UNIVERSIDADE TECNOLÓGICA FEDERAL DO PARANÁ
DEPARTAMENTO ACADÊMICO DE INFORMÁTICA
CURSO DE ESPECIALIZAÇÃO EM TECNOLOGIA JAVA

JOCEANO ALVES DE BORBA

SISTEMA WEB PARA EMISSÃO DE CUPONS DE DESCONTO

MONOGRAFIA DE ESPECIALIZAÇÃO

PATO BRANCO
2017

JOCEANO ALVES DE BORBA

SISTEMA WEB PARA EMISSÃO DE CUPONS DE DESCONTO

Monografia de especialização apresentada na disciplina de Metodologia da Pesquisa, do Curso de Especialização em Tecnologia Java, do Departamento Acadêmico de Informática, da Universidade Tecnológica Federal do Paraná, Câmpus Pato Branco, como requisito parcial para obtenção do título de Especialista.

Orientador: Profa. Beatriz Terezinha Borsoi

PATO BRANCO
2017

MINISTÉRIO DA EDUCAÇÃO
Universidade Tecnológica Federal do Paraná
Câmpus Pato Branco
Departamento Acadêmico de Informática
Curso de Especialização em Tecnologia Java

UTFPR

TERMO DE APROVAÇÃO

SISTEMA WEB PARA EMISSÃO DE CUPONS DE DESCONTO

por

JOCEANO ALVES DE BORBA

Este trabalho de conclusão de curso foi apresentado em 26 de agosto de 2017, como requisito parcial para a obtenção do título de Especialista em Tecnologia Java. Após a apresentação o candidato foi arguido pela banca examinadora composta pelos professores Beatriz Terezinha Borsoi (orientador), Andreia Scariot Beulke e Vinicius Pegorini, membros da banca. Em seguida foi realizada a deliberação pela banca examinadora que considerou o trabalho aprovado.

Beatriz Terezinha Borsoi
Prof. Orientador (UTFPR)

Andreia Scariot Beulke
Banca (UTFPR)

Vinicius Pegorini
Banca (UTFPR)

Robison Cris Brito
Coordenador da IV Especialização
em Tecnologia Java

A Folha de Aprovação assinada encontra-se na Coordenação do Curso.

RESUMO

BORBA, Joceano Alves de. Sistema web para emissão de cupons de desconto. 2017. 52 f. Monografia (Trabalho de especialização) – Especialização em Tecnologia Java - Departamento Acadêmico de Informática, Universidade Tecnológica Federal do Paraná, Câmpus Pato Branco. Pato Branco, 2017.

Cupons de desconto, promocionais e outras denominações que de alguma forma indiquem vantagem de compra são estratégias que as empresas podem utilizar para desfazer-se de produtos em estoque, realizar propaganda, promoção de produtos com vendas conjuntas, entre outras estratégias de propaganda e/ou de gestão. Para os consumidores, promoção é sempre uma oportunidade de adquirir produtos e serviços com preços mais acessíveis. Os cupons de desconto podem ser impressos ou simplesmente virtuais. Nesse caso, o usuário tem um código ou identificador para que o cupom que ele possui possa ser utilizado pela empresa emitente. A facilidade de acesso dos sistemas *web* os torna mais adequados quando o seu público usuário é grande e/ou geograficamente disperso. É esse o público que pode estar envolvido quando produtos e serviços são vinculados a cupons de desconto. Visando prover uma forma de facilitar o gerenciamento desses cupons pelas empresas e pelos seus usuários, por meio deste trabalho foi desenvolvimento um sistema para emitir e controlar o uso de cupons de desconto. Por meio do sistema as empresas “emitem” cupons e o sistema faz o controle da quantidade de cupons já utilizados pelos usuários e do prazo de validade da promoção. O sistema foi desenvolvido com tecnologia Java, destacando-se o uso de Angular JS Material para a implementação da Interface.

Palavras-chave: Cupons de desconto. Angular JS Material. Aplicações Internet Ricas.

ABSTRACT

BORBA, Joceano Alves de. Web system for issuing discount coupons. 2017. 52 f. Monografia (Trabalho de especialização) – Especialização em Tecnologia Java – Departamento Acadêmico de Informática, Universidade Tecnológica Federal do Paraná, Câmpus Pato Branco. Pato Branco, 2017.

Discount coupons, promotional coupons and other denominations that in some way indicate buying advantages are strategies that companies can use to discard products in stock, advertise, promote products with joint sales, and to be used as an advertising and/or management strategies. For consumers, promotion is always an opportunity to purchase products and services at more affordable prices. Discount coupons can be printed or virtual. In this case, the user has a code or identifier so that the coupon can be used by the issuing company. The ease of access of web systems makes them more suitable when there are many consumers and/or they are geographically dispersed. This is the public that may be involved when products and services are tied to discount coupons. Aiming to provide a way to facilitate the management of these coupons by companies and their users, through this work was developed a web system to control the emission and use of discount coupons. Through the system companies "issue" coupons and the system controls the amount of coupons distributed, already used by users and the expiration date of the promotion. The system was developed with Java technology, highlighting the use of Angular JS Material in the implementation of the interface.

Keywords: Discount coupons. Angular JS Material. Rich Internet Applications.

LISTA DE FIGURAS

LISTA DE QUADROS

LISTAGENS DE CÓDIGOS

LISTA DE SIGLAS

AOP	*Aspect-Oriented Programming*
API	*Applications Programming Interface*
CSS	*Cascade Style Sheet*
HTML	*Hypertext Markup Language*
HTTP	*Hypertext Transfer Protocol*
IDE	*Integrated Development Environment*
JDBC	*Java Database Connectivity*
JMS	*Java Message Service*
JPA	*Java Persistence API*
JSON	*JavaScript Object Notation*
JWT	*JSON Web Token*
MVC	*Model-View-Controller*
MVW	*Model-View-Whatever*
SQL	*Structured Query Language*

SUMÁRIO

1 INTRODUÇÃO

Cupons de desconto podem ser visto como uma espécie de código disponibilizado para que a pessoa tenha desconto ao adquirir produtos e serviços nas empresas que emitiram esses cupons ou que estejam vinculadas à referida promoção. Um cupom de desconto pode ter outros nomes como cupom promocional, vale-compra, vale de desconto, código promocional e código de desconto (LOUCOS POR CUPONS, 2016). Com o advento do comércio eletrônico, os cupons de desconto passaram a fazer parte do dia-a-dia das pessoas que fazem compras pela Internet. Eles têm sido usados como um atrativo pelas empresas que desejam aumentar suas chances de vendas no comércio eletrônico. Esse modelo de negócios se baseia no sistema de distribuição de cupons de descontos para determinados produtos ou até para toda uma loja (CASADIO, 2013).

Embora a disseminação dos cupons tenha crescido exponencialmente com a possibilidade de compras pela Internet, Daroit (2016) lembra que os primeiros cupons de que se tem notícia datam de 1887, quando a Coca-Cola distribuiu tíquetes que poderiam ser trocados por refrigerantes grátis por meio de seus vendedores. Mais recentemente, Tuttle (2010) declara que a recessão de 2009 foi um ano de grande uso dos cupons, chegando à marca de 3,5 bilhões de cupons distribuídos somente nos Estados Unidos.

Na Internet é possível encontrar diversos sites especializados em cupons de desconto. Por exemplo, o site Loucos por Cupons, disponível em http://loucosporcupons.com.br/, se define como um portal que disponibiliza as melhores lojas com as melhores oportunidades de cupons de desconto.

Considerando as possibilidades do comércio eletrônico e da grande disseminação dos cupons de desconto, neste trabalho é apresentado o desenvolvimento de um sistema *web* que tem como objetivo principal possibilitar que empresas disponibilizem cupons de desconto. Esses cupons são cadastrados pela própria empresa, que indica dados como as características da promoção (valor ou percentual de desconto), quantidade de cupons, prazo de validade da promoção. Entre os objetivos específicos do sistema, voltados para o usuário dos cupons, estão a possibilidade de o cliente localizar a promoção desejada, cadastrar-se no sistema informando seu e-mail, adquirir o cupom de desconto e realizar a sua utilização, caso deseje.

O sistema não inclui a realização de transações financeiras, apenas permite que o usuário adquira cupons de desconto que foram previamente cadastrados pelas respectivas empresas, podendo utilizá-lo quando desejar.

2 FERRAMENTAS E TECNOLOGIAS

A ênfase deste capítulo está em reportar as ferramentas e as tecnologias utilizadas para a modelagem e a implementação do sistema.

2.1 FERRAMENTAS E TECNOLOGIAS

O Quadro 1 apresenta as ferramentas e as tecnologias utilizadas na modelagem e na implementação do aplicativo desenvolvido como resultado da realização deste trabalho.

Ferramenta / Tecnologia	Versão	Disponível em	Aplicação
Astah Community	7.1.0	http://astah.net/editions/community	Modelagem do sistema: desenvolvimento do diagrama de casos de uso
SQL Power Architect	1.0.8	http://www.sqlpower.ca/page/architect_download_os	Modelagem do sistema: desenvolvimento do banco de dados
Java Platform (JDK)	1.8.0_73	http://www.oracle.com/	Linguagem para desenvolvimento da aplicação
Angular JS Material	1.0.9	https://material.angularjs.org/latest/	Framework para *Hypertext Markup Language* (HTML), *Cascade Style Sheet* (CSS) e JavaScript.
Spring Framework, Spring Data JPA e Spring Security	4.3.8	http://www.spring.io/	*Framework Model-View-Controller* (MVC) de desenvolvimento web
AngularJS	1.6.4	https://angularjs.org/	Biblioteca JavaScript
Eclipse Neon	4.6.0	https://projects.eclipse.org/releases/luna	*Integrated Development Environment* (IDE) para desenvolvimento da aplicação
Apache Maven	3.5.0	https://maven.apache.org/index.html	Ferramenta de automação de compilação de projetos Java
MySQL	5.7	http://dev.mysql.com/downloads/mysql/	Banco de dados
MySQL Workbench	6.3.8	http://dev.mysql.com/downloads/workbench/	IDE de gerenciamento de bases de dados MySQL

Quadro 1 – Tecnologias e ferramentas utilizadas

As ferramentas e tecnologias listadas no Quadro 1 são de uso bastante comum, embora algumas sejam bem recentes, no desenvolvimento de aplicações *web* utilizando a linguagem Java. A seguir é apresentado sobre as tecnologias consideradas mais recente no desenvolvimento *web*. Não será exposto sobre a linguagem Java e as IDEs utilizadas no desenvolvimento, apenas tecnologias.

O AngularJS Material é um *framework* JavaScript *Model-View-Whatever* (MVW) da Google, alternativo ao MVC. Esse *framework* é utilizado para a criação de aplicações *web* que se conectam a *Applications Programming Interface* (API) e não precisam de uma página para serem atualizadas (GALDINO, 2017). Na página do Angular JS Material, essa tecnologia é definida como um *framework* de componentes de interface com o usuário e uma referência à implementação da especificação do Design Material da Google. Esse projeto provê um conjunto de componentes de interface com o usuário baseados no Material Design que são reusáveis, bem testado e acessíveis (ANGULAR, 2017).

O Spring é um *framework* de código aberto criado por Rod Johnson visando simplificar o desenvolvimento em Java. Spring conta com diversos módulos, como o Spring Data e Security, sendo que alguns deles são aplicáveis somente ao desenvolvimento *web*, porém o principal pode ser aplicado em qualquer aplicativo Java (GENTIL, 2017).

A biblioteca Spring Framework fornece um modelo de desenvolvimento e configuração de aplicativos baseados em Java. O foco do Spring é montar a estrutura para os aplicativos de maneira que as equipes fiquem centradas nas regras de negócio do sistema (SPRING FRAMEWORK, 2017).

O Spring Framework fornece injeção de dependência, *Aspect-Oriented Programming* (AOP), desenvolvimento facilitado do padrão MVC para aplicações *web* e RESTful, suporte à *Java Database Connectivity* (JDBC), *Java Persistence API* (JPA), *Java Message Service* (JMS), entre outros, em vários módulos distintos.

O Spring Data JPA é um dos projetos Spring Data com o propósito de fornecer suporte a repositórios JPA. A implementação de acesso a dados é, geralmente, complexa, seja por repetição de código ou por necessidade de readequação das consultas para diferentes bancos de dados. O objetivo do Spring Data JPA é facilitar o acesso aos dados, uma vez que é necessário apenas que uma interface seja codificada para que os métodos padrões, como consultas por chave,

persistência, atualização de dados, entre outros, sejam fornecidos automaticamente pelo *framework* (SPRING DATA JPA, 2017).

O Spring Security é um *framework* de autenticação e controle de acesso focado em oferecer autenticação e autorização em aplicações Java. Possui como principais funcionalidades proteção contra ataques em sessões, clickjacking (roubo de clique, quando os cliques do usuário em uma página *web* são capturados para realizar ações maliciosas e/ou fraudulentas), possibilidade de integração com o Spring Web MVC, dentre outros (SPRING SECURITY, 2017).

O MySQL é o sistema de gerenciamento de banco de dados *Structured Query Language* (SQL) *open source*. Desenvolvido e distribuído pela Oracle, é um banco de dados relacional de fácil utilização e escalável. Possui suporte a muitos tipos de dados, como *float* e *enum* e diversas funcionalidades como as cláusulas Group By e Order By, retorno de número de linhas afetadas em comandos de Delete, Insert, Update e Replaces, recuperação de informações das tabelas por meio do comando Show e o comando Explain com a função de exemplificar como a consulta é resolvida (MYSQL, 2017).

3 RESULTADO

Este capítulo apresenta o resultado da realização do trabalho que é o desenvolvimento de um sistema para emissão de cupons de desconto.

