CENTRO UNIVERSITÁRIO FACVEST
CURSO DE CIÊNCIA DA COMPUTAÇÃO
TRABALHO DE CONCLUSÃO DE CURSO
QUÉLVIN DE OLIVEIRA RAMOS

CLASSIFICADOS LAGES: Aplicação Web de Classificados

LAGES
2015

QUÉLVIN DE OLIVEIRA RAMOS

CLASSIFICADOS LAGES: Aplicação Web de Classificados

Projeto apresentado à Banca Examinadora do Trabalho de Conclusão do curso de Ciência da Computação para análise e aprovação.

LAGES

2015

QUÉLVIN DE OLIVEIRA RAMOS

CLASSIFICADOS LAGES: Aplicação Web de Classificados

Trabalho de Conclusão do Curso de Ciência da Computação apresentado ao Centro Universitário FACVEST como parte dos requisitos para obtenção do título de bacharel em Ciência da Computação.

Aprovado em: ____/____/____

BANCA EXAMINADORA

_____

Prof. MSc. Márcio José Sembay

_____

Prof. Coordenador do Curso de Ciência da Computação

LAGES

2015

EQUIPE TÉCNICA

Acadêmico

Quélvin de Oliveira Ramos

Professores Orientadores

Profº. Joaquim Rodrigo de Oliveira, Msc.

Profº. Márcio José Sembay, Msc.

Coordenador de TCC

Profº. Márcio José Sembay, Msc.

Coordenador do Curso

Profº. Márcio José Sembay, Msc.

Dedico este trabalho a Beatriz Luz e Alice Luz Ramos os dois amores da minha vida. Que fazem todo o esforço e trabalho duro valerem à pena.

AGRADECIMENTOS

Aos meu pais, Dionel e Lucia, que com muito amor e dedicação, me apoiaram para que eu pudesse ter uma educação de qualidade;

A minha noiva, Beatriz, pelo incentivo para que este trabalho pudesse ser realizado;

A minha filha, Alice, que embora ainda não compreenda, é o meu maior incentivo para me tornar uma pessoa melhor;

Aos professores que estiveram presentes durante toda essa etapa e que me ajudaram para que ela pudesse ser concluída, em especial aos professores Márcio José Sembay e Joaquim Rodrigo de Oliveira.

RESUMO

Os sites de classificados são uma tendência atual no Brasil, e são amplamente utilizados para fechamento de negócios diariamente. Em Lages-SC, embora existam sistemas que atendam a cidade, é necessário um site completo que possua uma busca efetiva, várias categorias de anúncios e tenha um bom funcionamento em dispositivos móveis. O presente trabalho de conclusão de curso teve como objetivo desenvolver um sistema para suprir as necessidades da cidade de Lages-SC no que tange a classificados.

Palavras-chave: Classificados, Lages-SC, Anúncios.

ABSTRACT

The classified sites are a current trend in Brazil, and are widely used for closing deals daily. In Lages-SC, although there are systems that meet the city, a comprehensive website needs to have an effective search, several categories of ads and is mobile-friendly. This course conclusion work aimed to develop a system to meet the Lages-SC city's needs with respect to classified.

Keywords: Classifieds, Lages-SC, Ads.

RESUMEN

Los sitios clasificados son una tendencia actual en Brasil, y son ampliamente utilizados para cerrar tratos diaria. En Lages-SC, aunque hay sistemas que cumplan con la ciudad, un completo sitio web tiene que tener una búsqueda efectiva, la mayoría de categorías de los anuncios y obtener un buen centro en los dispositivos móviles. Este trabajo conclusión curso dirigido a desarrollar un sistema para satisfacer las necesidades de la ciudad de Lages-SC con respecto al anuncio.

Palabras clave: Clasificados, Lages-SC, Anuncios.

LISTA DE ABREVIATURAS

AJAX	Asynchronous Javascript and XML
API	Application Programming Interface
CSS	Cascading Style Sheets
DER	Diagramas Entidade Relacionamento
HTML	HyperText Markup Language
HTTP	HyperText Transfer Protocol
JSON	Javascript Object Notation
MVC	Model-View-Controller
ORM	Object-Relational Mapping
PHP	PHP: Hypertext Preprocessor
REST	Representational State Transfer
SGBD	Sistema de Gerenciamento de Banco de Dados
SQL	Structured Query Language
TCC	Trabalho de Conclusão de Curso
UML	Unified Modeling Language
URL	Uniform Resource Locator
XML	Extensible Markup Language

LISTA DE ILUSTRAÇÕES

SUMÁRIO

1 INTRODUÇÃO

As empresas e a população regional necessitam de constante aperfeiçoamento no que tange a meios de realizar negócios e procura de novas oportunidades. Desta forma, a tecnologia da informação contribui efetivamente para que esta problemática possa ser resolvida ou amenizada.

Com a popularização da Internet criou-se um amplo campo para o surgimento de novas oportunidades de negócios e relações. Para suprir a essas novas necessidades, tem sido desenvolvido cada vez mais programas que tem como meio a Internet.

Pode-se considerar a Internet, como um campo ainda novo e propício ao surgimento de novas ideias, criação de programas que abrem por sua vez, novos campos.

Segundo Sommerville (2007, p. 4) “muitas pessoas associam o termo *software* aos programas de computador. Na verdade, essa é uma visão muito restritiva. *Software* não é apenas o programa, mas também todos os dados de documentação e configuração associadas, necessários para que o programa opere corretamente”. Para o correto desenvolvimento de *software* não se deve pensar apenas na programação, mas também em uma boa documentação, facilitando assim o planejamento e manutenção.

O desenvolvimento desse trabalho teve como objetivo criar uma aplicação Web que centralize os negócios da cidade de Lages, Santa Catarina, permitindo que empresas e a população geral criem anúncios de classificados (compra e venda, veículos e imóveis), vagas de empregos e anunciem serviços.

Essa aplicação Web visa centralizar para que os anúncios atinjam de uma forma mais rápida o seu público alvo.

Para o *front-end* dessa aplicação foi utilizado HTML 5, CSS 3 e JavaScript. No *back-end* foi utilizado PHP (juntamente com o *framework* Laravel 5) e banco de dados MySQL.

Sendo assim, teve-se como problemática desse projeto: como desenvolver um sistema web de classificados para Lages-SC, e que supra as necessidades da população local?

1.1 Justificativa

Pressman (2011, p. 31) destaca que “o software assume um duplo papel. Ele é um produto e, ao mesmo tempo, o veículo para distribuir um produto”. Então criar um sistema que seja utilizado para auxiliar na venda de produtos é uma ótima forma de contribuir com a população local.

O Classificados Lages, tem como objetivo ser acessível a qualquer pessoa que tenha interesse em anunciar, contando com um plano gratuito. Além disso abrange várias outras categorias possibilitando um maior alcance e valorização como referência em anúncios. ,

1.2 Importância

Desenvolver meios tecnológicos que sejam de utilidade pública, devem ser uma das prioridades dos cientistas da computação. Pois o conhecimento adquirido deve ser utilizado para que a melhoria da população seja um dos focos principais de trabalho.

Atualmente já existem serviços semelhante, mas que atuam em apenas uma parte da problemática, e ainda assim são genéricos em sua proposta, pretendendo atingir um público maior, o que faz com que anúncios da cidade de Lages acabem sendo encontrados em diversos locais diferentes. Ao criar uma aplicação em que o público destinado esteja fortemente definido, pode se tornar uma referência e o principal local acesso para os interessados.

Ao criar um sistema Web que atraia a atenção da população local, pode reforçar a ideia de que existe um mercado mal explorado localmente e que o investimento em tecnologias que utilizem a internet pode dar um retorno tanto social, quanto financeiro.

1.3 Objetivo do Trabalho

Ao buscar meios que viabilizem desenvolver o projeto deste trabalho, foi determinado um objetivo geral, que por sua vez dependem de objetivos específicos para tornar factível o resultado final.

1.3.1 Objetivo Geral

Desenvolver um sistema de informação de classificados para a cidade de Lages-SC e região, para que seja um ponto de centralização de anúncios, de forma organizada e de fácil acesso pela população.

1.3.2 Objetivos Específicos

a) Descrever ferramentas e tecnologias que foram utilizadas, demonstrando de que forma foram utilizadas;

b) Criar e revisar, a documentação de projeto de *software*;
c) Desenvolver um sistema web de publicação de classificados com cinco categorias principais (Compra e Venda, Veículos, Imóveis, Serviços e Empregos);
d) Desenvolver um sistema de fácil administração utilizando-se de boas práticas;
e) Disponibilizar a aplicação web totalmente funcional e dentro do escopo pré-determinado.

2 REFERENCIAL TEÓRICO

2.1 Sistemas Correlatos

Neste tópico estão descritos sites de classificados e meios de anúncios, como grupos no Facebook.

2.1.1 OLX

O OLX se auto define como o maior site de classificados grátis do Brasil. No início de 2015 juntou-se ao Bom Negócio que antes era seu maior concorrente no Brasil.

O site ALEXA o coloca como o 21º site mais popular do Brasil (http://www.alexa.com/topsites/countries/BR).

São diversas categorias com uma enorme quantidade de produtos e serviços. O grande problema é que o público que atinge em cidades pequena e médias não é o suficiente para que os anúncios consigam uma boa divulgação.

2.1.2 Lages Imóveis

O site é o maior de Lages na divulgação de venda e aluguel de imóveis. Desenvolvido pelo Grupo MSR.

Possui apenas planos pagos, mas atinge um bom público. Além disso a pesquisa de anúncios é satisfatória.

2.1.3 Ponto do Negócio

O site é o maior de Lages na divulgação de venda de veículos. Assim como o Lages Imóveis também é desenvolvido pelo Grupo MSR.

Como tem uma base semelhante ao Lages Imóveis também possui apenas planos pagos. É uma referência em Lages na venda de veículos, por ter uma forte parceria com revendas de veículos promove anualmente um “Feirão” com diversos veículos.

2.1.4 Grupos no Facebook

O Facebook revolucionou a comunicação. Ele é muito utilizado na promoção de produtos e empresas.

Os grupos no Facebook são bons para classificados, o fator social facilita a comunicação, o maior problema é a desorganização, não há categorias de anúncios.

A divulgação dos anúncios é ótima, principalmente por conta das notificações.

2.2 HTML 5

HTML é uma abreviação de Hypertext Markup Language.

Refere-se ao conceito de linguagem de marcação de hipertexto do qual são conjuntos de elementos interligados que podem ser vídeos, imagens, áudio, conteúdo de texto, etc. O que permite a organização dos dados relacionados.

Um sistema feito baseado em HTML pode ser acessado apenas estando conectado à internet de qualquer dispositivo através um navegador moderno.

O Working Group W3C foi o responsável por manter o padrão do código por décadas e focava-se na segunda versão do XHTML (HTML com XML) em paralelo surgiu depois, em meados de 2004 o WHATWG um grupo livre para a criação do novo HTML5 como uma opção mais flexível para a criação de websites e sistemas web.

O HTML5 é inovador porque modifica a maneira como escrevemos o código e organizamos o conteúdo na página. Conforme destaca EIS (2012, p. 30):

> O HTML5 modifica a forma de como escrevemos código e organizamos a informação na página. Seria mais semântica com menos código. Seria mais interatividade sem a necessidade de instalação de plugins e perda de performance. É a criação de código interoperável, pronto para futuros dispositivos e que facilita a reutilização da informação de diversas formas.

2.3 CSS 3

CSS é a abreviação de Cascading Style Sheets – Folha de Estilo em Cascata.

Uma folha CSS basicamente é um conjunto organizado de informações sobre a formatação e exibição do conteúdo do layout de uma página.

O CSS é um código separado responsável pelas alterações de apresentação da página.

Possui três tipos de regras, do qual, uma regra CSS basicamente é formada de três partes: o seletor identificador da regra, a propriedade que informa o que está sendo definido e os valores

que modificam as propriedades.

O CSS pode ser incluído diretamente na tag e devem estar sempre entre aspas (" "). Na página web que permite a inclusão de informações a todo documento ou vinculados ao arquivo XHTML (arquivo externo), através das tags específicas <link> ou @import.

EIS (2012, p. 21) simplifica o conceito de CSS:

> O CSS é a linguagem responsável por controlar o visual da informação exibida pelo HTML. O CSS formatará o conteúdo de forma que seja visualmente agradável em qualquer meio de acesso. A informação é acessada por diferentes meios de acesso, desde sistemas de busca até aparelhos como tablets, smartphones etc e o CSS é o responsável por formatar a informação para que ela seja consumida em qualquer meio de **acesso** de forma simples.

2.4 Bootstrap

Mazza (2012, p. 182) define o Bootstrap como um "toolkit criado por designers do Twitter e adotado por diversas empresas e equipes para acelerar o início de projetos por já incluir diversos componentes e facilidades de CSS, HTML e JavaScript".

O Bootstrap facilita a tarefa de design e CSS, pois as padronizações criadas facilitam o momento da criação das páginas Web, no que diz respeito a *front-end*.

2.5 Javascript

HTML e CSS não são linguagens de programação. Para resolver determinados problemas, é necessário utilizar uma linguagem de programação.

Javascript é a linguagem de programação mais popular no desenvolvimento Web. Suportada por todos os navegadores, a linguagem é responsável por praticamente qualquer tipo de dinamismo nas páginas Web (K19 TREINAMENTOS, 2013).

Com a linguagem Javascript pode-se construir páginas extremamente dinâmicas e interativas, já que seu foco é implementar o comportamento ou a inteligência.

Ao utilizar-se de todo o potencial da linguagem, pode-se chegar a grandes resultados, como por exemplo: Facebook, Google Maps, Google Docs e YouTube.

A seguir um exemplo de um código Javascript simples, ao acionar o botão, é criada uma caixa de diálogo com a mensagem "Alerta!".

```
<!DOCTYPE html>
<html>
<head>
<title>Exemplo básico Javascript</title>
<script>
function funcaoAlerta()
{
alert("Alerta!");
}
</script>
</head>
<body>

<input type="button" onclick="funcaoAlerta()" value="Exibir Alerta" />

</body>
</html>
```

Figura 1 - Exemplo em Javascript
Fonte: Próprio autor.

2.5.1 JQuery

JQuery, segundo Balduino (2012, p. 33) é "uma biblioteca Javascript que simplifica a manipulação de documentos HTML, eventos, animações e interações com AJAX para desenvolvimento rápido de aplicações web".

O JQuery possui ampla utilização devido a ser flexível e simples (se comparado ao Javascript puro).

Uma das vantagens na utilização de uma biblioteca Javascript é expandir a compatibilidade de um mesmo código para diversos *browsers*. Utilizando-se da criação de funções que verificam as características necessárias, permitindo assim um código universal para os mais diversos navegadores.

Além dessa vantagem, o JQuery, que é hoje a biblioteca padrão na programação *front-end* para Web, traz uma sintaxe mais "fluida" nas tarefas mais comuns ao programador que são: selecionar um elemento do documento e alterar suas características.

2.6 JSON

JSON significa Javascript *Object Notation*. Tem esse nome, pois o Javascript foi a primeira linguagem a utilizar esse método.

É um método mais humano para armazenar *arrays* e objetos com valores como *strings*. Tem como seu principal uso as transferências de grandes quantidades de dados, devido a possuir um tamanho mais compacto se comparado com XML, por exemplo.

