            }
        return fileBytes;
      }
      @RequestMapping("/aprovacoes_pendentes")
      public ModelAndView listarAprovacoesPendentes() {
            ModelAndView mv = new ModelAndView("AprovacoesPendentes");
            mv.addObject("itens", itemRepository.findByAprovado(null));
            return mv;
      }
}
```

Listagem 6 – ItemController

Outra característica importante nas categorias apresentadas (CategoriaController e ItemController), é a função para salvar os registros. Após verificar se existem erros, o item é salvo para que se obtenha um identificador.

O usuário ou o administrador tem a opção de adicionar uma figura referente ao exemplo que deseja cadastrar ou sugerir. Um método verifica se o arquivo está vazio, cria uma pasta, caso ela não exista e atribui a ela o nome de “item” ou “categoria”. Caso uma imagem seja adicionada, um arquivo é criado e gravado por meio de uma função.

Ressalta-se que os usuários que não necessitam fazer *login* no sistema, podem enviar sugestões para serem cadastradas e, após serem aceitas pelo administrador, são apresentadas no sistema. Um *if* ao final do método salvar faz a seguinte verificação: se o usuário for nulo, representa que ele não está logado ao sistema e uma mensagem é exibida informando que o item foi salvo, mas aguarda a aprovação do administrador. Caso o usuário esteja logado o

item é gravado e o método retornará um ModelAndView, mostrando a tela de “Novo Item” ao usuário.

A classe abstrata FileUtils possui dois métodos, criarPastaSeNaoExistir e gravarArquivo, ambos são usados nos métodos salvar do controlador de categorias e itens. Essa classe possui os métodos mostrados na Listagem 7.

```
package br.edu.utfpr.tutorials.utils;
import java.io.BufferedOutputStream;
import java.io.File;
import java.io.FileOutputStream;
import org.springframework.util.FileCopyUtils;
import org.springframework.web.multipart.MultipartFile;

public abstract class FileUtils {

	public static void criarPastaSeNaoExistir(String caminho) {
		File file = new File(caminho);
		// verifica se o arquivo existe
		if (!file.exists()) {
			// cria o arquivo, caso nao exista
			file.mkdirs();
		}
	}