3.1 ESCOPO DO SISTEMA

O sistema visa possibilitar que empresas disponibilizem cupons de desconto. A empresa cadastrada no sistema define os dados relacionados aos cupons, valor promocional, prazo de validade, a quantidade de cupons a serem disponibilizados, entre outras informações e o sistema faz o controle da quantidade de cupons já utilizados pelos usuários e do prazo de validade da promoção. A apresentação do escopo do sistema a seguir está organizada pelas suas principais funcionalidades, definindo um fluxo de execução do processo de negócio.

a) Cadastro de acesso ao sistema

Todos que desejarem disponibilizar um anúncio de desconto, ou adquirir um cupom de desconto de um anúncio já cadastrado no sistema, sendo ele pessoa física ou jurídica, deve inicialmente cadastrar-se no sistema. Esse cadastro poderá ser acessado pelo *link* disponibilizado na página principal de anúncios de descontos. Esse cadastro conterá informações como nome, *e-mail*, senha, telefone, entre outros.

b) Cadastro de anúncios

Após o anunciante ter se cadastrado no sistema, poderá realizar o cadastro dos seus anúncios de ofertas. Esses, por sua vez, ficarão aguardando aprovação do administrador do sistema. A necessidade dessa aprovação visa assegurar melhor controle e evitar fraudes. Esse cadastro conterá informações como descrição da oferta, preço normal e promocional, imagem, detalhes do produto/serviço, regulamento/regras, quantidade máxima de cupons que podem ser adquiridos por cliente, quantidade máxima total de cupons disponibilizados, período de vigência da promoção, categoria e usuário da oferta.

c) Cadastro de categorias de anúncios

Todo o anúncio de oferta cadastrado no sistema deverá estar associado a uma categoria. Essa informação torna-se importante no momento da implementação

de filtros na tela principal dos anúncios. A manutenção das categorias poderá ser realizada apenas pelos usuários administradores do sistema.

d) Página principal de anúncios

Após o anúncio ter sido cadastrado, aprovado pelo administrador e estar em um período de vigência válido, ele ficará visível na página principal do sistema, com algumas informações relevantes, por exemplo, imagem, preço normal, preço promocional, prazo final de vigência, quantidade restante de cupons, entre outras.

e) Página específica de anúncios

Se o cliente se interessar por alguma das ofertas listadas na página principal, ele poderá clicar em um *link*, apresentado abaixo da imagem do produto, e o sistema o direcionará para uma página específica do anúncio selecionado, contendo todas as informações do produto ou serviço a que o cupom de desconto se refere.

f) Geração/Aquisição de cupons de desconto

Com a autenticação realizada no sistema, o usuário poderá gerar/adquirir o cupom de desconto desejado. O sistema por sua vez, fará um controle interno de numeração a fim de manter um histórico dos registros e das informações.

g) Baixa de cupons de desconto

Quando o cliente realiza o pagamento de um produto/serviço utilizando um cupom de desconto válido, o anunciante poderá realizar a baixa deste cupom. Esse processo de baixa serve para informar que determinado cupom foi realmente utilizado. Assim, poderá alimentar informações para possíveis relatórios e também habilitar a avaliação do produto. Para realizar esse procedimento, basta que o anunciante acesse a página específica para esse fim e localize o cupom do seu cliente por meio do *email (username*), nome, telefone ou número do cupom.

h) Avaliação de cupons utilizados

Depois que um cupom de desconto for utilizado e baixado, possibilitará que o cliente realize uma avaliação do produto e do atendimento recebido. Essa avaliação ficará salva na base de dados, podendo alimentar informações para possíveis relatórios e gráficos para o anunciante.

3.2 MODELAGEM DO SISTEMA

A Figura 1 apresenta o diagrama de casos de uso do sistema que possui três atores, sendo eles:

a) Anunciante – é o usuário que registra os anúncios de cupons de desconto no sistema. Após a inclusão, o anúncio ficará aguardando a aprovação do administrador do sistema. O ator anunciante possui direitos de acesso para realizar a manutenção do seu perfil, alterando dados como a senha, por exemplo. Também faz a manutenção de anúncios por ele cadastrados.

b) Administrador – mantém os dados de cadastro que são utilizados pelo usuário anunciante como as categorias de produto. Esse usuário aprova os anúncios incluídos no sistema pelos anunciantes e pode realizar a manutenção de todos os usuários cadastrados, podendo inativá-los em casos de suspeita de fraudes, por exemplo. O administrador faz a manutenção de todos os anúncios cadastrados independente do anunciante e visualiza todos os cupons gerados por todos os usuários.

c) Cliente – usuário que utiliza os cupons de desconto disponíveis e realiza as avaliações relacionadas aos cupons que utilizou. Esse ator tem permissão para visualizar todos os seus cupons adquiridos e também possui acesso para manter o seu perfil, alterando dados como senha, por exemplo.

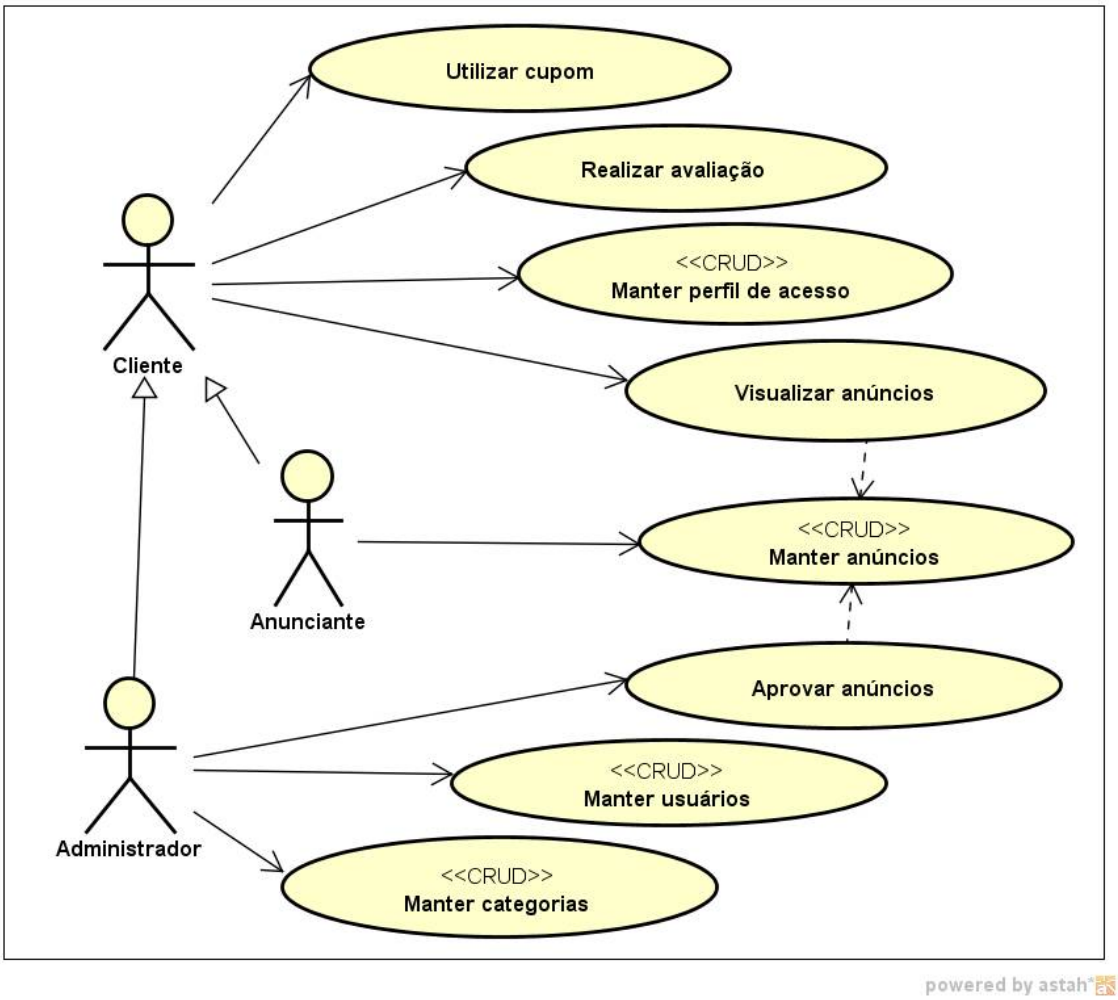

Figura 1 – Diagrama de casos de uso

O diagrama de atividade da Figura 2 apresenta o fluxo, que representa o processo de negócio, das duas principais funcionalidades do sistema que são o anúncio e a aquisição de cupons de desconto relacionados aos anúncios. Na Figura 2, à direita estão as atividades do anunciante e do administrador para o cadastro de anúncios de desconto; à esquerda um usuário quando obtém esses cupons de desconto (cliente) e o portal (representando o sistema *web* nas funcionalidades relacionadas aos anúncios exibidos, registro de cupons utilizados e de avaliações realizadas).

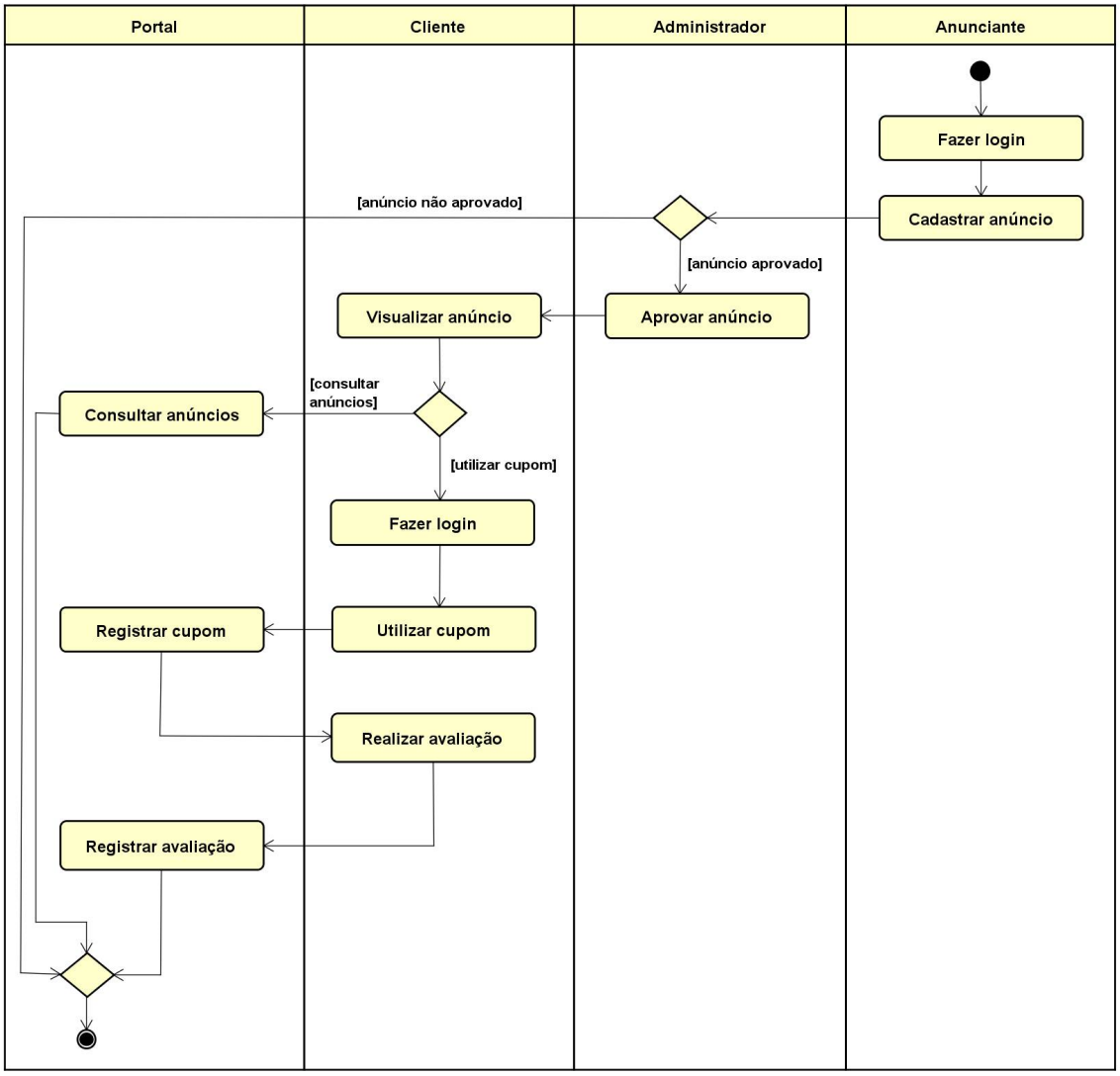

Figura 2 – Diagrama de atividades

Na Figura 3 estão apresentadas as tabelas do sistema com os seus respectivos campos e relacionamentos, representando o banco de dados.