Geralmente é usado quando o *front-end* da aplicação precisa de alguma informação do *back-end* sem ter que recarregar a página. Isso é normalmente feito usando Javascript e uma requisição AJAX, segundo Balduíno (2012, p. 78) “Usar AJAX com JSON é algo muito simples com jQuery”.

2.7 AJAX

A interação entre usuários e páginas web é limitada. Podem ser tratadas duas abordagens para essa interação, conforme descrito a seguir.

Quando o usuário acessa uma página, através de um *link* ou botão por exemplo, é realizada uma requisição HTTP ao servidor, o mesmo processa, e envia uma resposta contendo uma página inteira, que no lado cliente deverá ser carregada. Em certas situações essa abordagem pode ser utilizada, mas com algumas ressalvas já que ocorre um desperdício de transferência de informações desnecessárias, e tempo nesse novo carregamento (K19 TREINAMENTOS, 2013).

Mas para resolver essa questão pode-se utilizar uma segunda abordagem, chamada de AJAX. Através dessa tecnologia é possível que o navegador envie uma requisição através de Javascript, com um XML ou JSON, informando apenas a ação que deseja realizar e informando somente o necessário para que o processamento ocorra conforme o esperado. Após o processamento no servidor, o mesmo retorna somente o necessário, assim como o envio (em XML ou JSON), e essa informação é utilizada para atualizar a página pré-carregada no navegador cliente.

Conforme destacado por Mclaughlin (2006, p. 7), “Não há nada mais incômodo do que um aplicativo que redesenha a página inteira sempre que você pressiona um botão ou digita um valor”. Sendo assim ao utilizar a segunda abordagem demonstrada, pode-se diminuir o “atrito” com usuário que uma página que carrega a cada requisição provoca.

2.8 PHP

PHP é uma linguagem de programação criada em 1994 por Rasmus Lerdof.

Ele criou um conjunto de scripts voltados à criação de páginas dinâmicas (princípio básico do PHP) para monitorar o acesso ao seu currículo na Internet. Com o crescimento da ferramenta, Rasmus teve de escrever uma implementação em C, permitindo as pessoas desenvolverem de forma mais simples suas aplicações para a web.

Segundo o The PHP Group (2015), PHP é uma linguagem de programação de código aberto, amplamente utilizada, que tem como sua principal função a de criação de scripts. Esses scripts são executados no servidor web para a manipulação de página HTML, ou seja, utiliza-se PHP para criar páginas dinâmicas e automáticas. Essa linguagem é diferente de outras linguagens, onde o código é interpretado e não compilado.

A seguir um exemplo de um código em PHP. Nele é inserido dentro do corpo do HTML a frase: “Olá. Eu sou um script PHP!”, através de um comando “echo”, muito utilizado na linguagem:

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
    "http://www.w3.org/TR/html4/loose.dtd">
<html>
    <head>
        <title>Exemplo</title>
    </head>
    <body>

        <?php
        echo "Olá, eu sou um script PHP!";
        ?>

    </body>
</html>
```

Figura 2 - Exemplo em PHP
Fonte: The PHP Group (2015)

A sintaxe da linguagem se assemelha a outras linguagens orientadas a objeto como C# e Java. O objetivo principal da linguagem é permitir a desenvolvedores escreverem páginas que serão geradas dinamicamente de forma rápida, permitindo aplicações avançadas utilizando conexões com os SGBDs mais comuns do mercado atualmente, como o MySQL, Oracle e PostgreSQL.

Fayad e Schmidt (1997) descreve *framework* como uma estrutura formada por partes pré-fabricadas de *software* que os programadores podem usar, estender ou adaptar para uma solução específica. O PHP possui vários *frameworks* amplamente utilizado pelo mercado.

Skvorc (2015) realizou uma pesquisa sobre os *frameworks* PHP mais populares, o “Laravel”, foi o que obteve mais votos, conforme a figura a seguir.

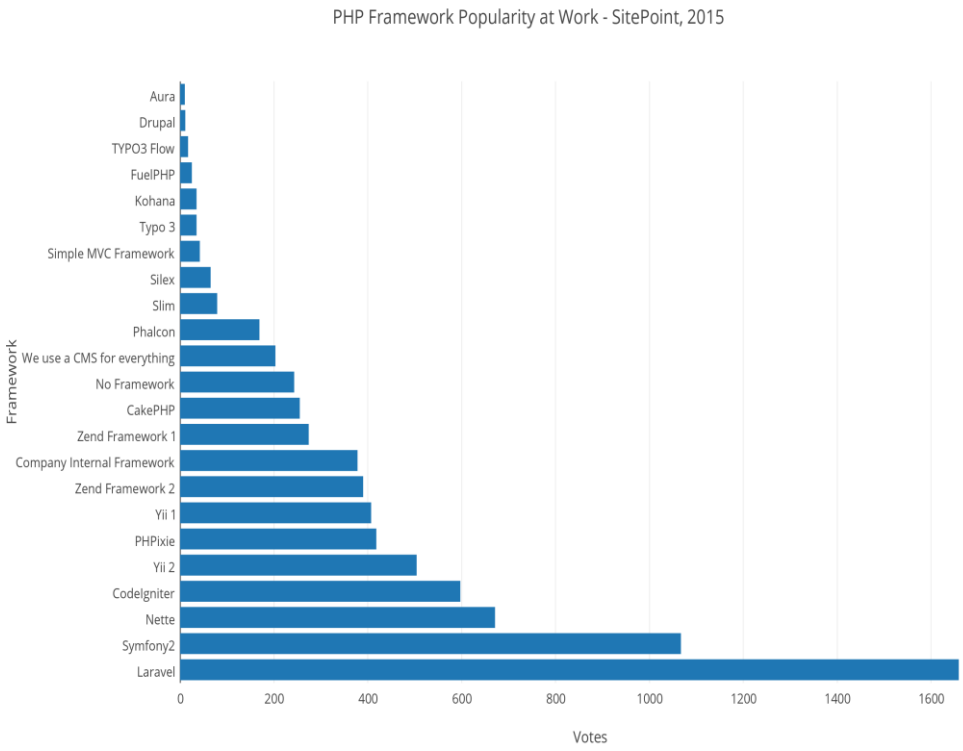

Figura 3 - Frameworks PHP mais populares em 2015
Fonte: Skvorc (2015)

Escolher um *framework* de acordo com sua popularidade pode ser um bom parâmetro para a escolha. Se existe uma boa avaliação do mesmo, a comunidade que o mantém pode fornecer e produzir uma boa quantidade de material (artigos, matérias e livros) que servirão como ponto de partida, para o desenvolvimento de um sistema que o utilize como base. Além disso ele utiliza a arquitetura MVC (model-view-controller), muito utilizada nas mais diversas linguagens de programação.

2.9 Arquitetura MVC

Para Buschmann et al. (1996, p. 123), a "arquitetura *Model-View-Controller* (MVC) divide uma aplicação interativa em três componentes". O *Model* (modelo) contém o núcleo da aplicação e os dados. A *View* (visualização) exibe informações para o usuário. O *Controller* (controlador) manipula a entrada do usuário. As *Views* e *Controllers* juntos formam a interface do usuário.

A seguir uma figura que demonstra o fluxo básico do MVC.

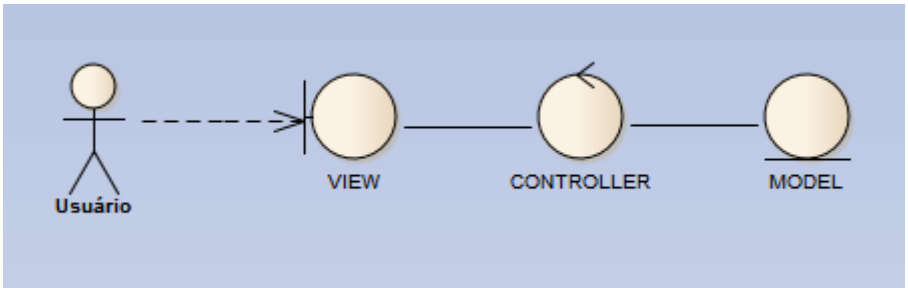

Figura 4 - Fluxo básico MVC
Fonte: Próprio autor.

2.10 Laravel Framework

Laravel é um *framework* PHP de aplicações web de código aberto, baseado na arquitetura MVC. É um dos mais populares entre os desenvolvedores PHP.

Ele facilita várias tarefas rotineiras utilizados na maioria dos projetos de web, como autenticação, roteamento, sessões, filas e cache. Para Otwell (2015), ele "fornece ferramentas poderosas necessárias para aplicativos grandes e robustos".

A última versão lançada do *framework* é a 5.0, que foi utilizada no desenvolvimento desse trabalho de conclusão de curso.

2.10.1 Artisan

Artisan é o nome da interface de linha de comando incluída no Laravel. Ela fornece uma série de comandos úteis para a utilização durante o desenvolvimento.

Conforme descreve Gilmore (2015, p. 14), "Artisan é uma interface comando, usada para desenvolver rapidamente novas partes das aplicações, tais como controladores e a gerência evolutiva seu banco de dados por meio de um grande recurso conhecido como *migration*".

Na figura a seguir é demonstrado um exemplo do comando utilizado para criar um *controller*, uma tarefa que pode ser realizada pelo Artisan.

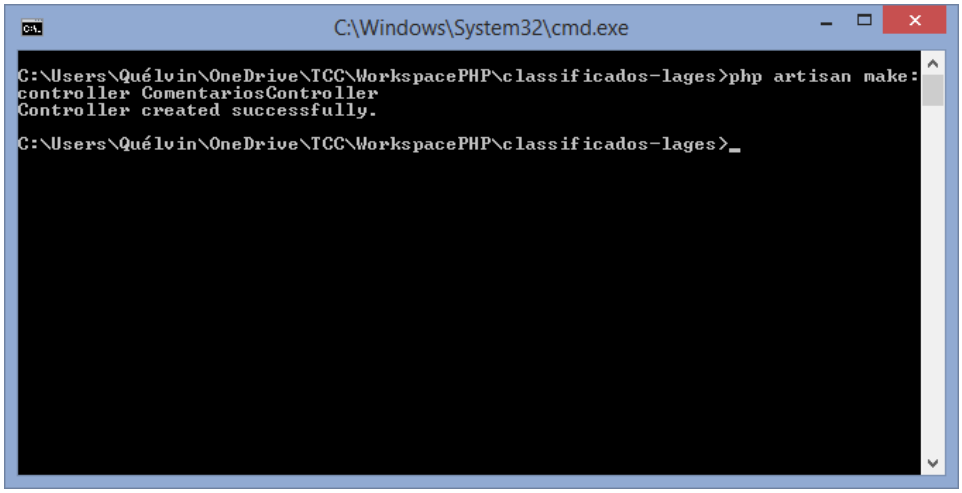

Figura 5 - Artisan exemplo
Fonte: Próprio autor.

Após a execução do comando o mesmo é criado contendo métodos vazios, com o nome dentro das “boas práticas” do Laravel, conforme a figura a seguir.

```php
<?php namespace App\Http\Controllers;
use App\Http\Requests;
use App\Http\Controllers\Controller;
use Illuminate\Http\Request;

class ComentariosController extends Controller {
    public function index()
    {
        //
    }
    public function create()
    {
        //
    }
    public function store()
    {
        //
    }
    public function show($id)
    {
        //
    }
    public function edit($id)
    {
        //
    }
    public function update($id)
    {
        //
    }
    public function destroy($id)
    {
        //
    }
}
```

Figura 6 - Controller criado através do Artisan
Fonte: Próprio autor.

2.10.2 Eloquent

Eloquent é o nome do ORM presente no framework Laravel. Sua principal função é abstrair os dados de um modelo.

Usar o Eloquent para representar objetos do banco de dados facilita várias tarefas rotineiras que necessitam da utilização do banco de dados.

No exemplo a seguir, a classe “Categoria” é uma extensão de “Model”, que por sua vez faz parte do Eloquent.

```php
<?php namespace App;

use Illuminate\Database\Eloquent\Model;

class Categoria extends Model {

    protected $table = 'categorias';

    protected $fillable = [
        'id',
        'nome',
        'descricao'
    ];

    public function getSubcategoriasListAttribute()
    {
        return $this->subcategorias->lists('id');

    }

    public function subcategorias()
    {
        return $this->hasMany('App\Subcategoria');
    }

}
```

Figura 7 - Eloquent Model
Fonte: Próprio autor.

A classe “Categoria” possui a variável local “*table”*, nela está armazenada o nome da tabela a qual ela se refere. No array “*fillable”*, estão armazenados os nomes das colunas da tabela “categorias” que podem ser manipuladas. O método “subcategorias” possui a declaração de relacionamento de tabelas, nesse caso o “hasMany”, significa que a classe “Categoria” pode ter várias subcategorias relacionadas.

2.10.3 Blade (View)

O Blade é o sistema de *templates* que acompanha o Laravel.

Um sistema de *templates*, basicamente permite criar *views*, não sendo necessário utilizar diretamente *tags* PHP.

Para criar uma *view* que utilize a sintaxe do Blade, o arquivo deve possuir a extensão ".blade.php", isto indica ao Laravel que serão utilizada estruturas do Blade.

O exemplo a seguir foi criado para demonstrar algumas sintaxes que podem ser utilizadas em neste sistema de *templates*.

```
1  @extends('app')
2
3  @section('content')
4      <h1>{{$anuncio->titulo}}</h1>
5
6      <article>
7          {{$anuncio->descricao}}
8      </article>
9
10     @unless ($anuncio->tags->isEmpty())
11         <h5>Tags:</h5>
12         <ul>
13
14             @foreach($anuncio->tags as $tag )
15                 <li>{{ $tag->name }}</li>
16
17             @endforeach
18         </ul>
19     @endunless
20
21 @stop
```

Figura 8 - Exemplo Blade
Fonte: Próprio autor.

O comando "extends" permite estender um *layout* básico previamente criado e com itens visuais que podem ser reaproveitados, dessa forma o comando "section" é utilizado para sobrescrever uma seção declarada no layout estendido.

2.10.4 Controller

No *controller* é armazenada a lógica de uma rota. As ações contidas dentro dele estão diretamente relacionadas com o processamento de requisições e respostas para elas.

Dias (2014) define o *controller* como sendo a "[...] central lógica da nossa aplicação. É no *controller* que processamos a entrada do usuário e executamos nossos models".

A seguir um exemplo de um controller:

```php
<?php namespace App\Http\Controllers;

use App\Http\Requests;
use App\Http\Controllers\Controller;
use Illuminate\Http\Request;
use App\Subcategoria;
use App\Http\Controllers\Response;

class SubcategoriasController extends Controller {

    public function show(Request $request)
    {
        $option = $request->input('option');

        $subcategorias = Subcategoria::where('categoria_id', '=', $option)->get();

        return response()->json($subcategorias, 200);
    }

}
```

Figura 9 - Exemplo de controller
Fonte: Próprio autor.

Neste exemplo está o desenvolvimento de um *controller* responsável por processar uma requisição “ajax” que informa o identificador do banco de dados de uma “categoria” e retorna a lista de “subcategorias” que pertencem a ela, no formato “json”.

2.10.5 Rotas

Segundo Rees (2014), “as rotas em Laravel, são as especificações, que definem qual *controller*, e qual método serão executados de acordo com a URL acessada pelo usuário”. O arquivo onde são feitas estas definições dentro da estrutura de diretórios do *framework* é o “app/Http/routes.php”. A figura a seguir possui um exemplo de um arquivo de rotas.