	public static void gravarArquivo(MultipartFile inputFile, File outputFile)
throws Exception {
		// arquivo que sera escrito
		FileOutputStream fileOutput = new FileOutputStream(outputFile);
		// usa o buffer
		BufferedOutputStream stream = new BufferedOutputStream(fileOutput);
		// executa o processo de copia do arquivo de entrada e gravacao do
arquivo de saida
	 FileCopyUtils.copy(inputFile.getInputStream(), stream);
	 // encerra o buffer
		stream.close();
	}
}
```

Listagem 7 - Classe FileUtils

As funções criarPastaSeNaoExistir e gravarArquivo são chamadas nos métodos salvar, pois uma delas criar a pasta onde as imagens serão salvas, caso ela ainda não exista, e a outra grava o arquivo que será salvo.

5 CONCLUSÃO

Este trabalho teve como objetivo principal apresentar a modelagem e o desenvolvimento de aplicação *web*, que consiste em permitir armazenamento e consulta de exemplos do uso de recursos de tecnologias. Neste trabalho foram registrados conceitos, códigos e *views* das tecnologias HTML5, CSS3, JavaScript e Bootstrap.

Atualmente há alguns sites que disponibilizam informações semelhantes ao que é proposto neste trabalho, porém a maioria está em inglês. A utilização de um sistema *web*, na forma de repositório, permite que os usuários possam sugerir itens ou categorias a serem adicionadas. Assim esse repositório poderá ser utilizado por qualquer usuário que tenha interesse em aprender as tecnologias cadastradas. Isso porque é possível disponibilizar exemplos de códigos, imagens dos resultados e as definições dos elementos.

Várias ferramentas foram utilizadas no desenvolvimento deste projeto, uma delas é o Bootstrap, um *framework* que possui características que agilizam o desenvolvimento, como, o sistema de *grids* para posicionar os elementos. Esse sistema tem a função de possibilitar *design* responsivo para suportar diferentes tamanhos de telas. Outras vantagens desse *framework* é a vasta documentação disponibilizada e, também, possui componentes suficientes para suportar o desenvolvimento de qualquer site que contém interfaces simples, além de facilitar a criação e edição de leiautes por manter os padrões da W3C para manter a aparência padronizada do sistema em todos os navegadores. Uma desvantagem do Bootstrap é a padronização dos elementos, pois para alterar a estilização dos mesmos deve-se criar uma folha de estilos separada utilizando-se o mesmo nome de classe.

O Spring é um conjunto de *frameworks* que fornece suporte e infraestrutura abrangente para o desenvolvimento de aplicações Java. Foi desenvolvido com o intuito de simplificar a programação, é estruturado em padrões de inversão de controle e injeção de dependências e também possui uma grande documentação. Uma desvantagem deste *framework* é que toda a configuração é feita por arquivos *Extensible Markup Language* (XML), o que torna complexo o entendimento do *framework* para desenvolvedores iniciantes.

O uso de *frameworks* e demais tecnologias, enriquecem a aplicação, e mantém o foco em atender as necessidades dos usuários possibilitando uma melhora na execução de suas atividades.

Como trabalhos futuros para a aplicação desenvolvida, sugere-se a possibilidade de inclusão de comentário aos exemplos cadastrados.

REFERÊNCIAS

BENJAMIN, Kamara; BOCHMANN, Gregor v.; JOURDAN, Guy-Vincent; ONUT, Iosif-Viorel. **Some Modeling Challenges when Testing Rich Internet Applications for Security.** Third International Conference on Software Testing, Verification, and Validation Workshops, p. 403=409, 2010.

CAMERON. O'Rourke. A look at rich internet application. **Oracle Magazine**. Jul./ago., 2004, p. 1-4.

CAZENAVE, Fabien; QUINT, Vincent; ROISIN, Cécile. **Timesheets.js: When SMIL Meets HTML5 and CSS3. In:** 11th ACM symposium on Document engineering, p. 43-52, 2011.

CORREIA, Edward J. **What's next for HTML5?**. Intel Software Adrenaline, 2013. Disponível em: < https://software.intel.com/en-us/articles/whats-next-for-html5>. Acesso em: 18 mar. 2016.

DEVMEDIA. **Responsive Design: dicas para tornar seu site acessível em qualquer resolução.** Disponível em: <http:// http://www.devmedia.com.br/responsive-design-dicas-para-tornar-seu-site-acessivel-em-qualquer-resolucao/28316>. Acesso em: 08 jun. 2016.

DISSANAYAKE, Nalaka R.; DIAS, G.K.A. **Best practices for rapid application development of AJAX based Rich Internet Applications**. In: International Conference on Advances in ICT for Emerging Regions (ICTer), 2014, p. 63-66.

DRIVER, Mark; VALDES, Ray; PHIFER, Gene. Rich internet applications are the next evolution of the web, **Tech. report**, Gartner, 2005.

FRANKSTON, Bob. HTML5. **IEEE Consumer Electronics Magazine**. April 2014, p. 62-67.

FREITAG, Dayne. **Information extraction from HTML: application of a general machine learning approach**. In AAAI/IAAI, p. 517–523, 1998.

GONZALEZ, Jose L.; MARCELIN-JIMENEZ, Ricardo. **Phoenix: a fault-tolerant distributed web storage based on URLs**. In Parallel and Distributed Processing with Applications (ISPA), 2011 IEEE 9th International Symposium on, pages 282–287. IEEE, 2011.

JEMEL, Mayssa; SERHROUCHNI, Ahmed. **Security Enhancement of HTML5 Local Data Storage**. 2014 International Conference and Workshop on the Network of the Future (NOF), 2014, p. 1-2.

JIANPING, Yang; JIE, Zhang. **Towards HTML 5 and Interactive 3D Graphics**. 20IO International Coriference on Educational and Information Technology (ICEIT 2010), VI-522-VI-527, 2010.

LAWTON, George. New ways to build rich Internet applications. **Computer. Published by the IEEE Computer Society**, p. 10-12, 2008.

NASEEM, Syed Zagham; MAJEED, Fiaz. **Extending HTML5 local storage to save more data; efficiently and in more structured way**. IEEE, 2013, p. 337-340.

PEINTNER, Daniel; KOSCH, Harald; HEUER, Jörg. **Efficient XML interchange for Rich Internet Applications**. IEEE International Conference on Multimedia and Expo, 2009, p. 149-152

SILVA, Maurício Samy. **CSS3: desenvolva aplicações web profissionais com uso dos poderosos recursos de estilização das CSS3**. São Paulo: Novatec Editora, 2012.

STEARN, Brent. XULRunner: a new approach for developing rich internet applications. **IEEE Internet Computing**, v. 11, p. 67-73, 2007.

STRÍBNÝ, Martin; SMUTNÝ, Pavel Using HTML5 Web Interface for Visualization and Control System. **International Carpathian Control Conference (ICCC), 14th, 2013, p.** 363 – 366.

STRÍBNÝ, Martin; SMUTNÝ, Pavel. **Using HTML5 web interface for visualization and control system**. 14th International Carpathian Control Conference (ICCC), 2013, p. 363 - 366.

TECMUNDO. **O que é CSS?** Disponível em: <http://www.tecmundo.com.br/programacao/2705-o-que-e-css-.htm>. Acesso em: 18 mar. 2016.

TERUEL, Evandro Carlos. **HTML5: Guia Prático**. São Paulo: Érica, 2011.

W3C. **CSS specifications**. Disponível em: <https://www.w3.org/Style/CSS/current-work.en.html>. Acesso em: 17 mar. 2016.