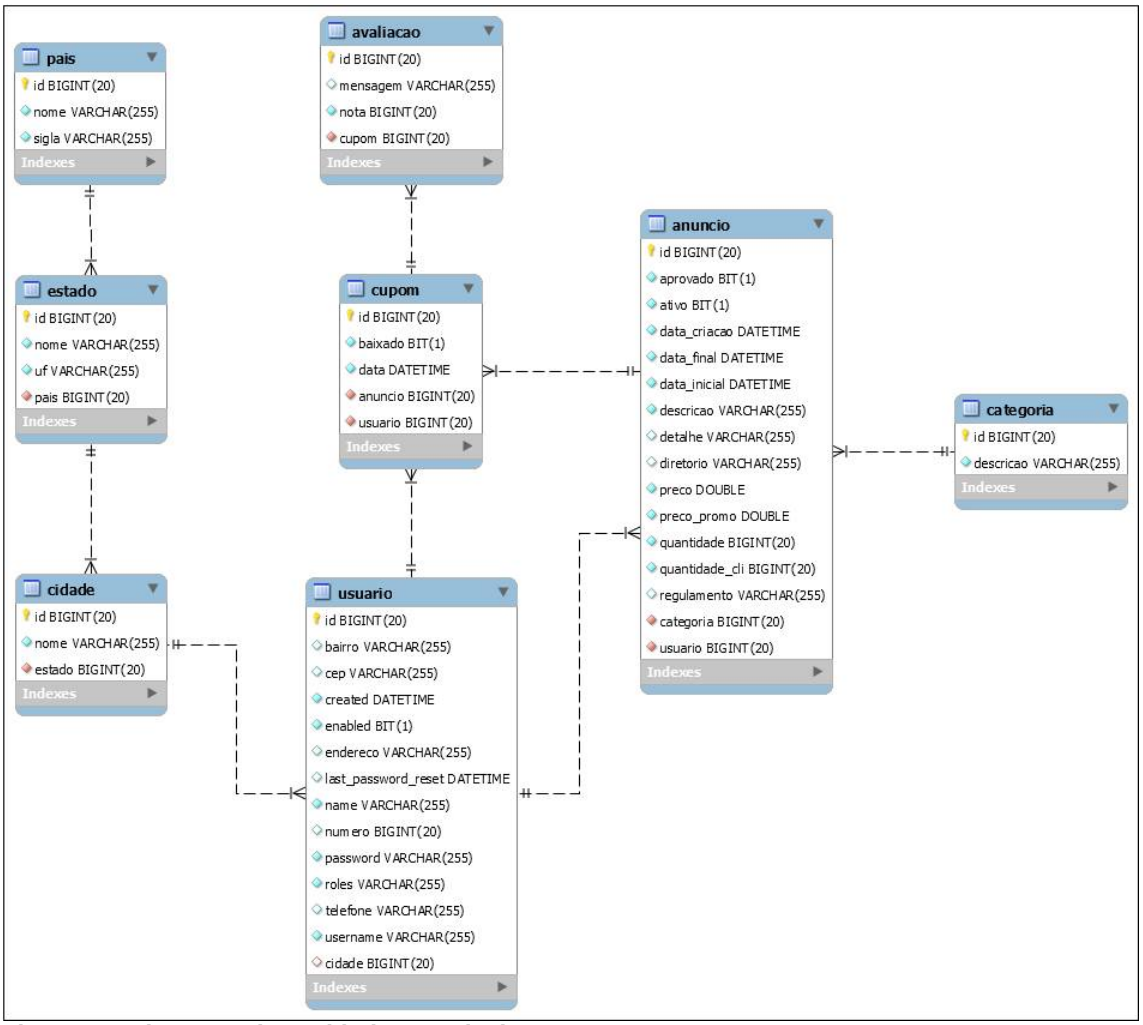

Figura 3 – Diagrama de Entidades e Relacionamentos

A seguir é apresentada a descrição das tabelas que compõem o diagrama da Figura 3.

Quadro 2 – Tabela user. Essa tabela é a responsável por armazenar os usuários cadastrados no sistema

Campo	Tipo	Nulo	Chave primária	Chave estrangeira	Observações
id	Inteiro	Não	Sim	Não	Chave primária
name	Texto	Não	Não	Não	Nome do usuário
username	Texto	Não	Não	Não	E-mail do usuário
created	Data/Hora	Não	Não	Não	Data criação do usuário
enabled	Lógico	Não	Não	Não	Se o usuário está ativo ou inativo
endereco	Texto	Sim	Não	Não	Endereço do usuário
last_password_reset	Data/Hora	Sim	Não	Não	Data última alteração da

					senha
bairro	Texto	Sim	Não	Não	Bairro do usuário
numero	Inteiro	Sim	Não	Não	Número do endereço
password	Texto	Não	Não	Não	Senha do usuário
roles	Texto	Não	Não	Não	Direitos (permissões) de acesso do usuário
telefone	Texto	Não	Não	Não	Telefone do usuário
cep	Texto	Não	Não	Não	Cep do endereço do usuário
cidade	Inteiro	Não	Não	Sim	Identificação da cidade

Quadro 2 – Tabela user

Quadro 3 – Tabela anúncio. Essa é a tabela que armazena os dados dos anúncios cadastrados pelos usuários.

Campo	Tipo	Nulo	Chave primária	Chave estrangeira	Observações
id	Inteiro	Não	Sim	Não	Chave primária
aprovado	Lógico	Não	Não	Não	Se o anúncio aprovado
ativo	Lógico	Não	Não	Não	Se o anúncio ativo
data_criacao	Data/Hora	Não	Não	Não	Data criação do anúncio
data_final	Data	Não	Não	Não	Data vigência final
data_inicial	Data	Não	Não	Não	Data vigência inicial
descricao	Texto	Não	Não	Não	Descrição do anúncio
detalhe	Texto	Sim	Não	Não	Detalhes do anúncio
diretorio	Texto	Sim	Não	Não	Diretório de armazenamento da imagem do anúncio
preco	Numérico	Não	Não	Não	Preço original do anúncio
preco_promo	Numérico	Não	Não	Não	Preço promocional do anúncio
quantidade	Inteiro	Não	Não	Não	Quantidade total do anúncio
quantidade_cli	Inteiro	Não	Não	Não	Quantidade de cupons por cliente
regulamento	Texto	Sim	Não	Não	Regulamento do anúncio
categoria	Inteiro	Não	Não	Sim	Identificação da categoria
usuario	Inteiro	Não	Não	Sim	Identificação do usuário

Quadro 3 – Tabela anúncio

Os campos da tabela cupom estão apresentados no Quadro 4. Essa tabela é responsável por armazenar os cupons que os usuários adquirem a partir dos anúncios disponíveis no sistema.

Campo	Tipo	Nulo	Chave primária	Chave estrangeira	Observações
id	Inteiro	Não	Sim	Não	Chave primária
baixado	Lógico	Não	Não	Não	Cupom utilizado pelo user
data	Data/Hora	Não	Não	Não	Data de aquisição do cupom
anuncio	Inteiro	Não	Não	Sim	Identificação do anúncio
usuario	Inteiro	Não	Não	Sim	Identificação do usuário

Quadro 4 – Tabela cupom

Quadro 5 – Tabela avaliação. Contém dados de avaliações realizadas pelos usuários. Após o usuário utilizar o cupom adquirido, será liberada uma avaliação do produto/atendimento relacionado ao cupom, cujos dados são armazenados nessa tabela.

Campo	Tipo	Nulo	Chave primária	Chave estrangeira	Observações
id	Inteiro	Não	Sim	Não	Chave primária
mensagem	Texto	Sim	Não	Não	Mensagem da avaliação
nota	Inteiro	Não	Não	Não	Nota da avaliação (0-5)
cupom	Inteiro	Não	Não	Sim	Identificação do cupom

Quadro 5 – Tabela avaliação

Quadro 6 – Tabela categoria. Essa tabela conterá as categorias cadastradas pelo administrador. Essas categorias são utilizadas no cadastro de anúncios para identificar a categoria a que o cupom se refere.

Campo	Tipo	Nulo	Chave primária	Chave estrangeira	Observações
id	Inteiro	Não	Sim	Não	Chave primária
descricao	Texto	Não	Não	Não	Nome que identifica a categoria

Quadro 6 – Tabela categoria

Quadro 7 – Tabela país. Os países utilizados no cadastro de Estados.

Campo	Tipo	Nulo	Chave primária	Chave estrangeira	Observações
id	Inteiro	Não	Sim	Não	Chave primária
nome	Texto	Não	Não	Não	Nome do país
sigla	Texto	Sim	Não	Não	Sigla do nome do país

Quadro 7 – Tabela país

Quadro 8 – Tabela estado. Contém as unidades de federação utilizadas no cadastro de cidades. Cada Estado estará vinculado a um país.

Campo	Tipo	Nulo	Chave primária	Chave estrangeira	Observações
id	Inteiro	Não	Sim	Não	Chave primária
nome	Texto	Não	Não	Não	Nome do estado
uf	Texto	Sim	Não	Não	Sigla do nome do estado
pais	Inteiro	Não	Não	Sim	Identificação do país

Quadro 8 – Tabela estado

Quadro 9 – Tabela cidade. A tabela de cidades conterá as cidades utilizadas no cadastro de usuários. Cada cidade estará vinculada a um Estado.

Campo	Tipo	Nulo	Chave primária	Chave estrangeira	Observações
id	Inteiro	Não	Sim	Não	Chave primária
nome	Texto	Não	Não	Não	Nome da cidade
estado	Inteiro	Não	Não	Sim	Identificação do estado

Quadro 9 – Tabela cidade

3.3 APRESENTAÇÃO DO SISTEMA

Nesta seção é apresentado o sistema *web* desenvolvido para o controle de cupons de desconto e a forma de utilizá-lo. Essa apresentação é realizada por meio de imagens das telas do sistema e explicações sobre os procedimentos e os processos executados. O acesso ao sistema é realizado pelo navegador na porta 3000.

A Figura 4 apresenta a tela principal do sistema, sendo também uma página pública, ou seja, não é necessário que o usuário esteja autenticado para ter acesso ao conteúdo dessa página. A interface apresentada nessa Figura é apresentada ao usuário quando ele acessa o sistema, nela estão apresentados todos os anúncios ativos e válidos.

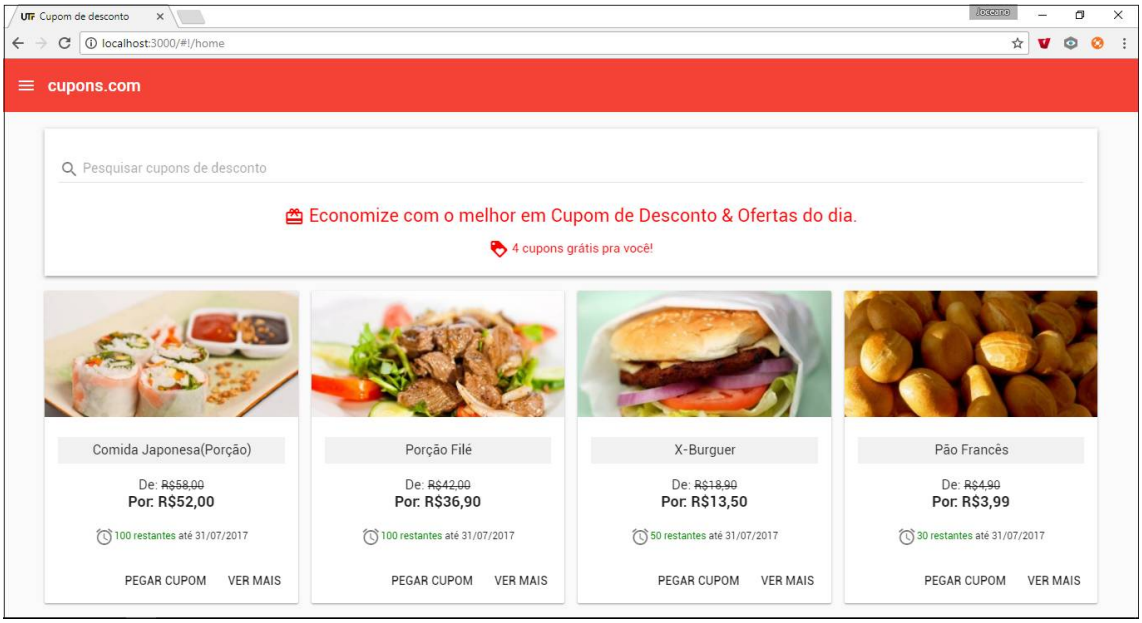

Figura 4 – Tela principal do sistema em resolução maior

O *framework* AngularJS Material permite criar páginas responsivas. Um exemplo dessa responsividade é ilustrado na Figura 5, que exibe a tela principal do sistema em um *smartphone*.

Figura 5 – Tela principal do sistema no smartphone

O menu de navegação do sistema pode ser acessado por meio do ícone existente na barra superior esquerda da página, conforme apresenta a Figura 6, na região destacada.

Figura 6 – Localização do ícone do menu

Na Figura 7 está o menu de navegação de um usuário anônimo, ou seja, não autenticado no sistema. As opções apresentadas, nesse caso, são “Início” para

acesso à página principal e “Fale conosco” para o envio de mensagens para o administrador do sistema.

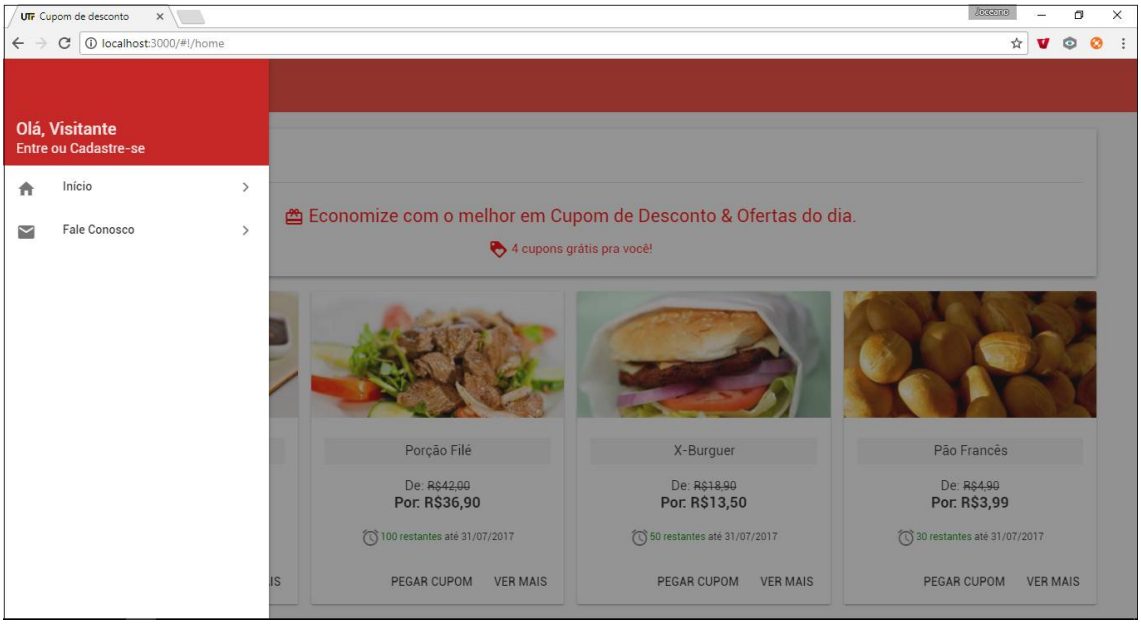

Figura 7 – Menu navegação usuário anônimo

Conforme apresentado na Figura 7, um usuário anônimo poderá acessar o menu de navegação e optar por entrar ou cadastrar-se no sistema.