```php
<?php
Route::get('/', function()
{
    return 'Teste';
});
```

Figura 10 - Exemplo de rota em Laravel
Fonte: Próprio autor.

A rota definida no exemplo acima especifica que ao acessar o endereço raiz da aplicação web, será mostrada uma mensagem “Teste” na página retornada. Ele não chama um *controller* específico por se tratar de um exemplo, mas na prática um *controller* deveria ser chamado, e nele estaria contido a lógica necessária para o processamento da requisição realizada pelo usuário. Não é uma boa prática inserir lógica de processamento no arquivo de rotas.

2.10.6 Composer

Para Belem (2012), “O Composer nada mais é do que um gerenciador de dependências… Com ele você define a lista de bibliotecas (e versões) das quais o seu projeto depende, e ele cuida da instalação, organização e “inclusão” das mesmas”.

Composer é uma ferramenta para gerenciamento de dependência em PHP, muito utilizada em Laravel. Ela permite a utilização de bibliotecas necessárias ao projeto e instala automaticamente no projeto (COMPOSER, 2015).

Para adicionar uma dependência a algum projeto através do Composer, basta alterar o arquivo “composer.json”. A figura a seguir possui um trecho de código de exemplo para adicionar a dependência a REST API do PayPal.

```
1  {
2      "require": {
3          "paypal/rest-api-sdk-php": "1.3.2"
4      }
5  }
```

Figura 11 - Exemplo dependência do Composer
Fonte: Próprio autor.

2.11 MySQL

Um banco de dados basicamente constitui-se por uma coleção de dados em uma estrutura. Pode ser utilizado para armazenar uma lista de texto simples, ou uma grande quantidade de dados complexos, como utilizado em alguns sistemas especialistas médicos, por exemplo.

Para adicionar, acessar e processar dados armazenados em um banco de dados de computador, é necessário um sistema de gerenciamento de banco de dados, como “MySQL Server”. Segundo o site oficial do MYSQL (2015), “Os computadores são muito bons em lidar com grandes quantidades de dados, sistemas de gerenciamento de banco de dados desempenham um papel central na computação, seja como utilitários independentes ou como partes de outras aplicações”.

MySQL é um dos sistemas de gerenciamento de banco de dados SQL *Open Source* mais populares do mundo, amplamente utilizado em PHP.

Ferrari (2010, p. 6) no trecho a seguir descreve um pouco sobre a história do MySQL:

> O MySQL, nascido em meados dos anos 1980, foi criado na Suécia, por apenas três desenvolvedores e visionários: Allan Larsson, David Axmark e Michael Widenius, membro fundador da empresa MySQL AB, a qual foi criada para ser responsável pelo banco de dados e seus suportes.

Atualmente o MySQL e seus códigos fontes foram adquiridos pela gigante SUN que posteriormente foi comprada pela Oracle, proprietária do banco de dados Oracle, e possui por volta de 500 desenvolvedores oficiais trabalhando em suas constantes melhorias.

3 METODOLOGIA

A figura a seguir mostra as etapas do desenvolvimento deste Trabalho de Conclusão de Curso:

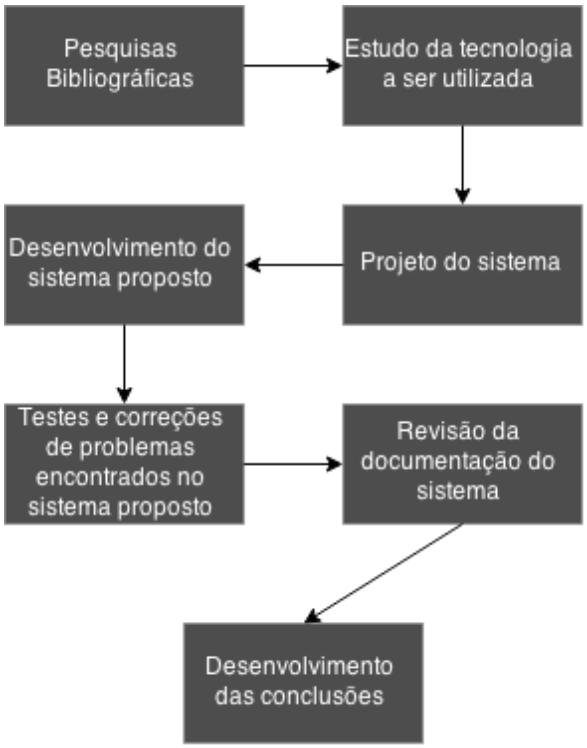

Figura 12 - Etapas do desenvolvimento
Fonte: Próprio autor.

Este trabalho realizou uma pesquisa analisando as tecnologias disponíveis para o desenvolvimento, conforme a proposta do trabalho.

3.1 Pesquisa Bibliográfica

Para a realização deste trabalho foram realizadas pesquisas em livros, periódicos e sites da Internet, os conceitos e tecnologias para o desenvolvimento de aplicações Web, com o objetivo de possuir uma base teórica que sustentasse o desenvolvimento do sistema, seguindo ideias base descritas por Appolinário (2011).

3.2 Desenvolvimento do Sistema

A partir da pesquisa e do estudo bibliográfico, foi possível determinar as tecnologias que mais se adaptavam ao necessário para cumprir o objetivo do trabalho.

Com a escolha das tecnologias, foi possível iniciar o desenvolvimento do sistema com a linguagem de programação determinada e com as ferramentas mais adequadas.

Durante o desenvolvimento foram elaborados diagramas de funcionamento do sistema.

Durante todo o ciclo de desenvolvimento foram realizados, para que no final fosse gerado um sistema de qualidade, funcional e que atendesse aos requisitos necessários para o público alvo.

3.3 Cronograma

O quadro a seguir mostra o cronograma final do Trabalho de Conclusão de Curso.

Cronograma											
Atividade	**Fev**	**Mar**	**Abr**	**Mai**	**Jun**	**Jul**	**Ago**	**Set**	**Out**	**Nov**	**Dez**
Pesquisas	X	X	X	X							
Desenvolvimento e apresentação da pré-proposta		X	X								
Modelagem do sistema			X	X							