A Figura 8 apresenta o menu de navegação de um usuário autenticado no sistema. Para usuários autenticados são apresentados, ainda, um *link* de acesso a sua conta e a opção para sair do sistema.

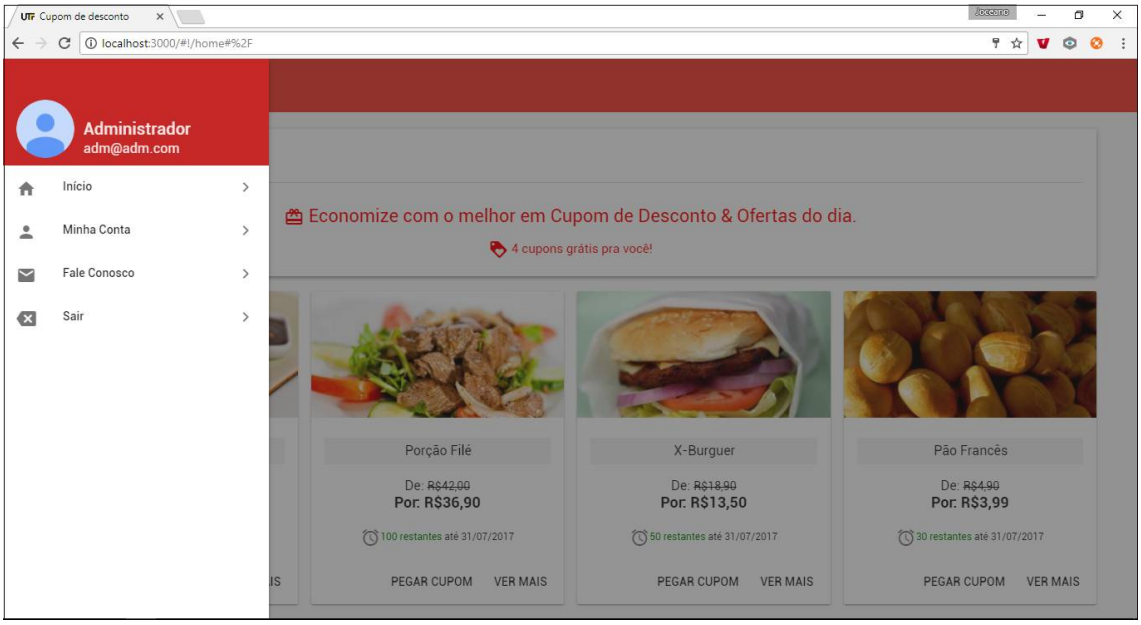

Figura 8 – Menu navegação usuário autenticado

A autenticação é um requisito obrigatório para a aquisição de um cupom de desconto de um anúncio existente ou para o cadastro de um novo anúncio. Portanto, só é apresentada para usuários identificados no sistema.

A Figura 9 exibe a tela que possibilita que o usuário realize a sua autenticação no sistema. Se o valor de algum dos campos estiver incorreto, considerando os dados armazenados no banco de dados para aquele usuário, será apresentada uma mensagem informativa avisando ao usuário para ele verificar o *e-mail* (que é utilizado como identificação de usuário no acesso) e a senha.

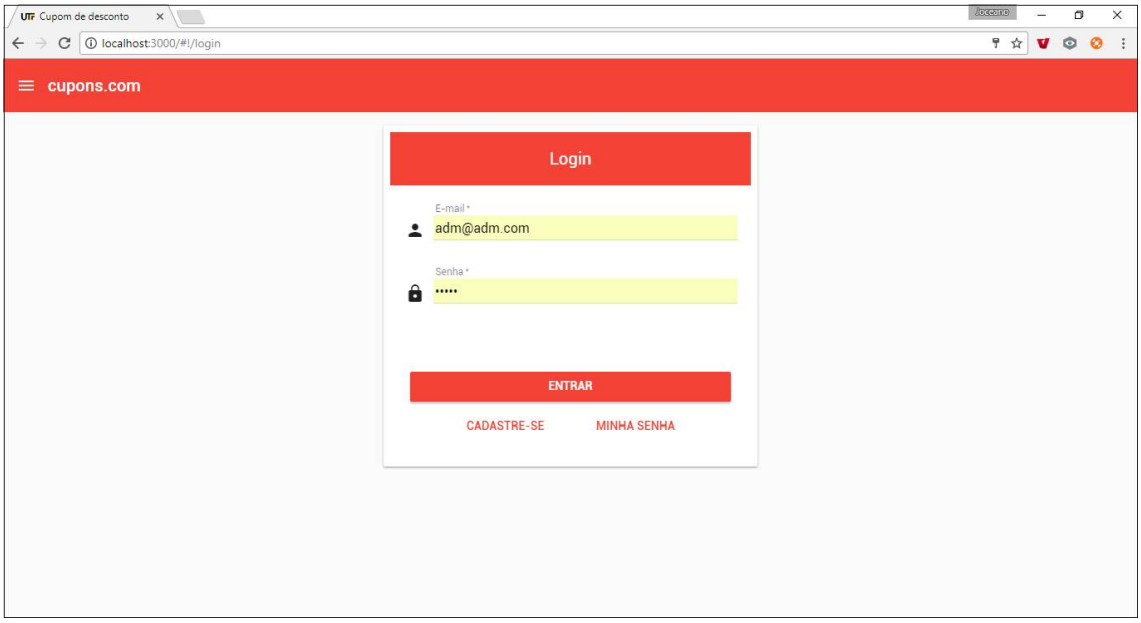

Figura 9 – Tela de login

Na página de *login* da aplicação existe a opção “Cadastre-se”. Ao selecionar essa opção é apresentada uma nova tela em forma de modal, possibilitando que novos usuários se cadastrem no sistema.

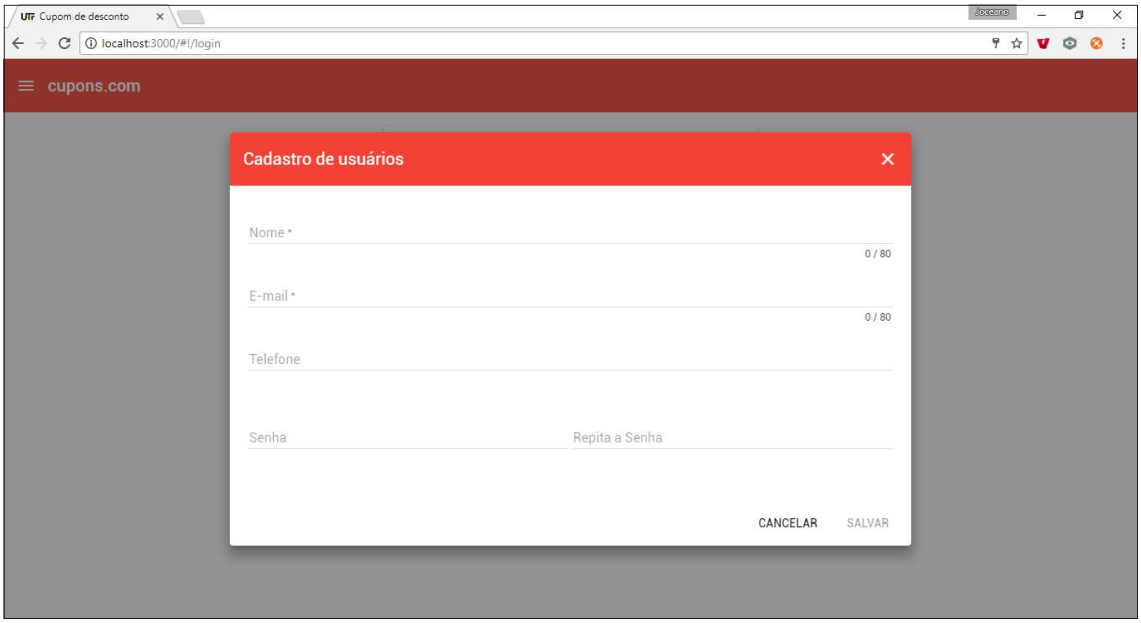

Figura 10 – Cadastro de usuários

Após autenticado, ao clicar na opção “Minha Conta” do menu de navegação, o sistema direcionará o usuário para uma página organizada por abas, contendo as informações do usuário, os seus cupons adquiridos (1), os seus anúncios lançados (2) e os seus dados cadastrais (3), conforme apresenta a Figura 11.

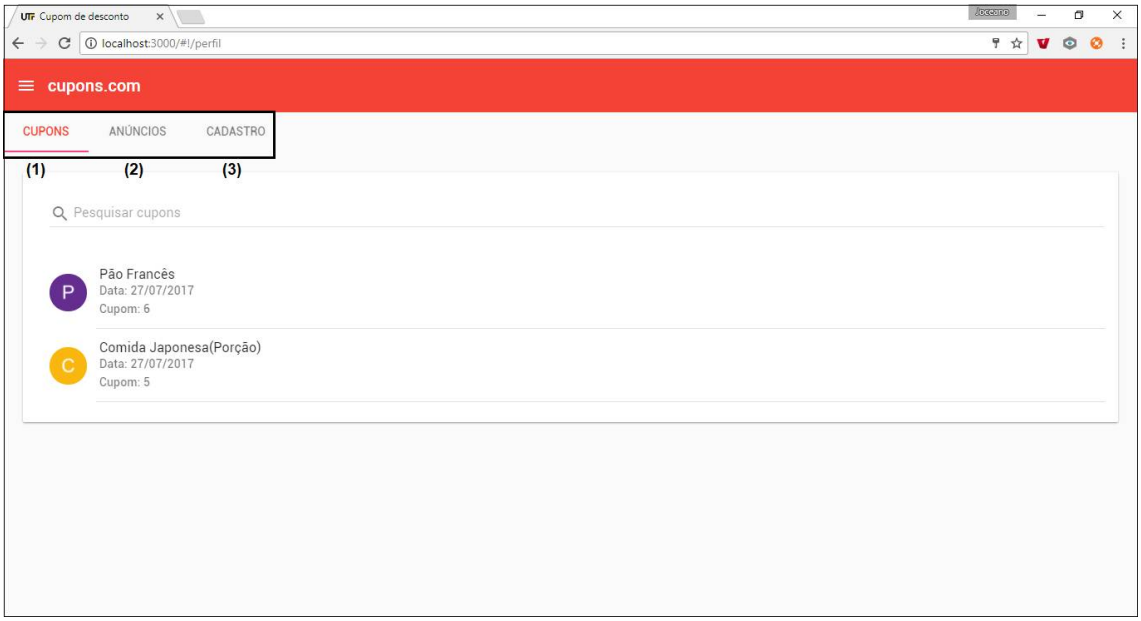

Figura 11 – Tela de perfil de usuários não administradores

A página de perfil sofre algumas alterações para usuários administradores, habilitando duas abas que possibilitam a manutenção de categorias de anúncios (1)

e a manutenção de usuários (2). As diferenças podem ser percebidas ao serem comparadas as Figuras 11 e 12.

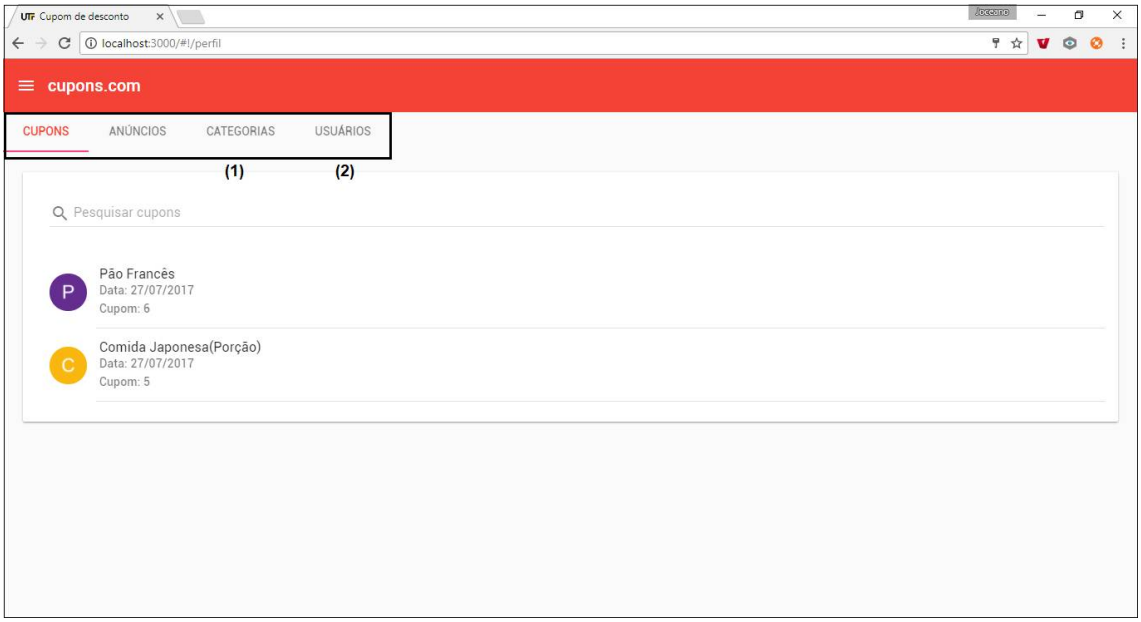

Figura 12 – Tela de perfil de usuários administradores

Na aba “anúncios” (Figura 13) são listados todos os anúncios cadastrados pelo usuário (1) e é disponibilizada a opção para a inclusão de novos registros (2). Após cadastrado, o anúncio ficará aguardando aprovação do administrador do sistema para definitivamente ser disponibilizado na página principal. Por meio da listagem de anúncios também é possível identificar os anúncios que já foram aprovados (3) e os anúncios que aguardam aprovação (4) utilizando o ícone mostrado na lista. A Figura 13 apresenta essa tela, destacando essas funcionalidades.

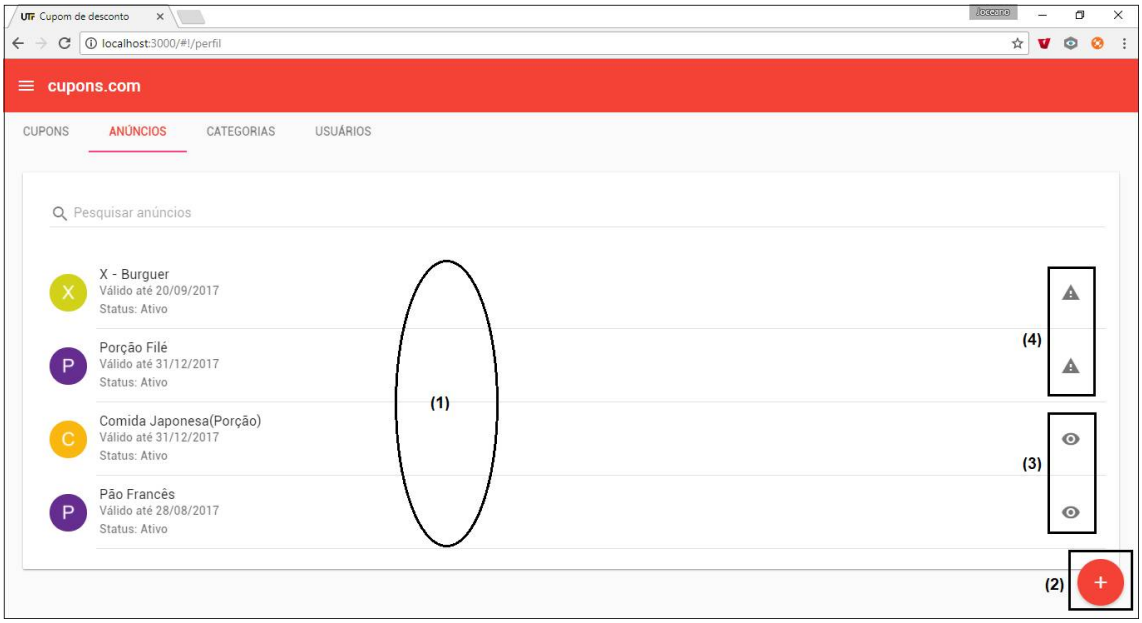

Figura 13 – Lista de anúncios

Ao clicar sobre um item da lista de anúncios será aberta uma tela no formato de modal que possibilita a manutenção do registro. A Figura 14 apresenta ao fundo a listagem de anúncios registrados e em primeiro plano está a tela de cadastro de anúncios.

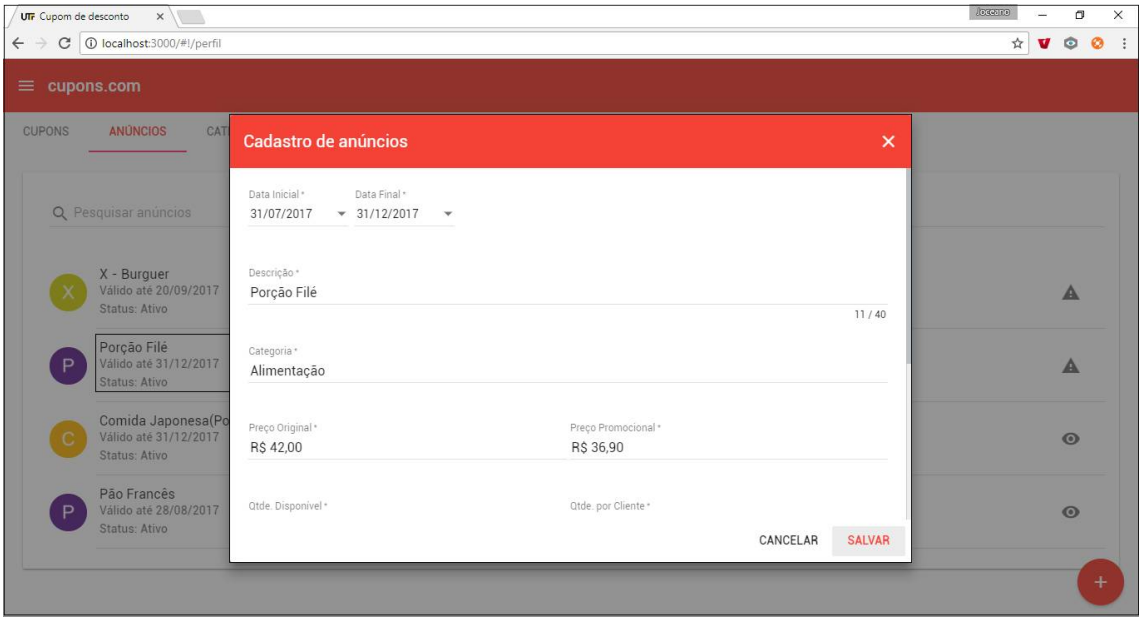

Figura 14 – Manutenção de anúncios

Ao clicar no ícone de anúncio aprovado, será aberta uma tela no formato de modal com a lista de todos os cupons que foram gerados a partir do anúncio (1).

Será por meio dessa lista de cupons que o usuário anunciante poderá realizar a baixa do cupom. O processo de baixa do cupom é utilizado para informar ao sistema que o cupom foi realmente utilizado. Assim, será possível liberar a avaliação do produto/atendimento ao usuário adquirente do cupom. Os cupons já utilizados/baixados são apresentados com um ícone diferenciado (2), como apresentado de forma destacada na Figura 15.

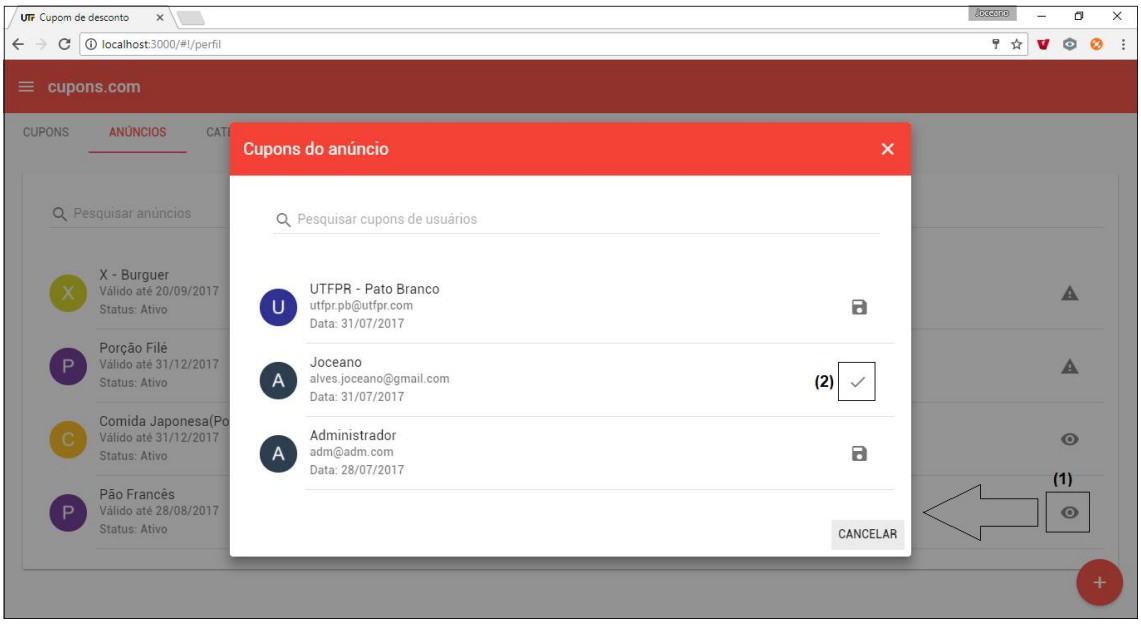

Figura 15 – Lista de cupons do anúncio

Na aba “cupons”, o usuário terá disponível todos os seus cupons adquiridos ao longo do tempo, em ordem decrescente, ou seja, cupons atuais serão listados primeiramente. A Figura 16 apresenta a listagem de cupons de um usuário.

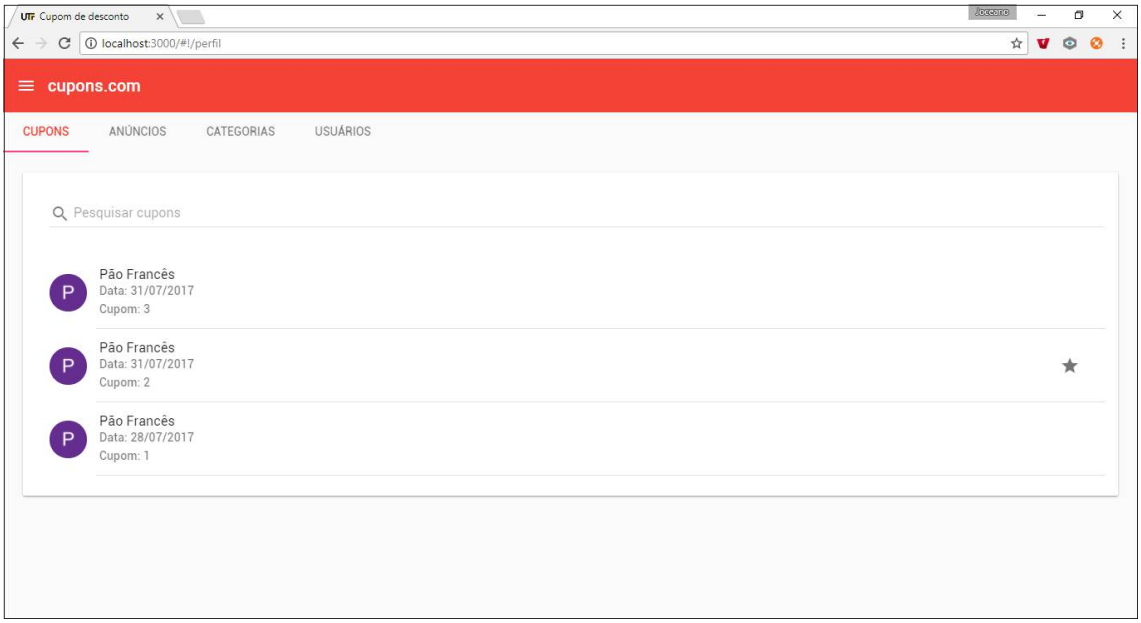

Figura 16 – Lista de cupons do usuário

Ao clicar sobre o registro será mostrada a tela modal contendo todo o detalhamento do cupom. Importante ressaltar que não é possível a manutenção de cupons. Eles são gerados por usuários a partir de um anúncio válido no sistema. Contudo, o cupom em si também não pode ser excluído. É por esse motivo que os campos desta tela são apresentados desabilitados, como apresentado na Figura 17.

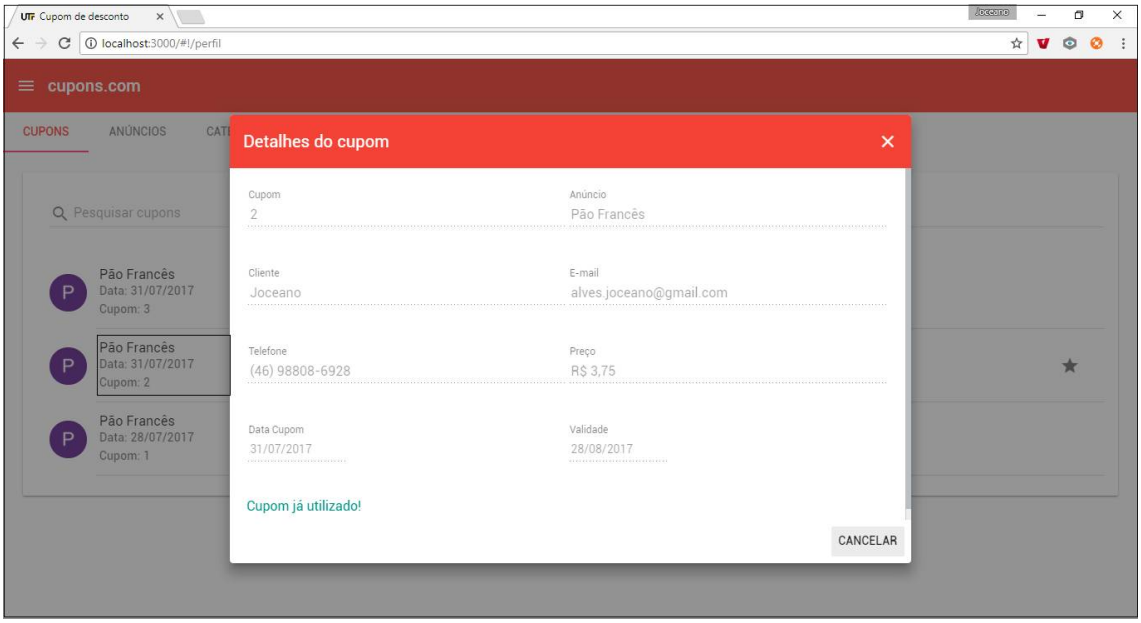

Figura 17 – Detalhes do cupom

Os cupons já baixados no sistema ou utilizados pelo usuário podem ser facilmente identificados na lista, pois possuem um ícone no formato de uma estrela (1) que se clicado direciona para uma tela no formato de modal que permite a realização da avaliação do produto/atendimento (2). O usuário poderá optar por uma nota de zero a cinco, sendo zero totalmente insatisfeito e cinco totalmente satisfeito. Também poderá adicionar um comentário, caso desejar.