Artigo			X	X							
Publicação do Artigo					X						
Defesa de Banca TCC 1						X					
Revisão na modelagem do sistema							X				
Implementação do sistema							X	X	X	X	
Testes no sistema							X	X	X	X	
Artigo final									X		
Publicação do artigo final										X	
Entrega do TCC										X	
Defesa do TCC à banca avaliadora											X

Quadro 1 - Cronograma do Trabalho

Fonte: Próprio autor.

4 PROJETO

A seguir estão os detalhes do projeto do sistema desenvolvido.

4.1 Introdução

O projeto consistiu no desenvolvimento de um sistema web de classificados para a cidade de Lages, Santa Catarina. Por se tratar de um sistema web, os usuários e o administrador poderão acessar o sistema de qualquer local com acesso à internet, e de qualquer dispositivo que possua um browser moderno, já que foram utilizadas técnicas de *Web Design* responsivo.

4.2 Análise de Requisitos

Os requisitos segundo Larman (2005, p. 81) “são capacidades e condições às quais o sistema deve atender”. Para o desenvolvimento foi realizado o levantamento de requisitos funcionais, não funcionais e as regras de negócios, que orientarão durante a fase de implementação do sistema.

A seguir serão mostrados os requisitos levantados para o desenvolvimento do sistema.

4.2.1 Requisitos Funcionais

Os requisitos funcionais descrevem as funcionalidades e serviços do sistema. A seguir estão enumerados os requisitos funcionais:

a) RF01: O sistema deve permitir que o usuário se cadastre e mantenhas os dados informados;
b) RF02: O sistema deve permitir que o administrador gerencie os usuários cadastrados;
c) RF03: O sistema deve permitir a publicação e manutenção dos anúncios pelos usuários;
d) RF04: O sistema deve permitir que o administrador gerencie os anúncios;
e) RF05: O sistema deve permitir a busca de anúncios pelos usuários (autenticados e não autenticados);
f) RF06: O sistema deve permitir a visualização detalhada de um anúncio pelos usuários (autenticados e não autenticados);

g) RF07: O sistema deve permitir que o usuário faça avalição em anúncios de Serviços;
h) RF08: O sistema deve exigir o *login* e senha do usuário para acessar o sistema;
i) RF09: O sistema deve exigir o *login* e senha do usuário administrador para acesso aos privilégios restritos do sistema;
j) RF10: O sistema deve permitir que o administrador gerencie todas as entidades;
k) RF11: O sistema deve permitir que o usuário possa alterar sua senha no caso de esquecimento. Nesse caso o sistema enviará um e-mail com um *link* de recuperação de conta;
l) RF12: O sistema deve permitir que o usuário possa gerenciar seus dados e seus anúncios em uma página intitulada “Minha Conta”;
m) RF13: O sistema deve permitir o administrador banir um usuário;
n) RF14: O sistema deve permitir o administrador desativar um anúncio;
o) RF15: O sistema deve permitir o administrador gerenciar subcategorias;
p) RF16: O sistema deve permitir usuários (autenticados e não autenticados) se cadastre na *newsletter*;
q) RF17: O sistema deve permitir usuários (autenticados e não autenticados) remova seu cadastro da *newsletter*;
r) RF18: O sistema deve permitir o administrador enviar e-mail para usuários cadastrados na *newsletter*;
s) RF19: Os e-mails de newsletter devem conter no final um link possibilitando a remoção do cadastro;
t) RF20: O sistema deve possuir um formulário de contato;
u) RF21: Os usuários cadastrados deverão possuir uma página com a lista de anúncios criados por ele;
v) RF22: O sistema deve permitir enviar mensagens para o criador do anúncio, diretamente na página de detalhes dos anúncios.

4.2.2 Requisitos Não Funcionais

Os requisitos não funcionais destacam aspectos externos as funcionalidades. Como tecnologias que deverão ser utilizadas. A seguir estão enumerados os requisitos não funcionais:

a) RNF01: O sistema deverá ser Web, utilizando HTML 5 e CSS 3;
b) RNF02: O sistema deverá ser desenvolvido na linguagem de programação PHP;
c) RNF03: O sistema deverá ser desenvolvido utilizando o framework Laravel;

d) RNF04: O banco de dados que será utilizado é o MySQL;
e) RNF05: Deve ser utilizado o módulo Apache como servidor.

4.2.3 Regras de Negócio

As regras de negócio definem restrições lógicas do sistema. A seguir estão enumeradas as regras de negócio:

a) RN01: O usuário só poderá acessar a página de manutenção dos dados após efetuar *login*;
b) RN02: O usuário só poderá acessar a página de manutenção de anúncios após efetuar *login*;
c) RN03: O usuário só poderá avaliar um anúncio da categoria "Serviços" após efetuar *login*;