A avaliação realizada será armazenada na base de dados para futuros gráficos e relatórios. A Figura 18 mostra a tela de avaliação de forma mais detalhada.

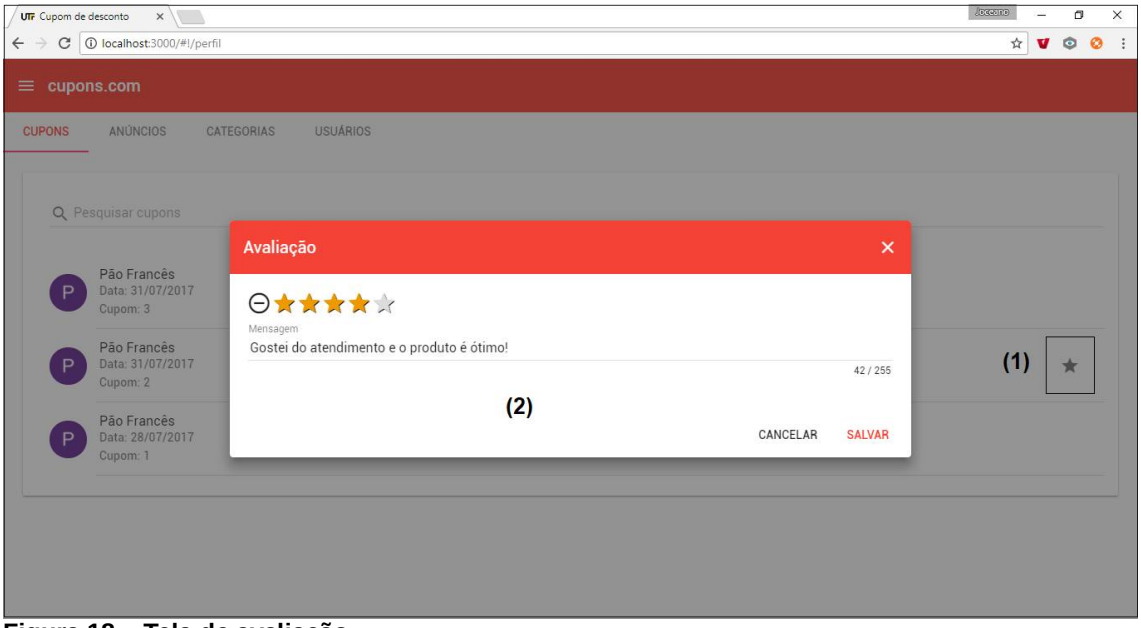

Figura 18 – Tela de avaliação

3.4 IMPLEMENTAÇÃO DO SISTEMA

Nesta seção são apresentados alguns códigos que merecem destaque e que exemplificam o desenvolvimento de um sistema utilizando as tecnologias apresentadas no Quadro 1.

O sistema foi dividido em duas partes, sendo uma API REST utilizando o *framework* Spring e um projeto *front-end* com AngularJs e AngularJs Material. A comunicação entre os dois projetos é realizada por meio de requisições *Hypertext Transfer Protocol* (HTTP) via GET, POST, PUT e DELETE utilizando o formato *JavaScript Object Notation* (JSON).

A Figura 19 mostra a estrutura utilizada para a implementação do projeto API REST com Spring.

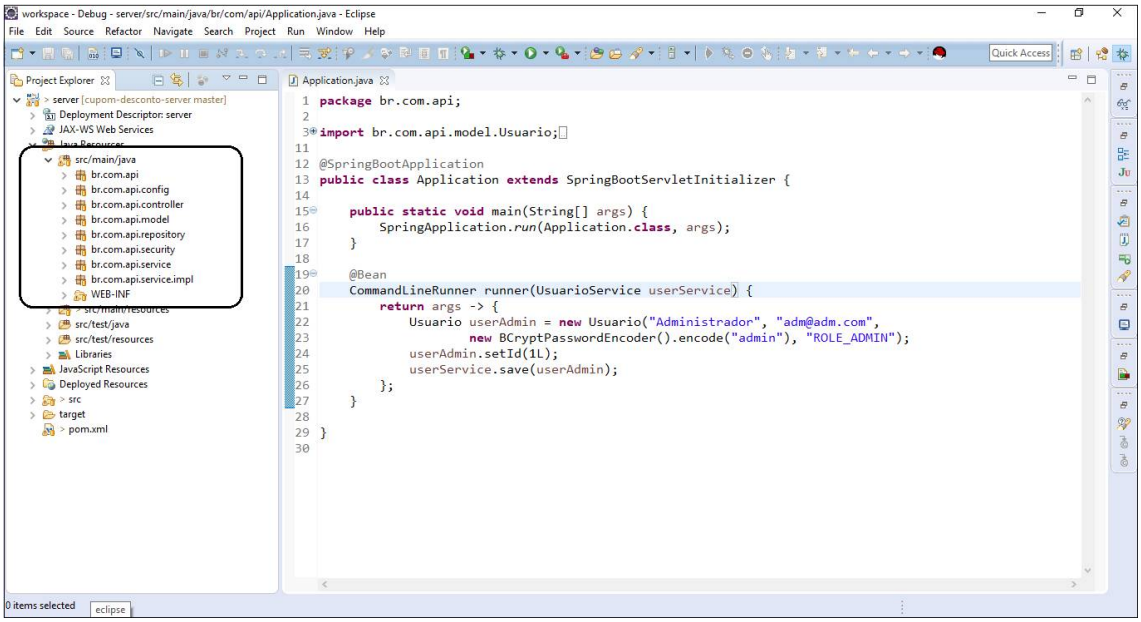

Figura 19 – Estrutura do projeto API

A Figura 20 apresenta a estrutura adotada para a implementação do projeto *front-end* com AngularJs.

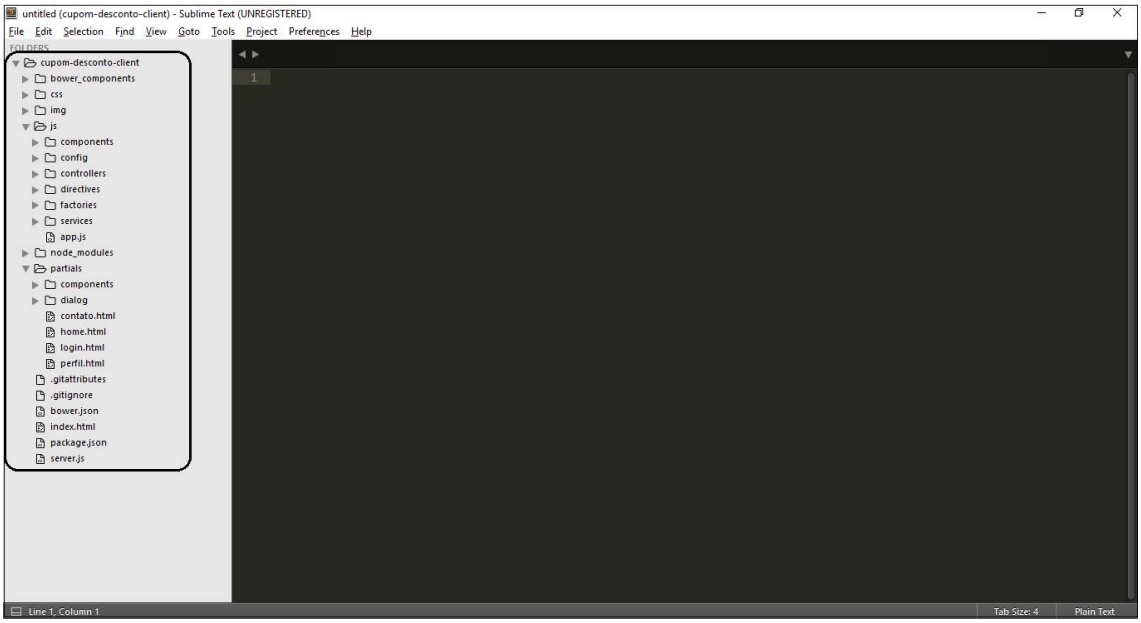

Figura 20 – Estrutura do projeto front-end

Com a utilização do *framework* Spring Boot, a inicialização do sistema é realizada pela classe principal Application.java, que contém o método *main*. O Spring

Boot usa um contêiner embarcado (por padrão é o Tomcat) para facilitar o desenvolvimento, por isso, para iniciar a aplicação, basta executar o método *main* mostrado na Listagem 1.

```
package br.com.api;

import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;
import org.springframework.boot.context.web.SpringBootServletInitializer;

@SpringBootApplication
public class Application extends SpringBootServletInitializer {

    public static void main(String[] args) {
        SpringApplication.run(Application.class, args);
    }
}
```

Listagem 1 – Classe application com o método main

No processo de autenticação, após o usuário ser autorizado, ou seja, informar um usuário e uma senha válidos, o servidor retorna um *JSON Web Token* (JWT) para a aplicação *client* a qual salva localmente essa informação em um *cookie*. Dessa forma, sempre que necessário realizar uma nova requisição, o Token é transmitido juntamente, evitando assim que o servidor solicite o usuário e a senha novamente para cada requisição. A Figura 21 representa essa explicação.

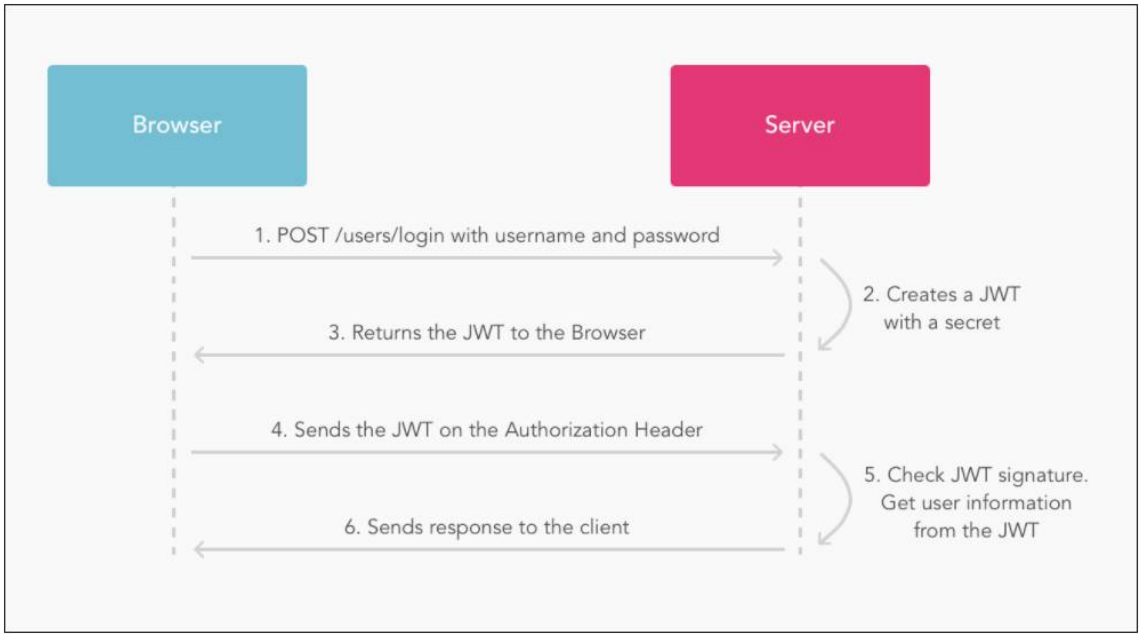

Figura 21 – Autenticação com JSON Web Token
Fonte: Auth0 (2017, p.s.n.).

A Listagem 2 apresenta o código implementado na API REST, o qual realiza a validação do usuário e da senha informados, gerando e retornando o Token para a aplicação *client.*

```
package br.com.api.controller;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.http.ResponseEntity;
import org.springframework.mobile.device.Device;
import org.springframework.security.authentication.AuthenticationManager;
import org.springframework.security.authentication.UsernamePasswordAuthenticationToken;
import org.springframework.security.core.Authentication;
import org.springframework.security.core.AuthenticationException;
import org.springframework.security.core.context.SecurityContextHolder;
import org.springframework.security.core.userdetails.UserDetails;
import org.springframework.security.core.userdetails.UserDetailsService;
import org.springframework.web.bind.annotation.RequestBody;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RequestMethod;
import org.springframework.web.bind.annotation.RestController;

import br.com.api.model.Auth;
import br.com.api.model.Token;
import br.com.api.security.TokenUtils;

@RestController
@RequestMapping("auth")
public class AuthenticationController {
    @Autowired
    private AuthenticationManager authenticationManager;
    @Autowired
    private TokenUtils tokenUtils
    @Autowired
    private UserDetailsService userDetailsService;
    @RequestMapping(method = RequestMethod.POST)
    public ResponseEntity<?> authenticationRequest(@RequestBody Auth auth,
Device device) throws AuthenticationException {
        Authentication authentication =
this.authenticationManager.authenticate(new
UsernamePasswordAuthenticationToken(auth.getUsername(),
auth.getPassword()));

SecurityContextHolder.getContext().setAuthentication(authentication);
        UserDetails userDetails =
this.userDetailsService.loadUserByUsername(auth.getUsername());
        String token = this.tokenUtils.generateToken(userDetails, device);
        return ResponseEntity.ok(new Token(token));
    }
}
```

Listagem 2 – Validação do usuário e geração do token

A aplicação *client* recebe o *token* e o armazena em um *cookie*, enviando essa informação junto com as demais requisições HTTP ao servidor, garantindo a segurança e evitando que a autenticação seja realizada a cada requisição.

A Listagem 3 mostra o código implementado no *controller* LoginController da aplicação *client,* que é responsável por solicitar a autenticação do usuário ao *service* LoginService. Se a autenticação for autorizada pelo servidor, então solicita que o *token* recebido seja salvo no *cookie* do navegador.

```
/**
 * @autor - Joceano Alves de Borba - <alves.joceano@gmail.com>
 * Controller: LoginController, controller da tela de Login.
 * data: 15/07/2017
 **/
angular.module("app.controllers").controller("LoginController",['$scope','Lo
ginService', 'modalService',
      function (scope, loginService, modalService) {
    /**
     * Faz requisição para a API para realizar a autenticação.
     **/
    scope.auth = function (login) {
        loginService.auth(login).then(function (result) {
            loginService.saveToken(result.data.token);
            location.hash = "/";
            scope.$emit('usuarioLogado');
        });
    }
}]);
```

Listagem 3 – Controller responsável pela autenticação do usuário

O *service* LoginService é responsável pela requisição de autenticação para o servidor e por salvar o *token*. Esse código é apresentado na Listagem 4.

```
/**
 * @autor -  Joceano Alves de Borba - <alves.joceano@gmail.com>
 * Service: LoginService, service com os controles de login e logout.
 * data: 15/07/2017
 **/
angular.module("app.services").service("LoginService",
    ['$cookies', 'HttpService', 'toastr',
    function (cookies, httpService, toastr) {

    this.auth = function (user) {
        return httpService.post('/auth', user).then( function (response) {
            return response;
        }, function (error) {
            toastr.warning('Verifique seu usuário e senha.');
        });
    }

    this.saveToken = function (token) {
        saveTokenAux(token);
```

```
    }

    this.getToken = function () {
        cookies.get('X-Auth-Token');
    }

    var saveTokenAux = function (token) {
        cookies.put('X-Auth-Token', token);
    }
}]);
```

Listagem 4 – Service responsável por salvar o token

O *token* sempre é transmitido para o servidor junto com as requisições HTTP. Para facilitar o processo foi criado o *service* HttpService que centraliza as requisições necessárias para a aplicação. Essa implementação é apresentada na Listagem 5.

```
/**
 * @autor -  Joceano Alves de Borba - <alves.joceano@gmail.com>
 * Service: HttpService, service padrão utilizado nas requisições HTTP para
a API.
 * data: 15/07/2017
 **/
angular.module("app.services").service("HttpService",
    ['$http', '$cookies', '$location',
    function (http, cookies, location) {
    var contextPath = "http://localhost:8080/server";

    this.get = function (url) {
        return http({
            method: 'GET',
            url: contextPath + url,
            contentType: "application/json",
            headers: {
                "X-Auth-Token": cookies.get('X-Auth-Token')
            }
        });
    }
    this.post = function (url, obj) {
        return http({
            method: 'POST',
            url: contextPath + url,
            data: JSON.stringify(obj),
            contentType: "application/json",
            headers: {
                "X-Auth-Token": cookies.get('X-Auth-Token')
            }
        });
    }
    this.delete = function (id) {
        return http({
            method: 'DELETE',
            url: contextPath + id,
```

```
            contentType: "application/json",
            headers: {
                "X-Auth-Token": cookies.get('X-Auth-Token')
            }
        });
    }

}]);
```

Listagem 5 – Service responsável pelas requisições HTTP

Antes da criação do mapeamento e das classes de acesso ao banco de dados é necessário criar e configurar o arquivo “pom.xml”. Neste arquivo estão definidas algumas informações do projeto, mas, principalmente, o registro de todas as dependências necessárias. A Listagem 6 apresenta essas configurações.

```
<project xmlns="http://maven.apache.org/POM/4.0.0"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
      xsi:schemaLocation="http://maven.apache.org/POM/4.0.0
http://maven.apache.org/xsd/maven-4.0.0.xsd">
      <modelVersion>4.0.0</modelVersion>

      <!-- Application information -->
      <groupId>br.com.api</groupId>
      <artifactId>server</artifactId>
      <version>1.0</version>
      <name>server</name>
      <packaging>jar</packaging>

      <!-- Properties information -->
      <properties>
            <project.build.sourceEncoding>UTF-
8</project.build.sourceEncoding>
            <project.reporting.outputEncoding>UTF-
8</project.reporting.outputEncoding>
            <java.version>1.8</java.version>
            <jjwt.version>0.4</jjwt.version>
            <jackson.version>2.5.3</jackson.version>
            <apache-commons.version>3.0</apache-commons.version>
            <junit.version>4.12</junit.version>
      </properties>

      <!-- Parent information -->
      <parent>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-starter-parent</artifactId>
            <version>1.3.5.RELEASE</version>
      </parent>

      <!-- Dependency information -->
      <dependencies>
            <dependency>
                  <groupId>org.springframework.boot</groupId>
                  <artifactId>spring-boot-starter-actuator</artifactId>
```

```
		</dependency>
		<dependency>
			<groupId>mysql</groupId>
			<artifactId>mysql-connector-java</artifactId>
		</dependency>
		<!--<dependency>
			<groupId>org.postgresql</groupId>
			<artifactId>postgresql</artifactId>
		</dependency>-->
		<dependency>
			<groupId>org.springframework.boot</groupId>
			<artifactId>spring-boot-starter-data-jpa</artifactId>
		</dependency>
		<dependency>
			<groupId>org.springframework.boot</groupId>
			<artifactId>spring-boot-starter-security</artifactId>
		</dependency>
		<dependency>
			<groupId>org.springframework.boot</groupId>
			<artifactId>spring-boot-starter-test</artifactId>
			<scope>test</scope>
		</dependency>
		<dependency>
			<groupId>org.springframework.boot</groupId>
			<artifactId>spring-boot-starter-web</artifactId>
		</dependency>
		<dependency>
			<groupId>org.springframework.boot</groupId>
			<artifactId>spring-boot-starter-tomcat</artifactId>
		</dependency>
		<dependency>
			<groupId>org.springframework.mobile</groupId>
			<artifactId>spring-mobile-device</artifactId>
		</dependency>
		<dependency>
			<groupId>io.jsonwebtoken</groupId>
			<artifactId>jjwt</artifactId>
			<version>${jjwt.version}</version>
		</dependency>
		<dependency>
			<groupId>com.fasterxml.jackson.core</groupId>
			<artifactId>jackson-databind</artifactId>
			<version>${jackson.version}</version>
		</dependency>
		<dependency>
			<groupId>org.apache.commons</groupId>
			<artifactId>commons-lang3</artifactId>
			<version>${apache-commons.version}</version>
		</dependency>
		<dependency>
			<groupId>junit</groupId>
			<artifactId>junit</artifactId>
			<version>${junit.version}</version>
			<scope>test</scope>
		</dependency>
	</dependencies>
```

```
	<!-- Build information -->
	<build>
		<plugins>
			<plugin>
				<groupId>org.springframework.boot</groupId>
				<artifactId>spring-boot-maven-plugin</artifactId>
			</plugin>
		</plugins>
	</build>
</project>
```

Listagem 6 – Arquivo pom.xml

Outro arquivo de configuração necessário é o *application.properties*. Neste arquivo são definidas as propriedades da base de dados e as variáveis de ambiente utilizadas no projeto. A Listagem 7 apresenta o código do arquivo.

```
spring.jpa.hibernate.ddl-auto:update
spring.jpa.show-sql: true
spring.jpa.properties.hibernate.dialect =
org.hibernate.dialect.MySQL5Dialect
spring.datasource.url=jdbc:mysql://localhost/cupom
spring.datasource.username=root
spring.datasource.password=posjava
spring.datasource.driverClassName=com.mysql.jdbc.Driver

# EMBEDDED SERVER CONFIGURATION
server.contextPath=/server

# JACKSON
spring.jackson.serialization.INDENT_OUTPUT=true

# ERRORS
error.whitelabel.enabled=true

# LOGGING
logging.level.org.springframework.web=DEBUG
logging.level.org.hibernate=DEBUG

# App configuration
api.token.header=X-Auth-Token
api.token.secret=sssshhhh!
api.token.expiration=31536000

# Diretório das Imagens dos Anúncios
diretorio.dirFisico=D:/GitHub/cupom-desconto-client/img/
diretorio.dirBd=/img/
```

Listagem 7 – Arquivo application.properties

No desenvolvimento da API foi necessária a realização do mapeamento entidade relacional. A Listagem 8 representa o mapeamento da tabela Anúncio na classe Anúncio.

```
package br.com.api.model;

import java.util.Date;
import javax.persistence.Column;
import javax.persistence.Entity;
import javax.persistence.FetchType;
import javax.persistence.GeneratedValue;
import javax.persistence.GenerationType;
import javax.persistence.Id;
import javax.persistence.JoinColumn;
import javax.persistence.ManyToOne;
import javax.persistence.Transient;
import javax.validation.constraints.NotNull;
import org.hibernate.validator.constraints.NotEmpty;

@Entity
public class Anuncio {
	@Id
	@GeneratedValue(strategy = GenerationType.IDENTITY)
	private Long id;
	@NotEmpty(message="A descrição não pode ser vazia")
	@Column(nullable = false)
	private String descricao;
	@NotNull
	private Boolean aprovado;
	@NotNull
	private Boolean ativo;
	@NotNull
	private Long quantidade;
	@NotNull
	private Long quantidadeCli;
	@NotNull
	private Date dataInicial;
	@NotNull
	private Date dataFinal;
	@NotNull
	private Date dataCriacao;
	@NotNull
	private Double preco;
	@NotNull
	private Double precoPromo;
	private String detalhe;
	private String regulamento;
	private String diretorio;
	@NotNull
	@ManyToOne(fetch = FetchType.EAGER)
	@JoinColumn(name = "categoria", referencedColumnName = "id")
	private Categoria categoria;
	@NotNull
	@ManyToOne(fetch = FetchType.EAGER)
	@JoinColumn(name = "usuario", referencedColumnName = "id")
```

```
	private Usuario usuario;
	@Transient
	private Integer restante;
	public Integer getRestante() {
		return restante;
	}
	public void setRestante(Integer restante) {
		this.restante = restante;
	}
	public Long getId() {
		return id;
	}
	public void setId(Long id) {
		this.id = id;
	}
	public String getDescricao() {
		return descricao;
	}
	public void setDescricao(String descricao) {
		this.descricao = descricao;
	}
	public Boolean getAprovado() {
		return aprovado;
	}
	public void setAprovado(Boolean aprovado) {
		this.aprovado = aprovado;
	}
	public Boolean getAtivo() {
		return ativo;
	}
	public void setAtivo(Boolean ativo) {
		this.ativo = ativo;
	}
	public Long getQuantidade() {
		return quantidade;
	}
	public void setQuantidade(Long quantidade) {
		this.quantidade = quantidade;
	}
	public Long getQuantidadeCli() {
		return quantidadeCli;
	}
	public void setQuantidadeCli(Long quantidadeCli) {
		this.quantidadeCli = quantidadeCli;
	}
	public Date getDataInicial() {
		return dataInicial;
	}
	public void setDataInicial(Date dataInicial) {
		this.dataInicial = dataInicial;
	}
	public Date getDataFinal() {
		return dataFinal;
	}
	public void setDataFinal(Date dataFinal) {
		this.dataFinal = dataFinal;
```

```
	}
	public Date getDataCriacao() {
		return dataCriacao;
	}
	public void setDataCriacao(Date dataCriacao) {
		this.dataCriacao = dataCriacao;
	}
	public Double getPreco() {
		return preco;
	}
	public void setPreco(Double preco) {
		this.preco = preco;
	}
	public Double getPrecoPromo() {
		return precoPromo;
	}
	public void setPrecoPromo(Double precoPromo) {
		this.precoPromo = precoPromo;
	}
	public String getDetalhe() {
		return detalhe;
	}
	public void setDetalhe(String detalhe) {
		this.detalhe = detalhe;
	}
	public String getRegulamento() {
		return regulamento;
	}
	public void setRegulamento(String regulamento) {
		this.regulamento = regulamento;
	}
	public Categoria getCategoria() {
		return categoria;
	}
	public void setCategoria(Categoria categoria) {
		this.categoria = categoria;
	}
	public Usuario getUsuario() {
		return usuario;
	}
	public void setUsuario(Usuario usuario) {
		this.usuario = usuario;
	}
	public String getDiretorio() {
		return diretorio;
	}
	public void setDiretorio(String diretorio) {
		this.diretorio = diretorio;
	}
}
```

Listagem 8 – Classe anúncio

Com a finalização do mapeamento de todas as entidades foi implementado o acesso aos dados utilizando camadas de serviços e repositórios.