d) RN03: O usuário só poderá alterar ou excluir anúncios que foram criados por ele;
e) RN04: O usuário poderá buscar anúncios no sistema, mesmo não tendo efetuado *login*;
f) RN05: O usuário poderá visualizar detalhes dos anúncios, mesmo não tendo efetuado *login*;
g) RN06: O usuário poderá enviar uma mensagem para o criador do anúncio, mesmo não tendo efetuado *login*;
h) RN07: O Administrador de sistema só poderá acessar a página de gerenciamento após efetuar *login* no sistema;
i) RN08: Na página inicial do sistema deverão ser mostrados os últimos anúncios publicados;
j) RN09: Na página inicial do sistema deverão ser mostrados os anúncios mais populares;
k) RN09: Na página inicial do sistema deverá possuir um *link* direto para cada uma das categorias ("Compra e Venda", "Imóveis", "Veículos", "Empregos" e "Serviços");
l) RN10: Na página inicial do sistema deverá possuir um *link* direto para as principais subcategorias (com mais anúncios cadastrados);
m) RN11: Na página inicial específica de cada categoria, deverão ser mostrados os últimos anúncios da mesma e uma opção para realizar buscas;

n) RN12: Nos anúncios da categoria "Serviços" os usuários poderão realizar uma avalição informando uma nota de 1 a 5, e opcionalmente realizar um comentário juntamente;
o) RN13: A página de cadastro de anúncios será dinâmica. De acordo com a categoria, serão carregadas as subcategorias e os campos específicos disponíveis para o preenchimento.

4.3 Diagrama de Entidade Relacionamento

Consiste num conjunto de objetos representativos de um ambiente, chamados entidades e nos relacionamentos que as interligam. Uma entidade se distingue de outras entidades pelos atributos que ela contém.

As entidades podem ser qualquer coisa real ou abstrata, que se faz necessário guardar informações.

A seguir está o diagrama de entidade relacionamento do sistema desenvolvido. O mesmo foi criado utilizando a ferramenta MySQL Workbench.

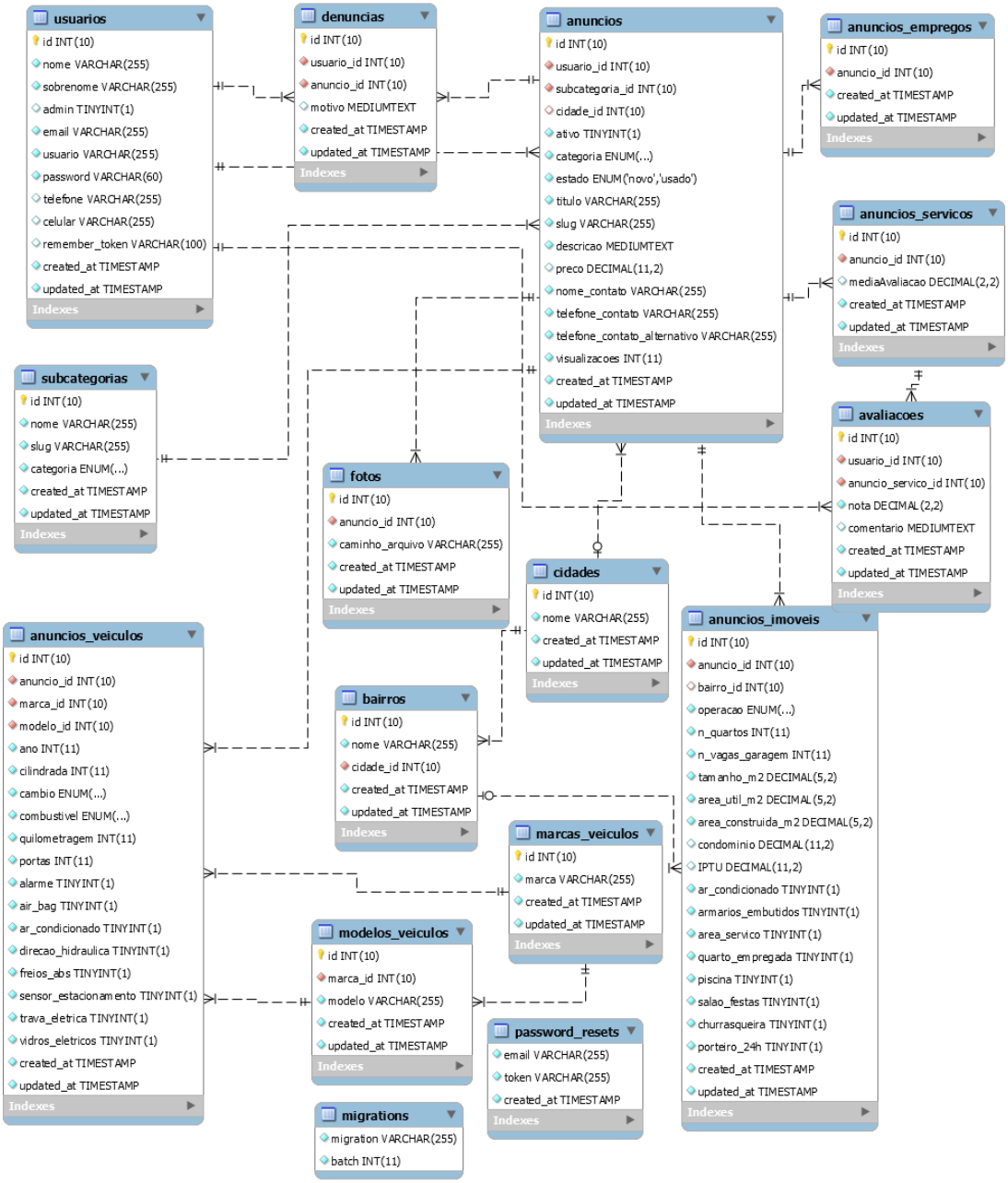

Figura 13 - Diagrama de Entidade Relacionamento
Fonte: Próprio autor.

4.4 Diagramas UML

Fowler (2004, p. 25) descreve UML como sendo "[...] uma família de notações gráficas, apoiada por um metamodelo único, que ajuda na descrição e no projeto de sistemas de software, particularmente daqueles construídos utilizando o estilo orientado a objetos [...]".

A Linguagem de Modelagem Unificada é uma linguagem gráfica (ou visual) para especificar, construir e documentar os artefatos dos sistemas de informação. Ao utilizá-la de forma adequada pode-se eliminar grande parte dos problemas encontrados nas etapas de

desenvolvimento. É primordial o planejamento antes do desenvolvimento.

Nesse trabalho de conclusão de curso foi utilizado UML para a criação dos diagramas de Implantação, Casos de Uso, Atividade e Sequência.

Os diagramas contidos nesse trabalho foram criados utilizando a ferramenta Enterprise Architect.

4.4.1 Diagramas de Casos de Uso

Os diagramas de casos de uso descrevem a relação entre atores e casos de uso de um sistema. Este diagrama permite dar uma visão ampliada e de alto nível, sendo fundamental a definição correta da sua limitação.

O sistema possui dois atores: “Usuário” e “Administrador”.

A figura a seguir mostra os casos de uso do ator “Usuário”.

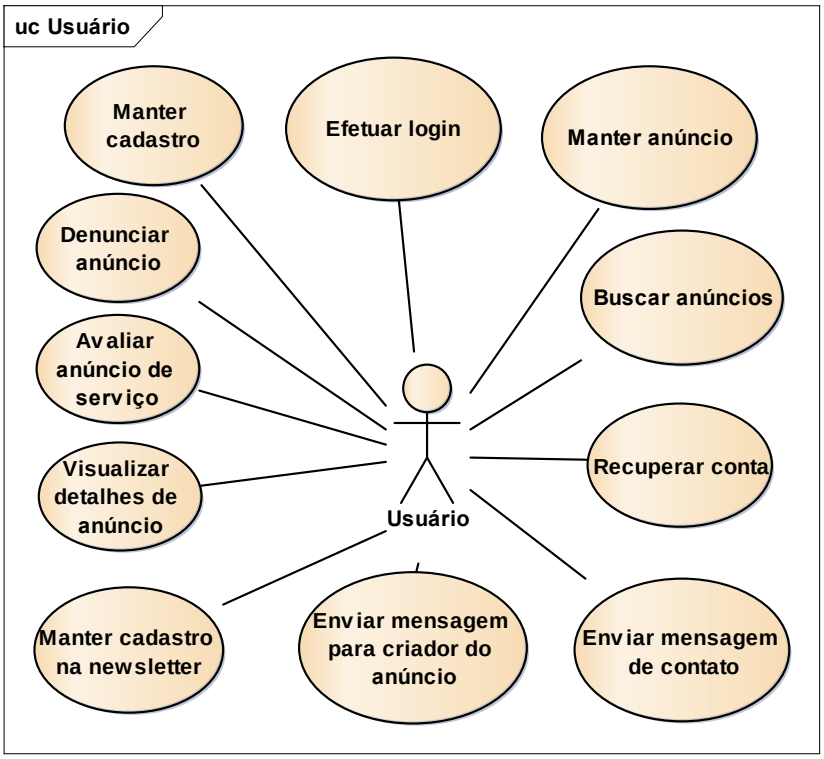

Figura 14 - Casos de Uso - Usuário
Fonte: Próprio autor.

Essa próxima figura mostra os casos de uso do ator “Administrador”.

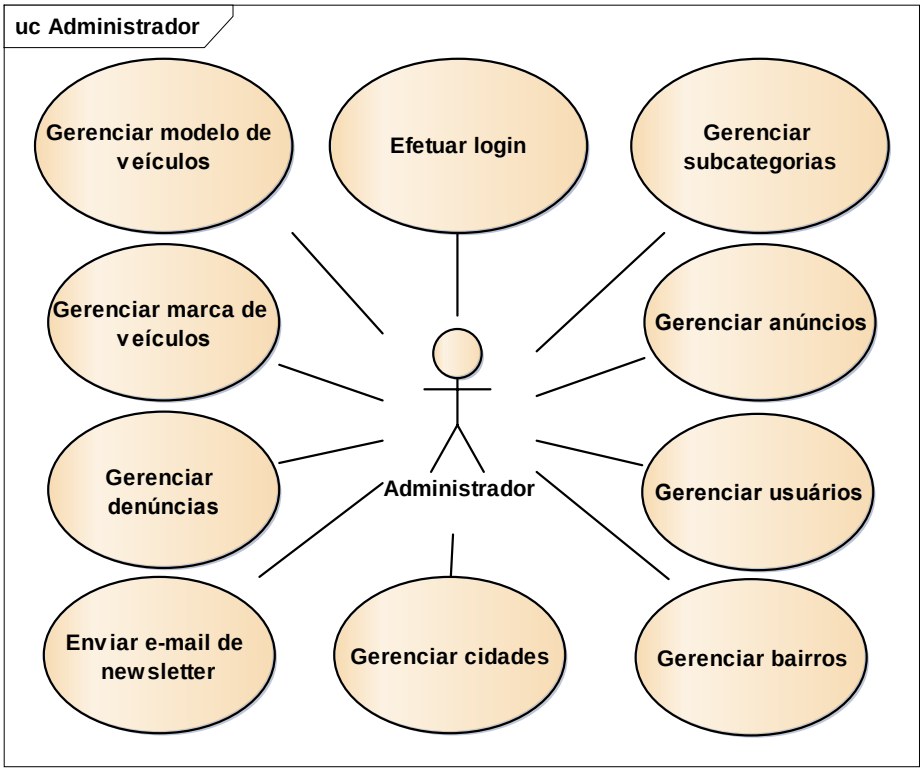

Figura 15 - Casos de Uso - Administrador
Fonte: Próprio autor.

4.4.2 Diagrama de Classes

O diagrama de classes ilustra as classes, objetos, estruturas e relacionamentos. Através dele é possível visualizar as implementações que deverão ser realizadas no projeto proposto.

A seguir está descrito o diagrama de classes. Por se tratar de um sistema que utiliza o padrão MVC, nele estão separadas as três camadas, respeitando o objetivo do *controller* como sendo o responsável pela lógica do sistema.

A figura 16 contém o diagrama de classes dos objetos e relações da interface de administração. Para os usuários comuns está explicitado na figura 17. Embora utilizem objetos e classes comuns entre as duas interfaces, existe uma separação lógica para que seja possível a existência de regras de permissão de acesso.

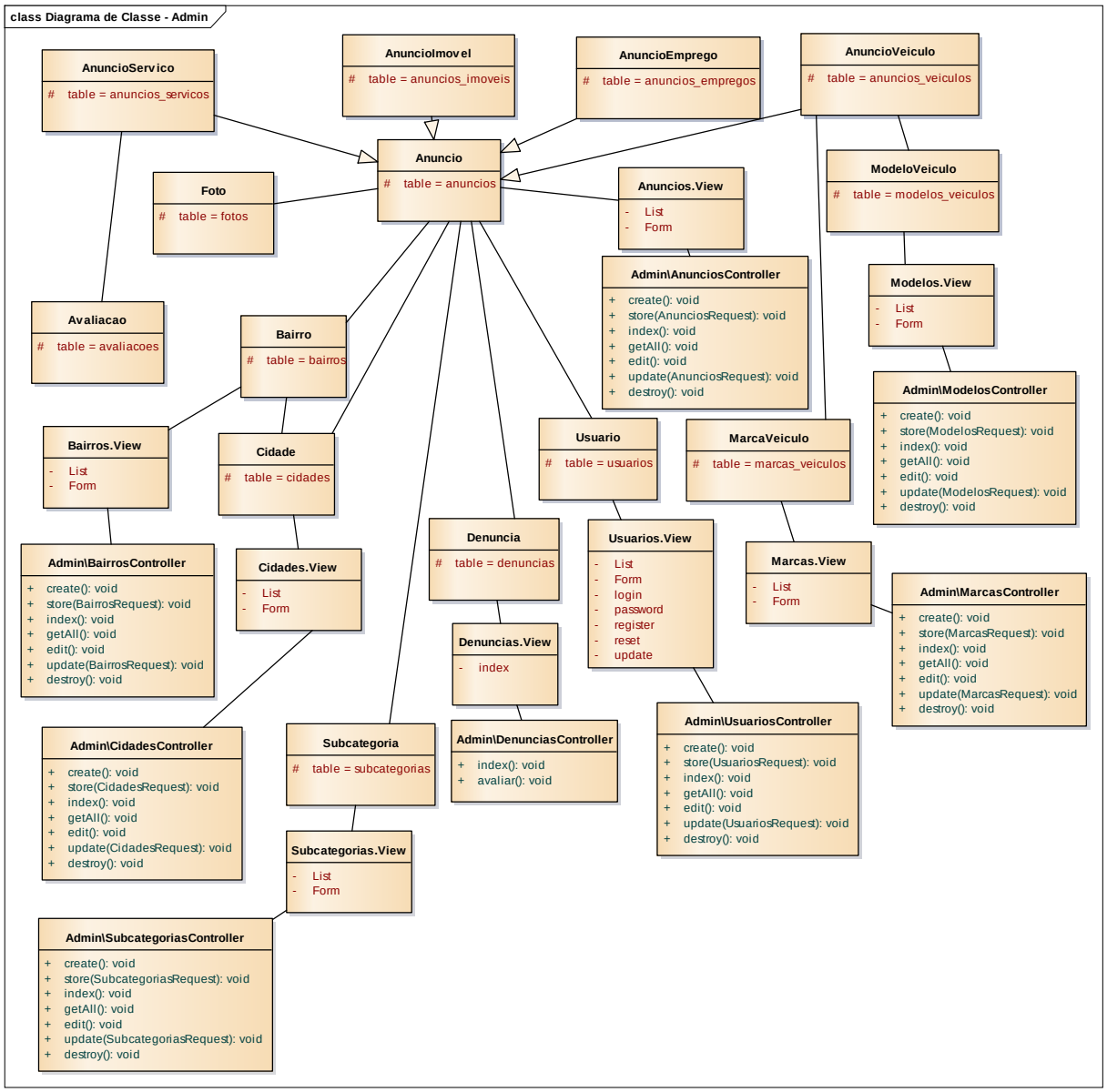

Figura 16 - Diagrama de Classes - Administrador
Fonte: Próprio autor.

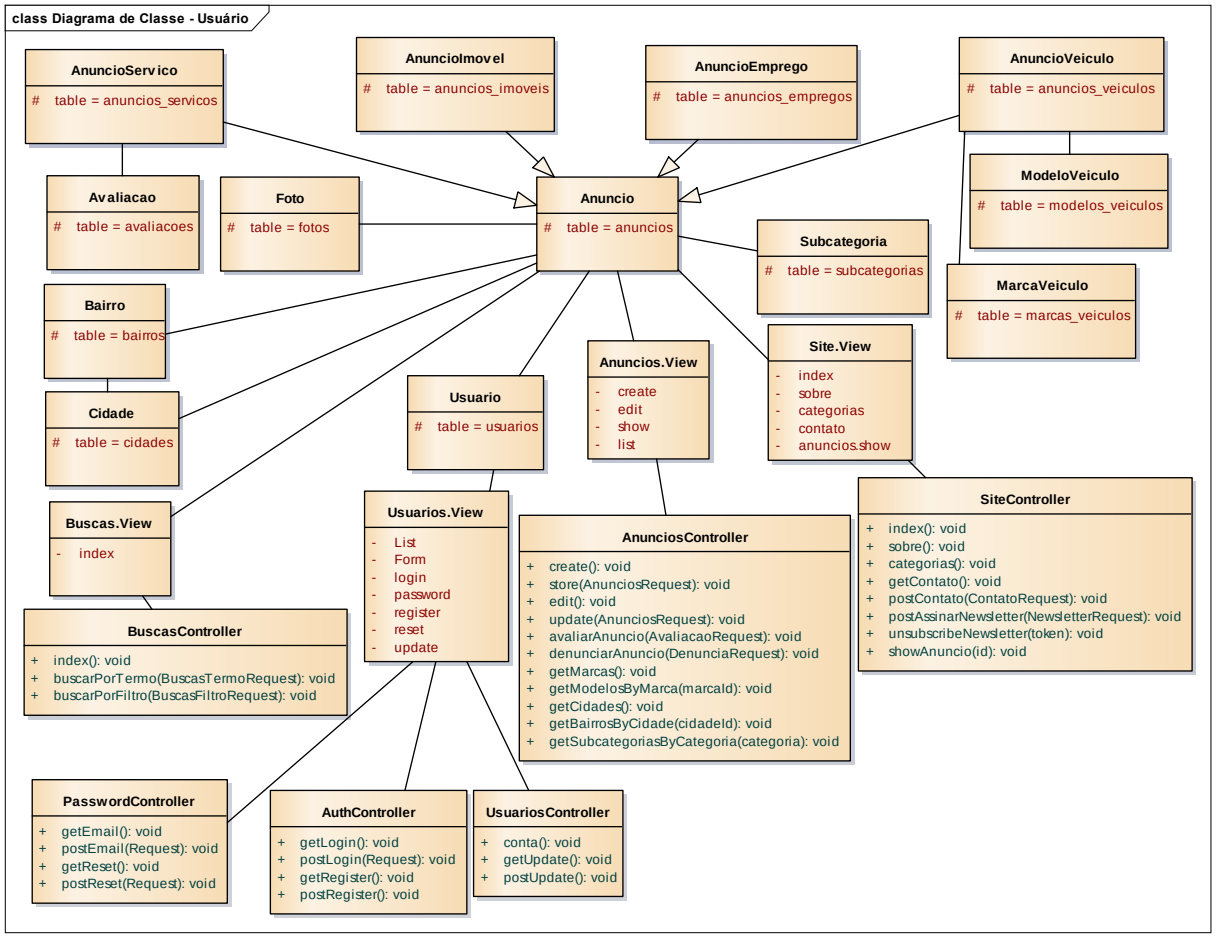

Figura 17 - Diagrama de Classes - Usuário
Fonte: Próprio autor.

4.4.3 Diagramas de Atividade

Fowler (2004, p. 118) define o diagrama de atividade como “[...] uma técnica para descrever lógica de procedimento, processo de negócio e fluxo de trabalho [...]”. Fowler (2004), complementa que os diagramas se assemelham a fluxogramas, com uma única diferença importante que é a possibilidade de comportamentos paralelos.

Os diagramas de atividade facilitam o entendimento sistemático, pois permite “quebrar” os casos de uso em partes menores e específicas do sistema, através disso a implementação pôde ser realizada tendo em vista as etapas necessárias para se atingir cada funcionalidade, tendo em vista as regras de negócios e requisitos já definidos.

A seguir estão descritos os diagramas de atividade do sistema.

4.4.3.1 Usuário - AD1.01 - Manter cadastro

O diagrama de atividade a seguir do ator “usuário” especifica como é o fluxo de manter o cadastro do usuário, que tem seu início na página de cadastro de usuário.

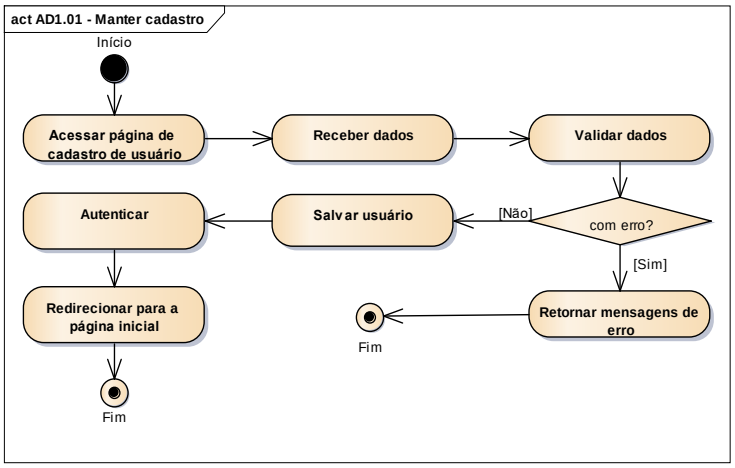

Figura 18 - AD1.01 - Manter cadastro
Fonte: Próprio autor.

4.4.3.2 Usuário - AD1.02 – Login

O diagrama de atividade a seguir do ator “usuário” especifica como é o fluxo de login, que tem seu início na página de login.

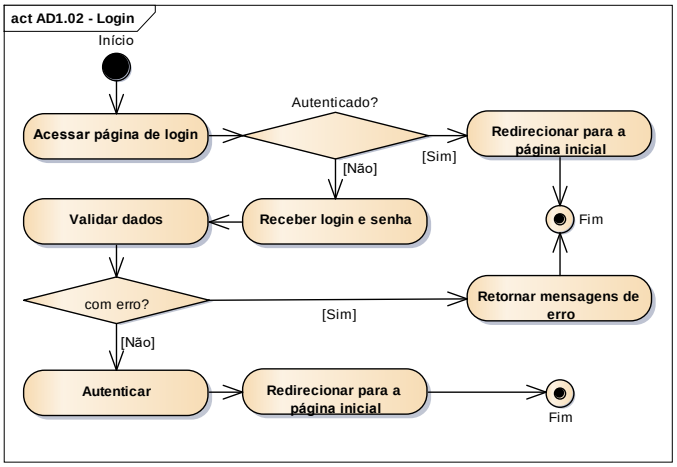

Figura 19 - AD1.02 - Login
Fonte: Próprio autor.

4.4.3.3 Usuário - AD1.03 - Criar anúncio

O diagrama de atividade a seguir do ator “usuário” especifica como é o fluxo de criar

anúncio, que tem seu início na página de criação de anúncio.

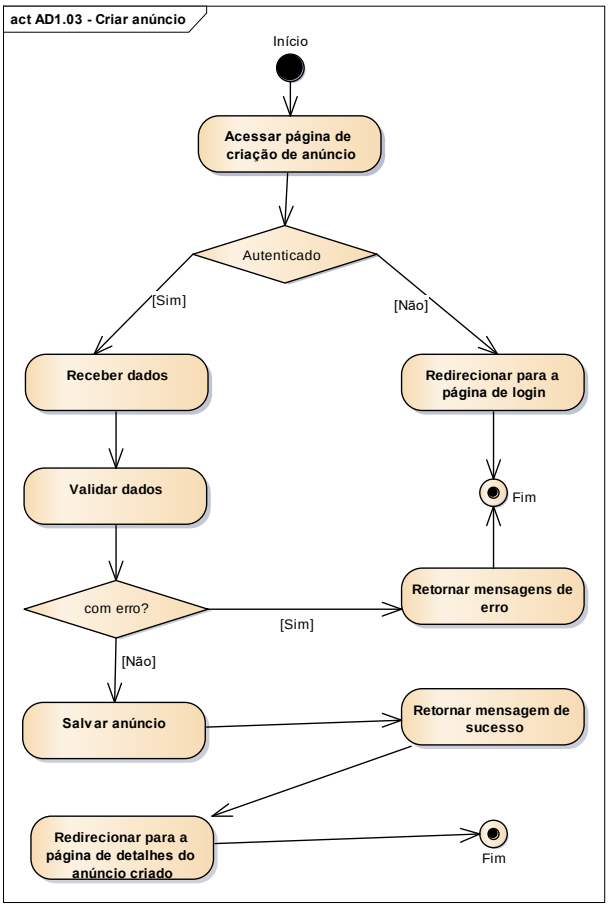

Figura 20 - AD1.03 - Criar anúncio
Fonte: Próprio autor.

4.4.3.4 Usuário - AD1.04 - Visualizar anúncios

O diagrama de atividade a seguir do ator “usuário” especifica como é o fluxo de visualizar anúncios, que tem seu início na página de visualização detalhada do anúncio.

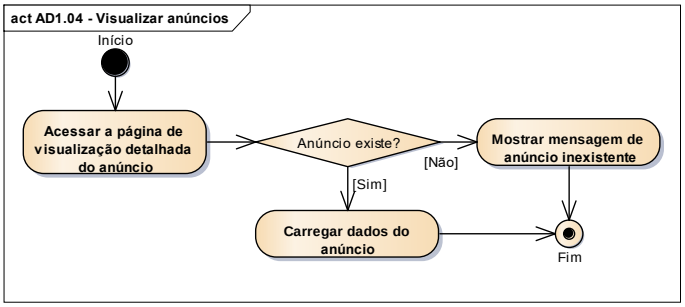

Figura 21 - AD1.04 - Visualizar anúncios
Fonte: Próprio autor.

4.4.3.5 Usuário - AD1.05 - Alterar anúncio

O diagrama de atividade a seguir do ator “usuário” especifica como é o fluxo de alterar anúncio, que tem seu início na página de anúncios do usuário.

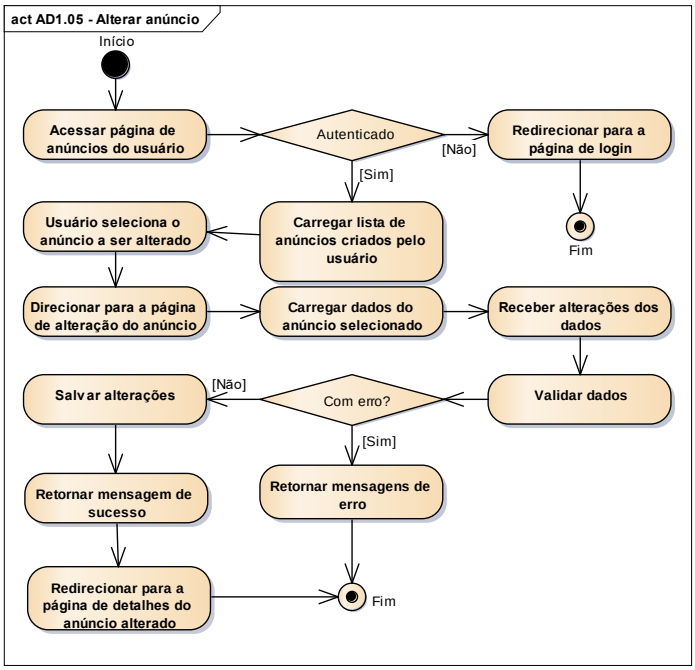

Figura 22 - AD1.05 - Alterar anúncio
Fonte: Próprio autor.

4.4.3.6 Usuário - AD1.06 - Excluir anúncio

O diagrama de atividade a seguir do ator “usuário” especifica como é o fluxo de excluir anúncio, que tem seu início na página de anúncios do usuário.

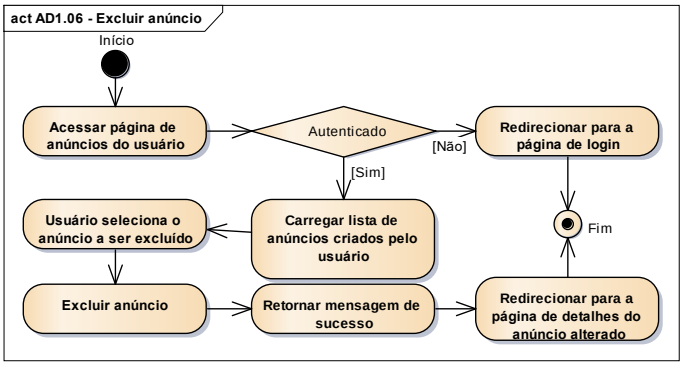

Figura 23 - AD1.06 - Excluir anúncio
Fonte: Próprio autor.

4.4.3.7 Usuário - AD1.07 - Buscar anúncios

O diagrama de atividade a seguir do ator “usuário” especifica como é o fluxo de buscar anúncio, que tem seu início na opção de busca.

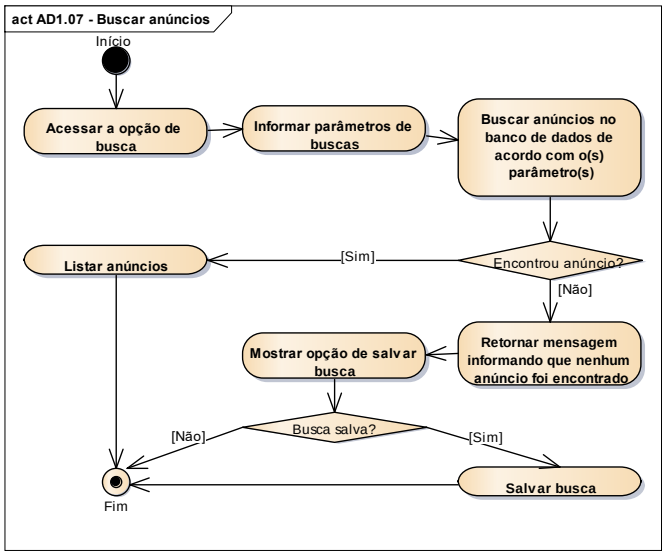

Figura 24 - AD1.07 - Buscar anúncios
Fonte: Próprio autor.

4.4.3.8 Usuário - AD1.08 - Avaliar anúncio de serviço

O diagrama de atividade a seguir do ator “usuário” especifica como é o fluxo de avaliar anúncio de serviço, que tem seu início na página de visualização detalhada do anúncio.

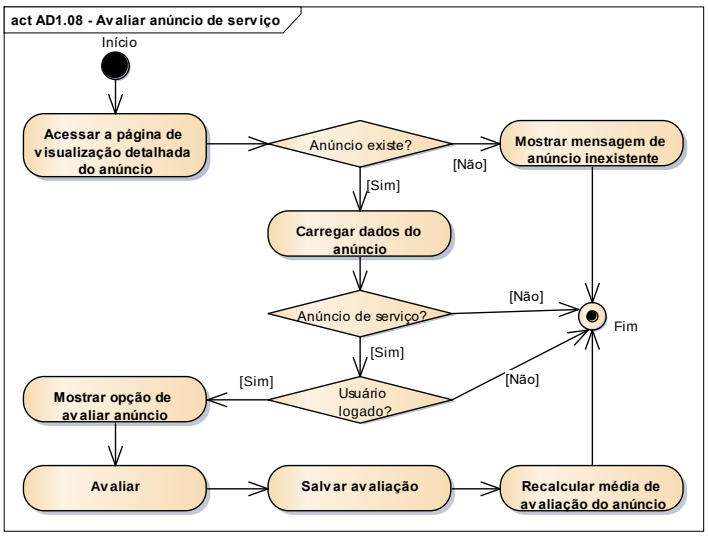

Figura 25 - AD1.08 - Avaliar anúncio de serviço
Fonte: Próprio autor.

4.4.3.9 Usuário - AD1.09 – Denunciar anúncio

O diagrama de atividade a seguir do ator “usuário” especifica como é o fluxo de denúncia de anúncios que tem seu início na página de visualização detalhada do anúncio.

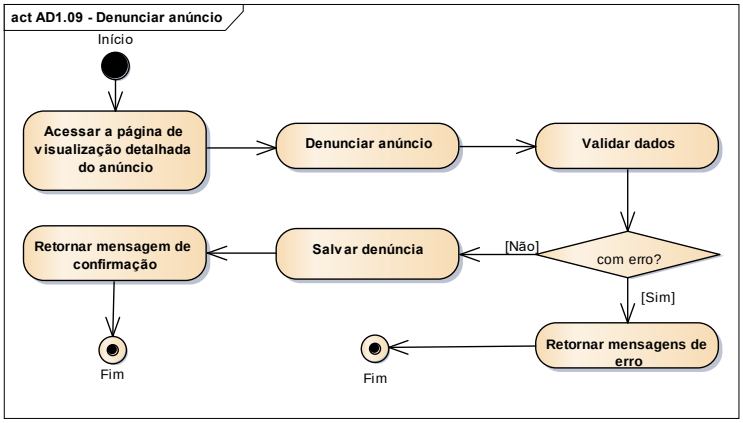

Figura 26 - AD1.09 – Denunciar anúncio
Fonte: Próprio autor.

4.4.3.10 Usuário - AD1.10 - Recuperar conta

O diagrama de atividade a seguir do ator “usuário” especifica como é o fluxo de recuperação de conta que tem seu início na página de recuperação de conta.

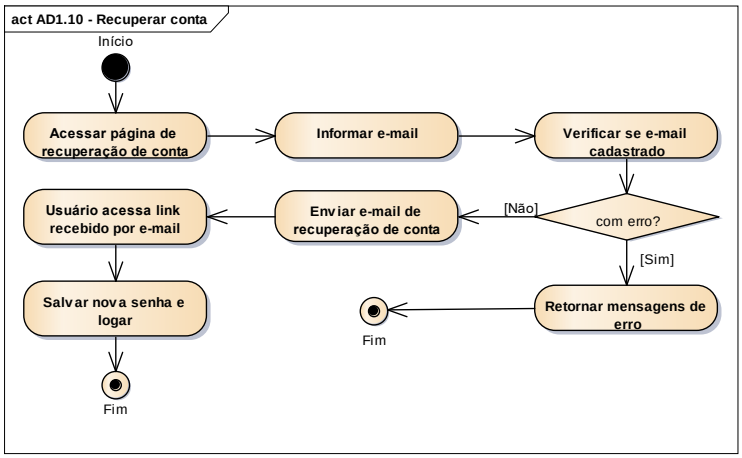

Figura 27 - AD1.10 - Recuperar conta
Fonte: Próprio autor.

4.4.3.11 Usuário - AD1.11 – Cadastramento na *newsletter*

O diagrama de atividade a seguir do ator “usuário” especifica como é o fluxo de cadastramento na *newsletter* que tem seu início na página inicial do sistema.

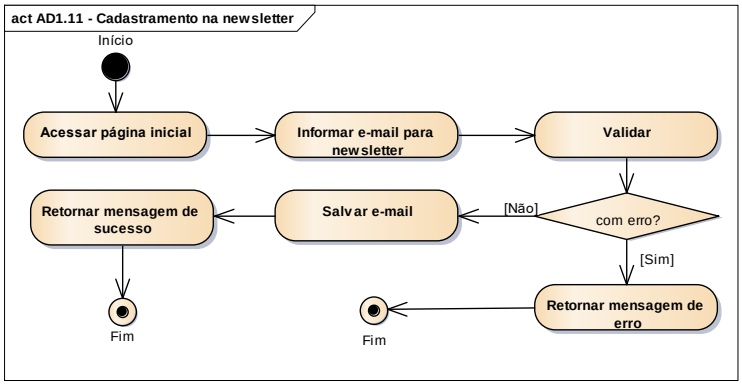

Figura 28 - AD1.11 - Cadastramento na newsletter
Fonte: Próprio autor.

4.4.3.11 Usuário - AD1.12 - Remover cadastro da newsletter

O diagrama de atividade a seguir do ator "usuário" especifica como é o fluxo de remoção da cadastro da newsletter que tem seu início a partir do acesso ao link de remoção de cadastro.

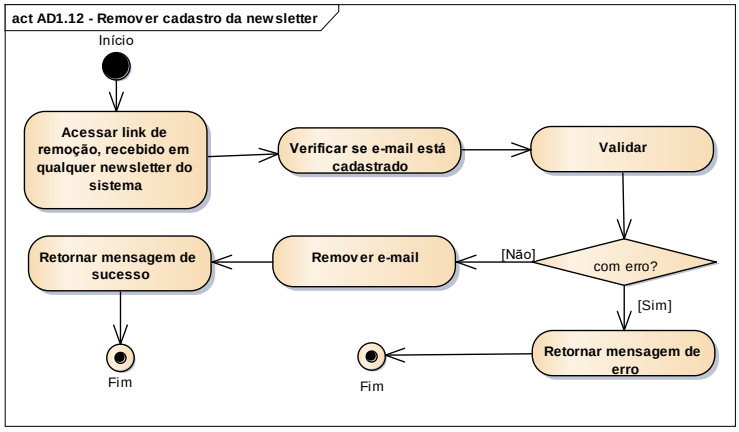

Figura 29 - AD1.12 - Remover cadastro da newsletter
Fonte: Próprio autor.

4.4.3.12 Administrador - AD2.01 - Cadastrar entidade