As classes de serviços possuem as validações e as regras de negócios e o que as diferencia das demais classes é a anotação @Service gerenciada pelo Spring, como apresentado na Listagem 9.

```
package br.com.api.service.impl;

import java.io.FileOutputStream;
import java.util.Date;
import java.util.List;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.beans.factory.annotation.Value;
import org.springframework.stereotype.Service;
import org.springframework.web.multipart.MultipartFile;
import br.com.api.model.Anuncio;
import br.com.api.model.Usuario;
import br.com.api.repository.AnuncioRepository;
import br.com.api.service.AnuncioService;
import br.com.api.service.SecurityService;

@Service
public class AnuncioServiceImpl implements AnuncioService{
	@Autowired
	private AnuncioRepository anuncioRepository;
	@Autowired
	private SecurityService securityService;
	@Value("${diretorio.dirFisico}")
	private String diretorioFisico;
	@Value("${diretorio.dirBd}")
	private String diretorioBd;
	@Override
	public Anuncio findOne(Long codigo) {
		return anuncioRepository.findOne(codigo);
	}
	@Override
	public List<Anuncio> findAll() {
		return retornarAnuncios();
	}
	@Override
	public void delete(Long codigo) {
		anuncioRepository.delete(codigo);
	}
	@Override
	public Anuncio save(Anuncio anuncio) {
		if (anuncio.getDataCriacao() == null) {
			anuncio.setDataCriacao(new Date());
		}
		if (anuncio.getUsuario() == null) {
			anuncio.setUsuario(securityService.userLogged());
		}
		return anuncioRepository.save(anuncio);
	}
	@Override
	public void edit(Anuncio anuncio) {
		anuncioRepository.save(anuncio);
	}
```

```
	private List<Anuncio> retornarAnuncios() {
	  Usuario userLogado =  securityService.userLogged();

		if (userLogado != null &&
userLogado.getRoles().equals(Usuario.ROLE_USER)) {
			return anuncioRepository.findByUsuario(userLogado);
		} else {
			return anuncioRepository.findAll();
		}
	}
	@Override
	public void upload(Long codigo, MultipartFile uploadfile) throws
Exception {
		Anuncio anuncio = anuncioRepository.findOne(codigo);
		if (!uploadfile.getOriginalFilename().isEmpty()) {
			String extensao[] =
uploadfile.getOriginalFilename().split("\\.");
			String extencao = extensao[1];
			if (extencao.equalsIgnoreCase("PNG") ||
extencao.equalsIgnoreCase("JPEG") ||
				extencao.equalsIgnoreCase("JPG") ||
extencao.equalsIgnoreCase("BMP")) {
				String nomeArq = codigo + "." + extencao;
				String dirFisico = diretorioFisico + nomeArq;
				String dirBd = diretorioBd + nomeArq;
				FileOutputStream fos = new
FileOutputStream(dirFisico);
				fos.write(uploadfile.getBytes());
				fos.close();
				anuncio.setDiretorio(dirBd);
				anuncioRepository.save(anuncio);
			}
		} else {
			if (anuncio.getDiretorio() == null ||
anuncio.getDiretorio().isEmpty()) {
				anuncio.setDiretorio(diretorioBd + "semImagem.png");
				anuncioRepository.save(anuncio);
			}
		}
	}
}
```

Listagem 9 – Classe de serviço “AnuncioServiceImpl”

Com as validações e as regras de negócios implementadas, é necessário o acesso aos dados. Para isso foram criadas as classes de repositório, as quais possuem a anotação @Repository. A Listagem 10 apresenta essa implementação.

```
package br.com.api.repository;

import java.util.Date;
import java.util.List;
import org.springframework.data.jpa.repository.JpaRepository;
import org.springframework.data.jpa.repository.Query;
```

```
import br.com.api.model.Anuncio;
import br.com.api.model.Usuario;

public interface AnuncioRepository extends JpaRepository<Anuncio, Long>{
	List<Anuncio> findByUsuario(Usuario usuario);
	@Query("select a from Anuncio a where a.aprovado = ?1 and a.ativo = ?2
and "+"?3 between a.dataInicial and a.dataFinal")
	List<Anuncio> findByAnuncioValido(Boolean aprovado, Boolean ativo,
Date dataIni);
}
```

Listagem 10 – Classe de repositório "AnuncioRepository"

Com os serviços e os repositórios implementados foram criados os controladores que compõem a camada responsável por fazer a ligação entre o modelo e a visualização (no caso, entre a API REST e o projeto *front-end*), definir a forma como os dados serão apresentados e encaminhar as ações dos usuários. Um *controller* é identificado pela anotação @RestController. A Listagem 11 contém o código do controlador dos anúncios.

```
package br.com.api.controller;

import java.util.List;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.web.bind.annotation.PathVariable;
import org.springframework.web.bind.annotation.RequestBody;
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.RestController;
import org.springframework.web.multipart.MultipartFile;
import br.com.api.model.Anuncio;
import br.com.api.service.AnuncioService;

@RestController
@RequestMapping("anuncio")
public class AnuncioController {
	@Autowired
	private AnuncioService anuncioService;
	@RequestMapping(value = "/{codigo}", method = RequestMethod.GET)
	public Anuncio findOne(@PathVariable Long codigo){
		return anuncioService.findOne(codigo);
	}
	@RequestMapping(value = "/", method = RequestMethod.GET)
	public List<Anuncio> findAll(){
		return anuncioService.findAll();
	}
	@RequestMapping(value = "/", method = RequestMethod.POST)
	public Anuncio save(@RequestBody Anuncio anuncio){
		return anuncioService.save(anuncio);
	}
	@RequestMapping(value = "/", method = RequestMethod.PUT)
	public void edit(@RequestBody Anuncio anuncio){
```

```
            anuncioService.save(anuncio);
        }
        @RequestMapping(value = "/upload/{codigo}", method =
RequestMethod.POST)
    public void upload(@PathVariable Long codigo,
            @RequestParam("uploadfile") MultipartFile uploadfile) throws
Exception {
            anuncioService.upload(codigo, uploadfile);
        }
}
```

Listagem 11 – Classe de controller "AnuncioController"

A aplicação *client* realiza uma requisição HTTP para a API REST que retorna um objeto no formato JSON. A Listagem 12 apresenta a requisição de categorias realizada no *controller* CategoriaController da aplicação *client.*

```
/**
 * Faz requisição para a API para retornar as categorias cadastradas.
 **/
var getCategorias = function() {
    scope.loading = true;
    httpService.get('/categoria/').then(function(res) {
        scope.categorias = res.data;
        scope.loading = false;
    }, function (error) {
        toastr.error('Não foi possível carregar as categorias.');
    });
};
```

Listagem 12 – Requisição HTTP da lista de categorias

O código implementado para listar as categorias para o usuário está na Listagem 13.

```
<div   layout="row"   layout-sm="column"   layout-align="space-around"   ng-
if="loading">
  <md-progress-circular md-mode="indeterminate"></md-progress-circular>
</div>
<div layout="column" flex layout-fill ng-cloak ng-if="!loading">
  <md-content style="height: 78vh;">
    <section class="back-fff">
      <md-card>
        <md-card-header class="pesquisa">
          <md-input-container class="md-block" id="pesquisa" flex>
            <label><md-icon>search</md-icon> Pesquisar categorias</label>
            <input ng-model="criterioDeBusca" />
          </md-input-container>
          </md-card-header>
          <md-card-content class="content">
            <md-list>
              <md-list-item class="md-2-line"
               ng-repeat="categoria in categorias | filter:criterioDeBusca |
orderBy: 'descricao'"
               ng-click="openDialog($event, categoria)">
```

```
                <ng-avatar initials="{{categoria.descricao}}" max-length="1"
                 auto-color="true" round-shape="true" upper-case="true"
                 class="avatarCadastro"></ng-avatar>
                <div class="md-list-item-text">
                  <h3                                                    ng-
bind="categoria.descricao.substring(0,{{substring}})"></h3>
                </div>
                <md-divider md-inset hide-sm></md-divider>
              </md-list-item>
            </md-list>
          </md-card-content>
      </md-card>
    </section>
  </md-content>
</div>
```

Listagem 13 – Listagem de categorias

Após a listagem das categorias para o usuário, é possível incluir novas categorias ou realizar a manutenção das que já estão salvas no sistema. A listagem 14 exemplifica o código implementado para as manutenções do sistema.

```
<md-dialog aria-label="Cadastro de categorias">
  <form ng-cloak name="formCadastro">
    <md-toolbar>
      <div class="md-toolbar-tools">
        <h2>Cadastro de categorias</h2>
        <span flex></span>
        <md-button class="md-icon-button" ng-click="cancel()">
          <md-icon aria-label="Fechar janela">close</md-icon>
        </md-button>
      </div>
    </md-toolbar>
    <md-dialog-content>
      <div class="md-dialog-content">
        <div class="md-dialog-content">
          <md-input-container class="md-block" flex-gt-sm>
            <label>Descrição</label>
            <input required name="descricao" ng-model="categoria.descricao"
             md-maxlength="80"/>
            <div ng-messages="formCadastro.descricao.$error"
             ng-show="formCadastro.descricao.$invalid">
              <div ng-message="required">A descrição é obrigatória.</div>
            </div>
          </md-input-container>
        </div>
      </div>
    </md-dialog-content>
    <md-dialog-actions layout="row">
      <md-button ng-click="cancel()">Cancelar</md-button>
      <md-button class="md-primary" ng-disabled="formCadastro.$invalid"
       ng-click="save(categoria)">Salvar</md-button>
    </md-dialog-actions>
  </form>
</md-dialog>
```

Listagem 14 – Manutenção de categorias

4 CONSIDERAÇÕES FINAIS

O objetivo do desenvolvimento deste trabalho foi realizar a implementação de um sistema *web* que possibilita a disponibilização de anúncios por empresas de cupons de descontos, a geração e o controle de uso desses cupons. Visando facilidade, agilidade e segurança foi desenvolvido uma interface com o objetivo que tendo em vista simplicidade e intuitividade, no sentido de que o usuário saiba o que está fazendo, o que efetivamente pretende fazer e tendo claro os resultados das suas ações.

Para a implementação do sistema várias tecnologias foram utilizadas, mas o *framework* AngularJS Material chama a atenção pela sua facilidade na configuração e também pelo fato de disponibilizar vários componentes *default*, permitir que o desenvolvedor crie seus próprios componentes com características responsivas. Isso no sentido que o leiaute é flexível e ajusta-se automaticamente ao dispositivo do usuário (*desktop*, *laptop*, celular, *tablet*, entre outros). Além disso, esse *framework* possui uma documentação *online* com exemplos de código, auxiliando o desenvolvedor obter maior agilidade e produtividade. A utilização dos componentes disponibilizados pelo *framework* assemelha-se com a linguagem HTML, ou seja, é baseada em *tags* que representam os seus elementos.

As maiores dificuldades enfrentadas no desenvolvimento do trabalho ocorreram na configuração inicial e na estruturação dos projetos. Incialmente foi criado o projeto *back-end* em Java utilizando a arquitetura do Apache Maven e configurado o arquivo de dependências. Posteriormente ocorreu a parametrização do *framework* Spring com Spring Security e a autenticação de usuário com JWT. Apesar de possuírem uma extensa documentação as situações específicas que ocorrem no decorrer do desenvolvimento geralmente dificultam a localização de soluções. Por fim, foi realizada a configuração e a estruturação do projeto *front-end* utilizando o *framework* AngularJs.

O desenvolvimento do trabalho possibilitou a aquisição de conhecimento em desenvolvimento Java para *web* bem como nas tecnologias que foram utilizadas. Isso foi possível devido à necessidade de pesquisas para obter conhecimento das tecnologias utilizadas como também para encontrar soluções em determinadas

situações específicas. Destaca-se ainda o conhecimento na linguagem Java, JavaScript e CSS.

Como trabalho futuro, pode-se citar a implementação de gráficos e relatórios que possam apresentar de forma clara e objetiva as avaliações realizadas pelos usuários, pois no projeto atual essa informação apenas é armazenada na base de dados.

Considerando o objetivo deste trabalho é possível afirmar que as tecnologias utilizadas possuem um grande impacto nos quesitos de qualidade e produtividade, visto que a implementação do sistema ocorreu em um tempo considerado pequeno para os requisitos do projeto e o resultado atingiu as expectativas e objetivos propostos.

REFERÊNCIAS

ANGULAR. **What is AngularJS Material?** Disponível em: <https://material.angularjs.org/latest/>. Acesso em: 01 jun. 2017.

AUTH0. **Introduction to JSON Web Tokens**. Disponível em: <https://jwt.io/introduction/>. Acesso em: 15 ago. 2017.

CASADIO, Gustavo. Após fim da febre das compras coletivas, Groupon muda modelo de negócio. **Economia**. Disponível em: <http://economia.terra.com.br/apos-fim-da-febre-das-compras-coletivas-groupon-muda-modelo-de-negocio,26638004848c2410VgnVCM3000009af154d0RCRD.html>. Acesso em: 17 set. 2016.

DAROIT, Guilherme. **Cupons de descontos viram mania na internet**. Disponível em: <http://jcrs.uol.com.br/site/noticia.php?codn=131651>. Acesso em: 17 set. 2016.

GALDINO, Fabrício. Primeiros passos com o Angular Material. **DevMedia**. Disponível em: <http://www.devmedia.com.br/primeiros-passos-com-o-angular-material/34537>. Acesso em: 01 jun. 2017.

LOUCOS POR CUPONS. **Lojas com cupom de desconto**. Disponível em: <>. Acesso em: <http://loucosporcupons.com.br/>. Acesso em: 17 set. 2016.

TUTTLE, Brad. **The history of coupons**. 2010. Disponível em: <http://business.time.com/2010/04/06/the-history-of-coupons/>. Acesso em: 17 set. 2016.