O diagrama de atividade a seguir do ator "Administrador" especifica como é o fluxo de cadastrar entidade, que tem seu início ao acessar a página de administração.

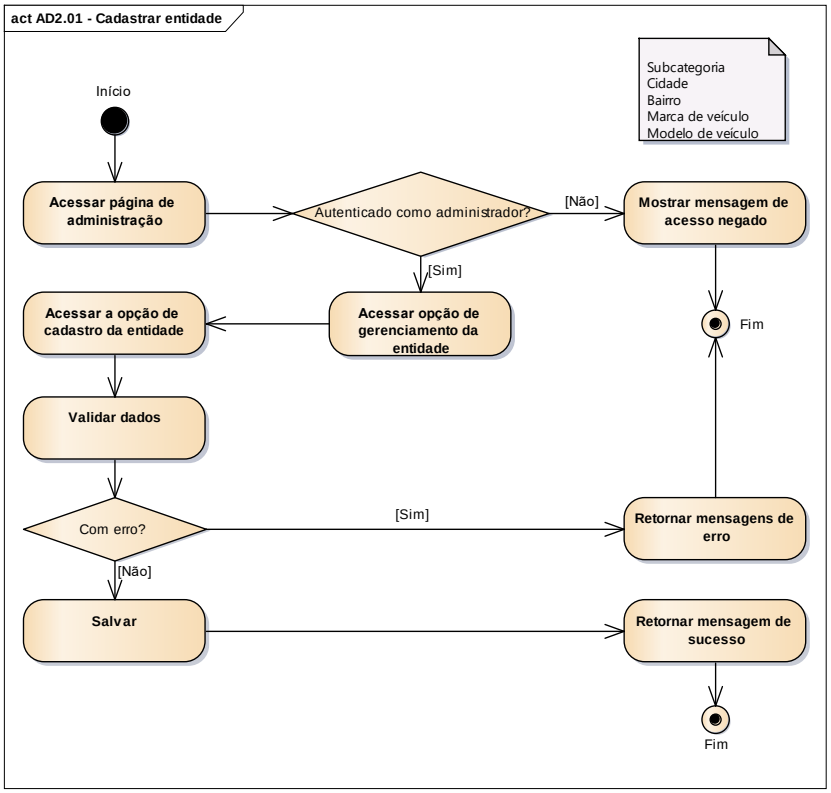

Figura 30 – AD2.01 - Cadastrar entidade
Fonte: Próprio autor.

4.4.3.14 Administrador - AD2.02 - Alterar entidade

O diagrama de atividade a seguir do ator “Administrador” especifica como é o fluxo de alterar entidade, que tem seu início ao acessar a página de administração.

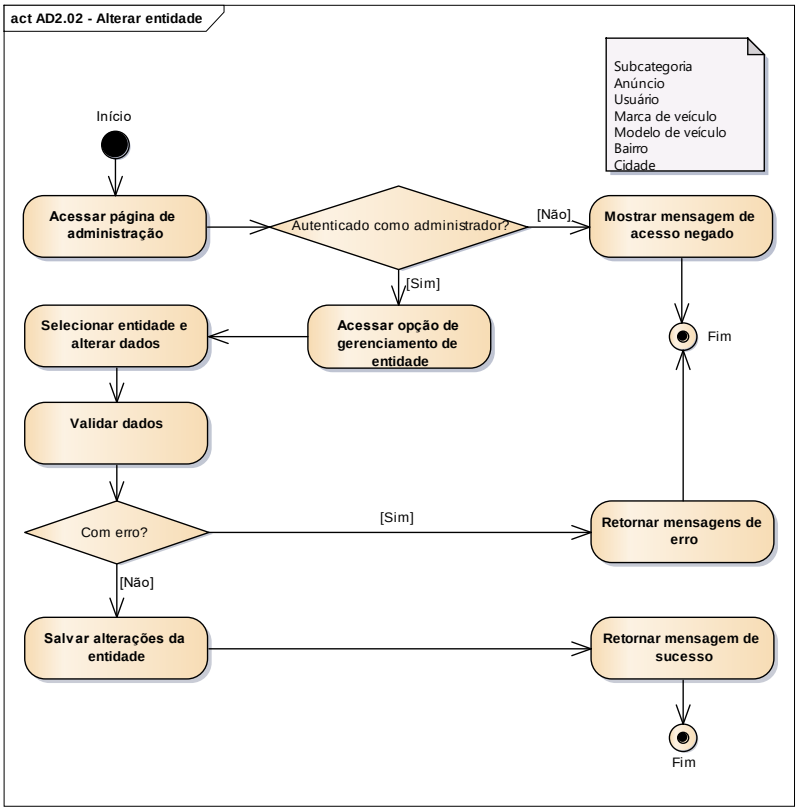

Figura 31 - AD2.02 - Alterar entidade
Fonte: Próprio autor.

4.4.3.15 Administrador - AD2.03 - Excluir entidade

O diagrama de atividade a seguir do ator “Administrador” especifica como é o fluxo de excluir entidade, que tem seu início ao acessar a página de administração.

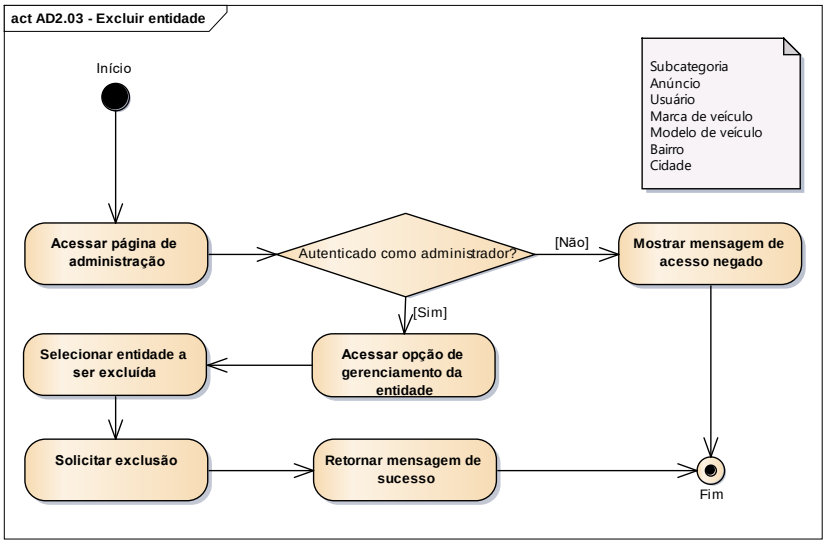

Figura 32 – AD2.03 - Excluir entidade
Fonte: Próprio autor.

4.4.3.16 Administrador - AD2.04 – Login

O diagrama de atividade a seguir do ator “Administrador” especifica como é o fluxo de login, que tem seu início ao acessar a página de login.

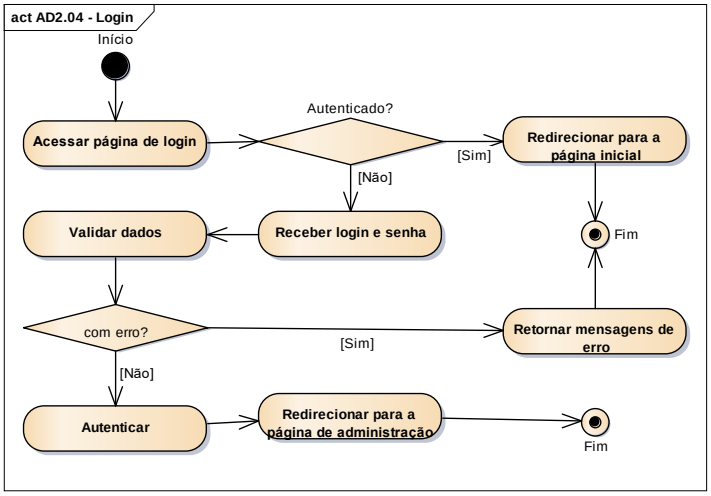

Figura 33 - AD2.04 - Login
Fonte: Próprio autor.

4.4.4 Diagramas de Sequência

O diagrama de sequência ilustra o comportamento de um cenário. Ele mostra várias situações e mensagens que são passadas entre os objetos dentro de um caso de uso.

Os diagramas de sequência mostram a interação, exibindo cada parte com uma linha de vida, que corre verticalmente na página. A ordem das mensagens deve ser lida de cima para baixo.

A seguir estão descritos os diagramas de sequência do projeto proposto.

4.4.4.1 Usuário - SEQ1.01 - Manter cadastro

O diagrama de sequência a seguir do ator “usuário” especifica a sequência de manter cadastro, que tem seu início ao acessar a página de cadastro de usuário.

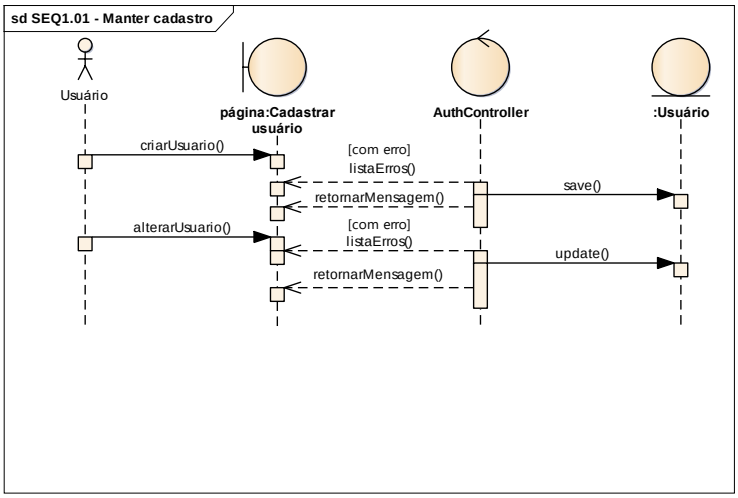

Figura 34 - SEQ1.01 - Manter cadastro
Fonte: Próprio autor.

4.4.4.2 Usuário - SEQ1.02 - Efetuar Login

O diagrama de sequência a seguir do ator “usuário” especifica a sequência de efetuar login, que tem seu início ao acessar a página de login.

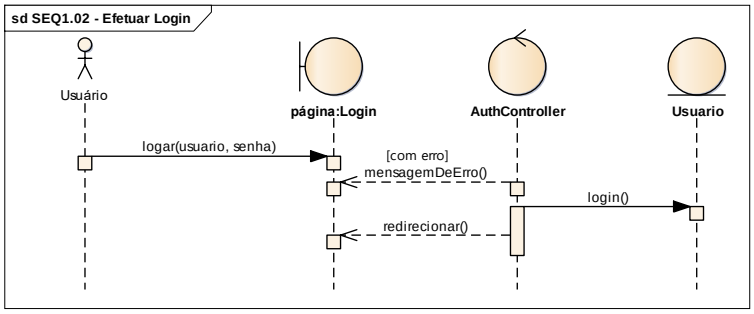

Figura 35 - SEQ1.02 - Efetuar Login
Fonte: Próprio autor.

4.4.4.3 Usuário - SEQ1.03 - Manter anúncio

O diagrama de sequência a seguir do ator “usuário” especifica a sequência de manter cadastro, que tem seu início ao acessar a página de cadastro de usuário.

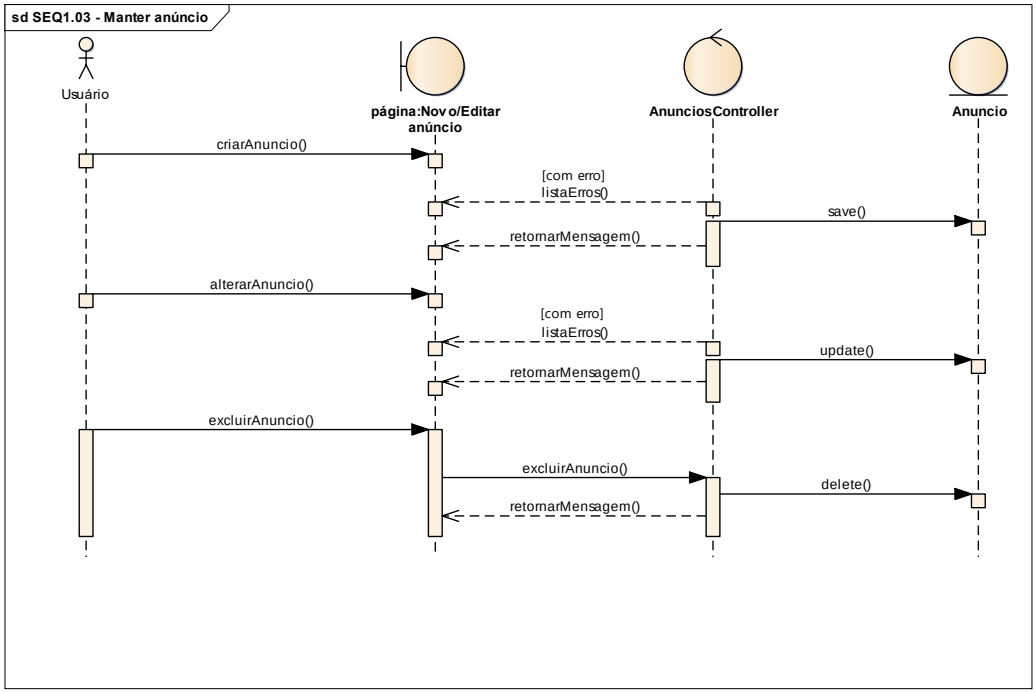

Figura 36 - SEQ1.03 - Manter anúncio
Fonte: Próprio autor.

4.4.4.4 Usuário - SEQ1.04 - Buscar anúncios

O diagrama de sequência a seguir do ator “usuário” especifica a sequência de buscar anúncios, que tem seu início ao acessar a opção buscar na barra de navegação.

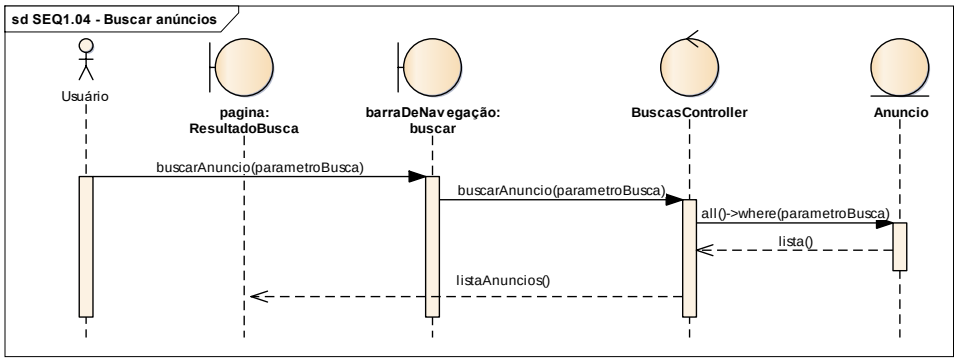

Figura 37 - SEQ1.04 - Buscar anúncios
Fonte: Próprio autor.

4.4.4.5 Usuário - SEQ1.05 - Recuperar conta

O diagrama de sequência a seguir do ator “usuário” especifica a sequência de recuperar conta, que tem seu início na página de solicitação de *link* de recuperação.

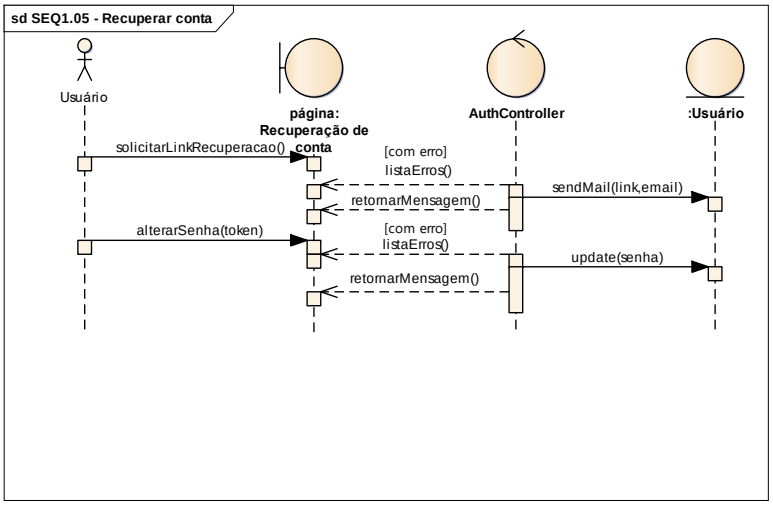

Figura 38 - SEQ1.05 - Recuperar conta
Fonte: Próprio autor.

4.4.4.6 Usuário - SEQ1.06 - Mensagem de contato

O diagrama de sequência a seguir do ator “usuário” especifica a sequência de recuperar conta, que tem seu início na página de visualização detalhada do anúncio.

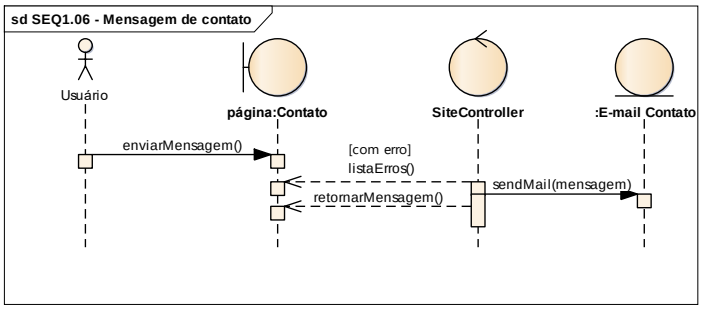

Figura 39 - SEQ1.06 - Mensagem de contato
Fonte: Próprio autor.

4.4.4.7 Usuário - SEQ1.07 - Detalhes anúncio

O diagrama de sequência a seguir do ator “usuário” especifica a sequência de visualizar detalhes de um anúncio específico, que tem seu início na página de visualização detalhada do anúncio.

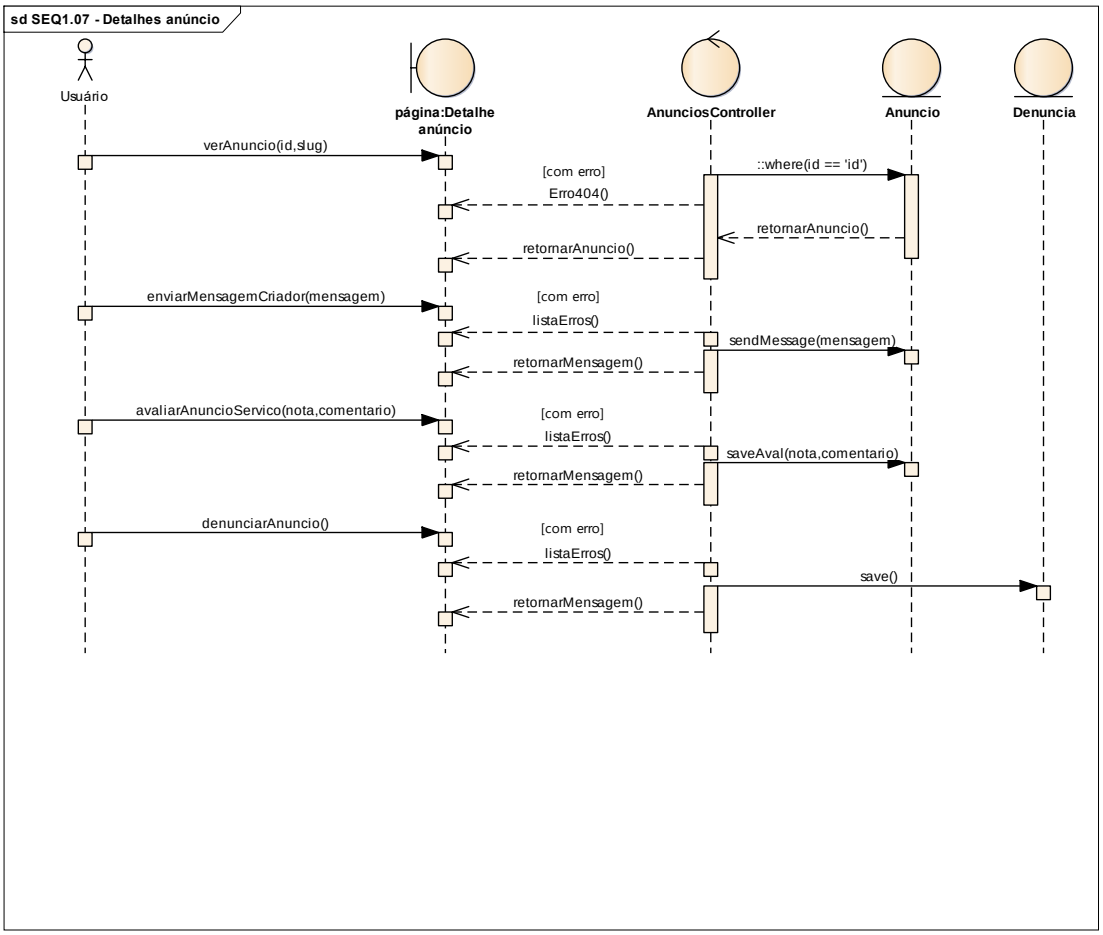

Figura 40 - SEQ1.07 - Detalhes anúncio
Fonte: Próprio autor.

4.4.4.8 Usuário - SEQ1.08 - Manter cadastro na *newsletter*

O diagrama de sequência a seguir do ator “usuário” especifica a sequência de manter-se cadastrado na *newsletter*.

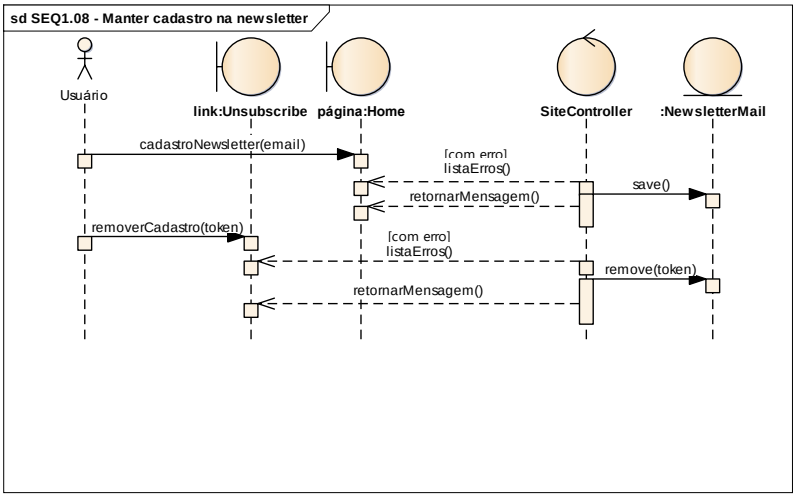

Figura 41 - SEQ1.08 - Manter cadastro na newsletter
Fonte: Próprio autor.

4.4.4.9 Administrador - SEQ2.01 - Efetuar Login

O diagrama de sequência a seguir do ator “administrador” especifica a sequência de efetuar login, que tem seu início ao acessar a página de login.

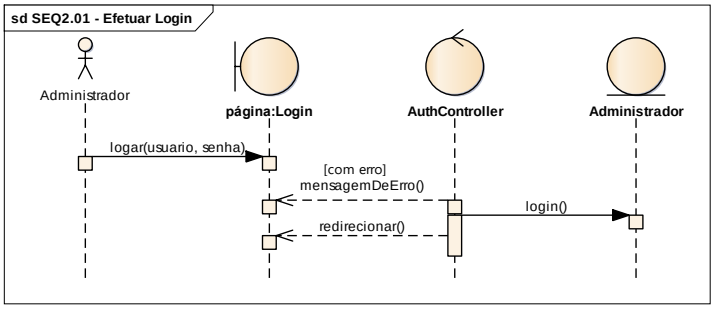

Figura 42 - SEQ2.01 - Efetuar Login
Fonte: Próprio autor.

4.4.4.10 Administrador - SEQ2.02 - Gerenciar subcategorias

O diagrama de sequência a seguir do ator “administrador” especifica a sequência de gerenciar subcategorias, que tem seu início ao acessar a página de gerenciamento de categorias. Nela ele pode criar, alterar e excluir subcategorias.

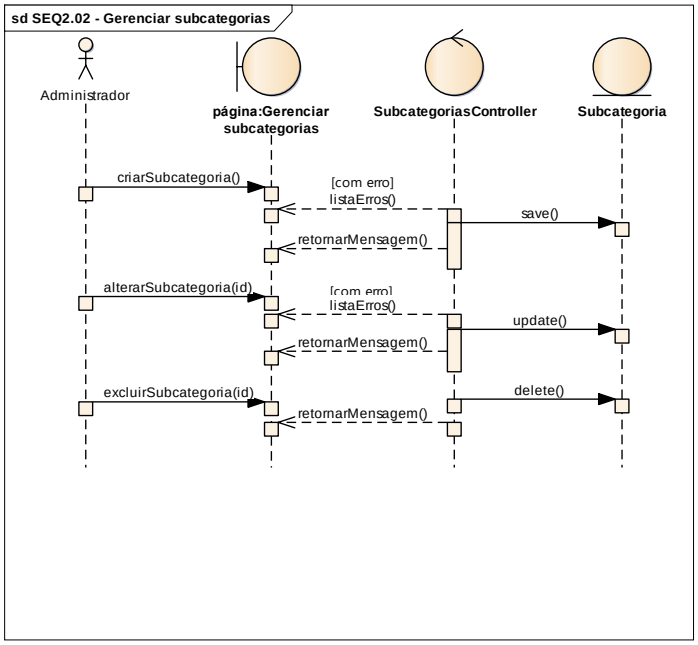

Figura 43 - SEQ2.02 - Gerenciar subcategorias
Fonte: Próprio autor.

4.4.4.11 Administrador - SEQ2.03 - Gerenciar anúncios

O diagrama de sequência a seguir do ator “administrador” especifica a sequência de gerenciar anúncios, que tem seu início ao acessar a página de gerenciar anúncios.

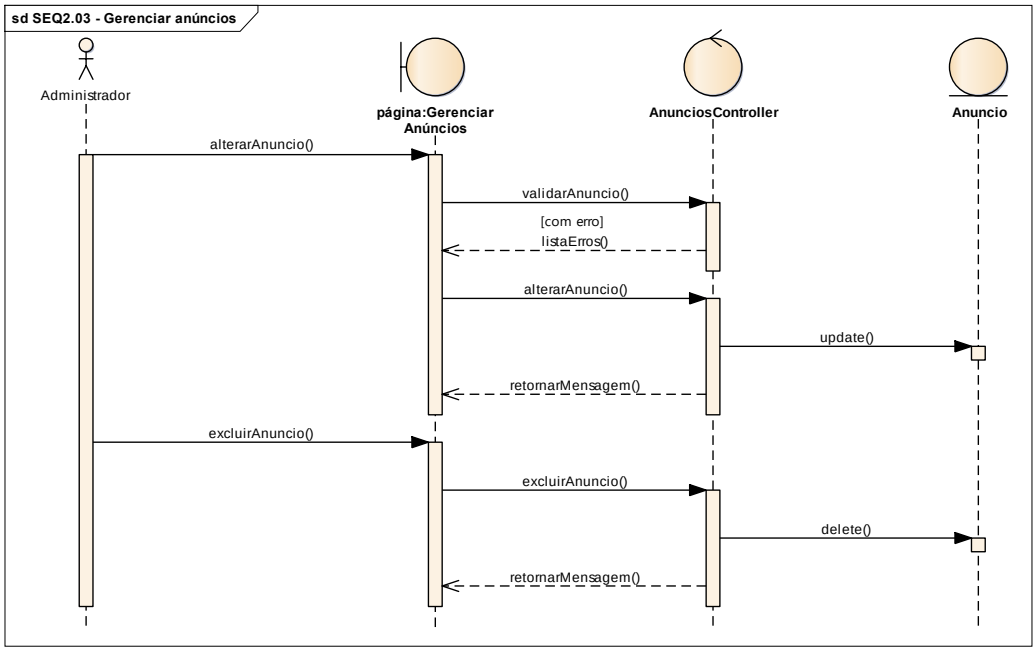

Figura 44 - SEQ2.03 - Gerenciar anúncios
Fonte: Próprio autor.

4.4.4.12 Administrador - SEQ2.04 - Gerenciar usuários

O diagrama de sequência a seguir do ator “administrador” especifica a sequência de gerenciar usuários, que tem seu início ao acessar a página de gerenciar usuários.

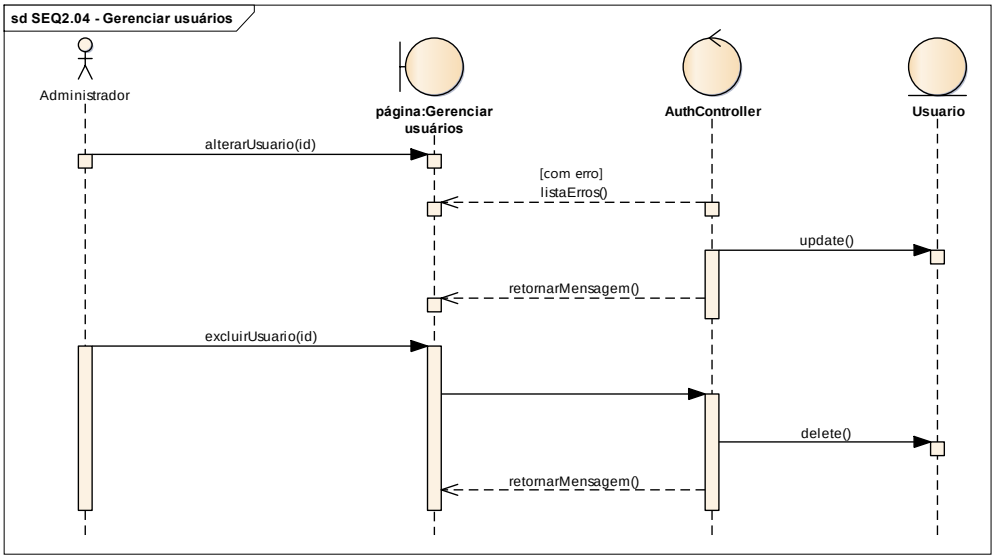

Figura 45 - SEQ2.04 - Gerenciar usuários
Fonte: Próprio autor.

4.4.4.13 Administrador - SEQ2.05 - Gerenciar cidades

O diagrama de sequência a seguir do ator “administrador” especifica a sequência de gerenciar cidades, que tem seu início ao acessar a página de gerenciar cidades.

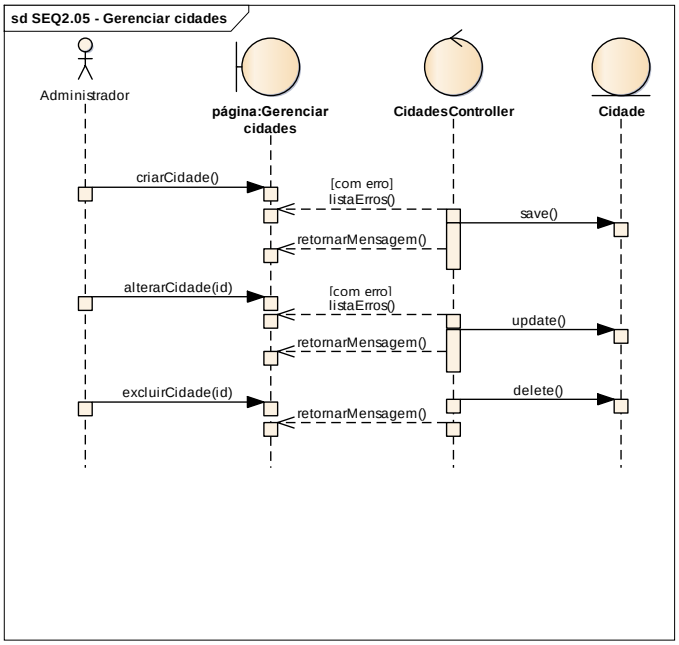

Figura 46 - SEQ2.05 - Gerenciar cidades
Fonte: Próprio autor.

4.4.4.14 Administrador - SEQ2.06 - Gerenciar bairros

O diagrama de sequência a seguir do ator “administrador” especifica a sequência de gerenciar bairros, que tem seu início ao acessar a página de gerenciar bairros.

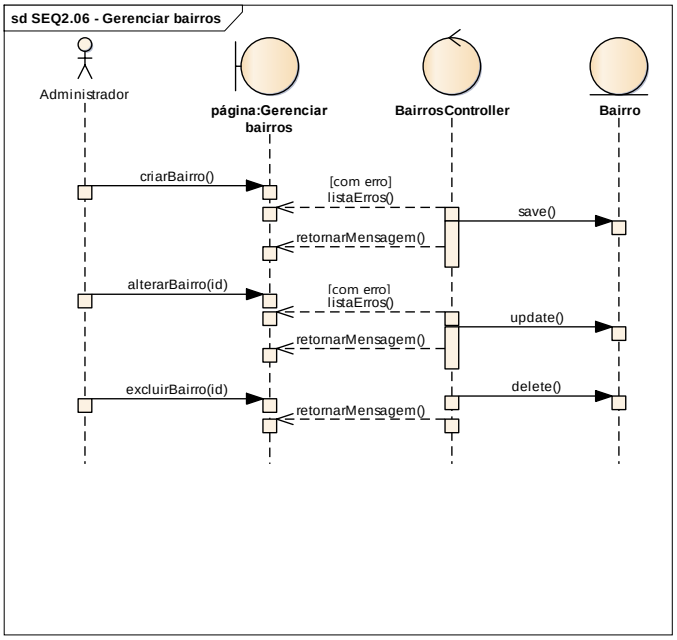

Figura 47 - SEQ2.06 - Gerenciar bairros
Fonte: Próprio autor.

4.4.4.15 Administrador - SEQ2.07 - Enviar *newsletter*

O diagrama de sequência a seguir do ator “administrador” especifica a sequência de envio de *newsletter*, que tem seu início ao acessar a página de envio de *newsletter*.

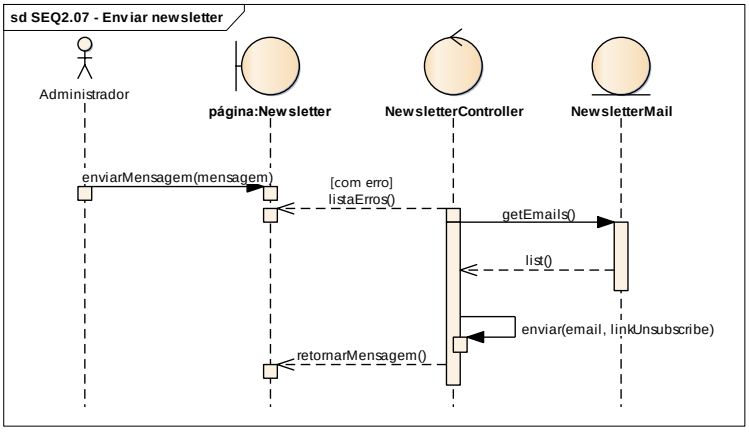

Figura 48 - SEQ2.07 - Enviar *newsletter*
Fonte: Próprio autor.

4.4.4.16 Administrador - SEQ2.08 - Gerenciar denúncias

O diagrama de sequência a seguir do ator “administrador” especifica a sequência de gerenciar denúncias, que tem seu início ao acessar a página de gerenciar denúncias.

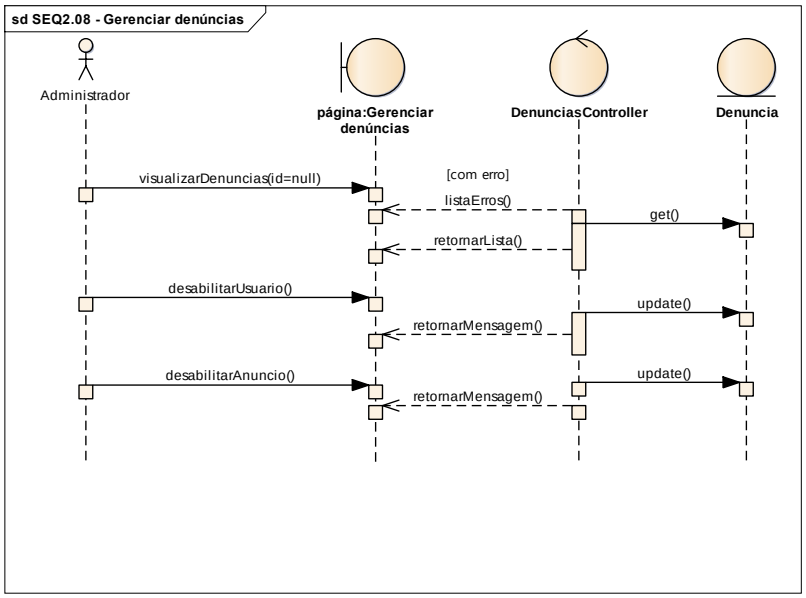

Figura 49 - SEQ2.08 - Gerenciar denúncias
Fonte: Próprio autor.

4.4.4.17 Administrador - SEQ2.09 - Gerenciar marcas de veículos

O diagrama de sequência a seguir do ator “administrador” especifica a sequência de gerenciar marcas de veículos, que tem seu início ao acessar a página de gerenciar marcas de veículos.

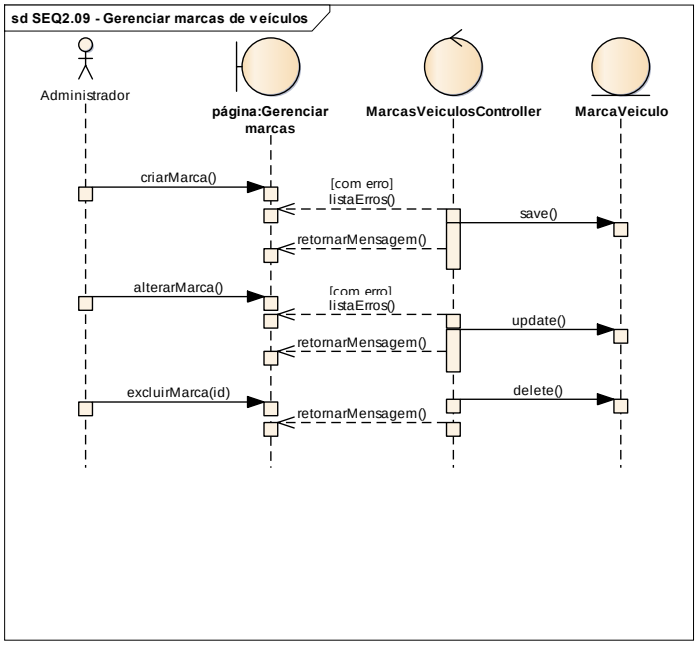

Figura 50 - SEQ2.09 - Gerenciar marcas de veículos
Fonte: Próprio autor.

4.4.4.17 Administrador - SEQ2.10 - Gerenciar modelos de veículos

O diagrama de sequência a seguir do ator “administrador” especifica a sequência de gerenciar modelos de veículos, que tem seu início ao acessar a página de gerenciar modelos de veículos.

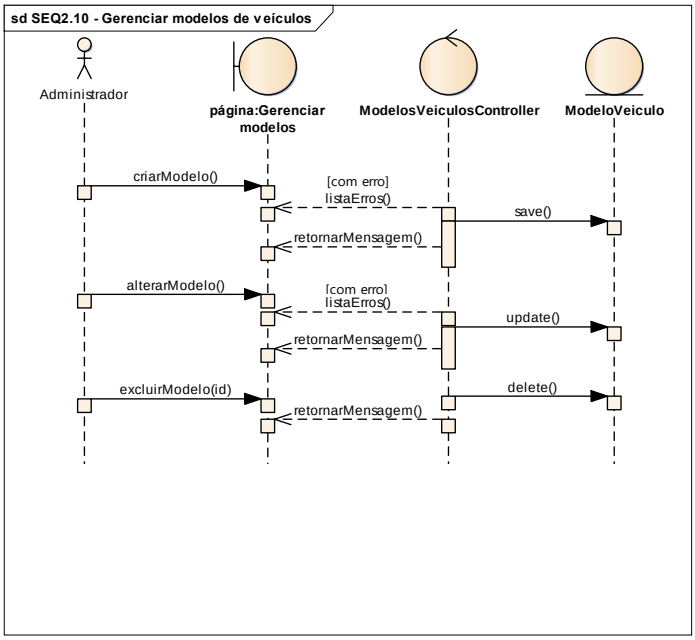

Figura 51 - SEQ2.10 - Gerenciar modelos de veículos
Fonte: Próprio autor.

4.4.5 Diagrama de Implantação

O diagrama de implantação especifica a configuração dos elementos de processamento e dos componentes do sistema.

Ele é formado por um conjunto de nós ligados por associações de comunicação. Os nós podem conter instâncias de componentes (de execução), o que significa que um componente é instalado e executado num nó.

O diagrama a seguir possui três nós. O primeiro diz respeito ao usuário que utilizará o sistema, nesse caso é o navegador web, que interpretará o HTML, o CSS e processará o Javascript.

O segundo nó será onde o sistema será hospedado, rodando em apache e interpretado pelo PHP.

O terceiro nó diz respeito ao banco de dados, nesse caso MySQL, que se comunicará de forma direta, apenas com o segundo nó.

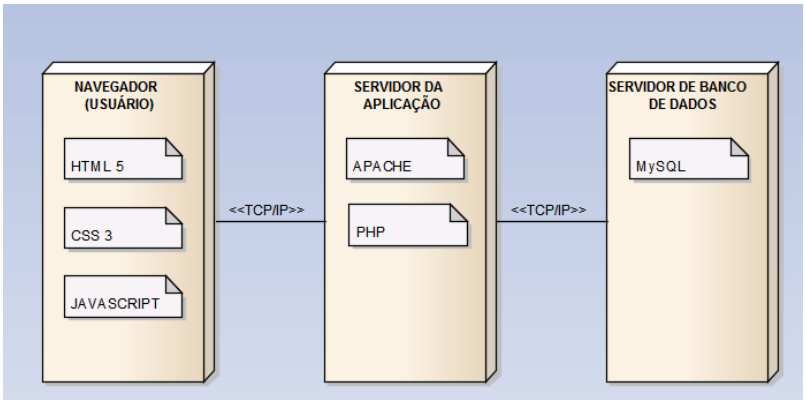

Figura 52 - Diagrama de Implantação
Fonte: Próprio autor.

4.5 Implementação

As imagens a seguir representam as principais telas e componentes do sistema, com seus respectivos códigos fontes ou trechos importantes.

4.5.1 Página inicial

A imagem a seguir é referente à página inicial do sistema.

Figura 53 - Página inicial
Fonte: Próprio autor.

Nessa primeira imagem é possível visualizar os itens cabeçalho, *slide* de imagens informativas, componente de busca e lista de categorias com as principais subcategorias.

Abaixo do *slide* existe uma opção de busca rápida, onde o usuário pode informar um termo de busca, a categoria e a subcategoria, em que deseja realizar a busca.

A imagem a seguir contém os componentes que constam na segunda metade da página.

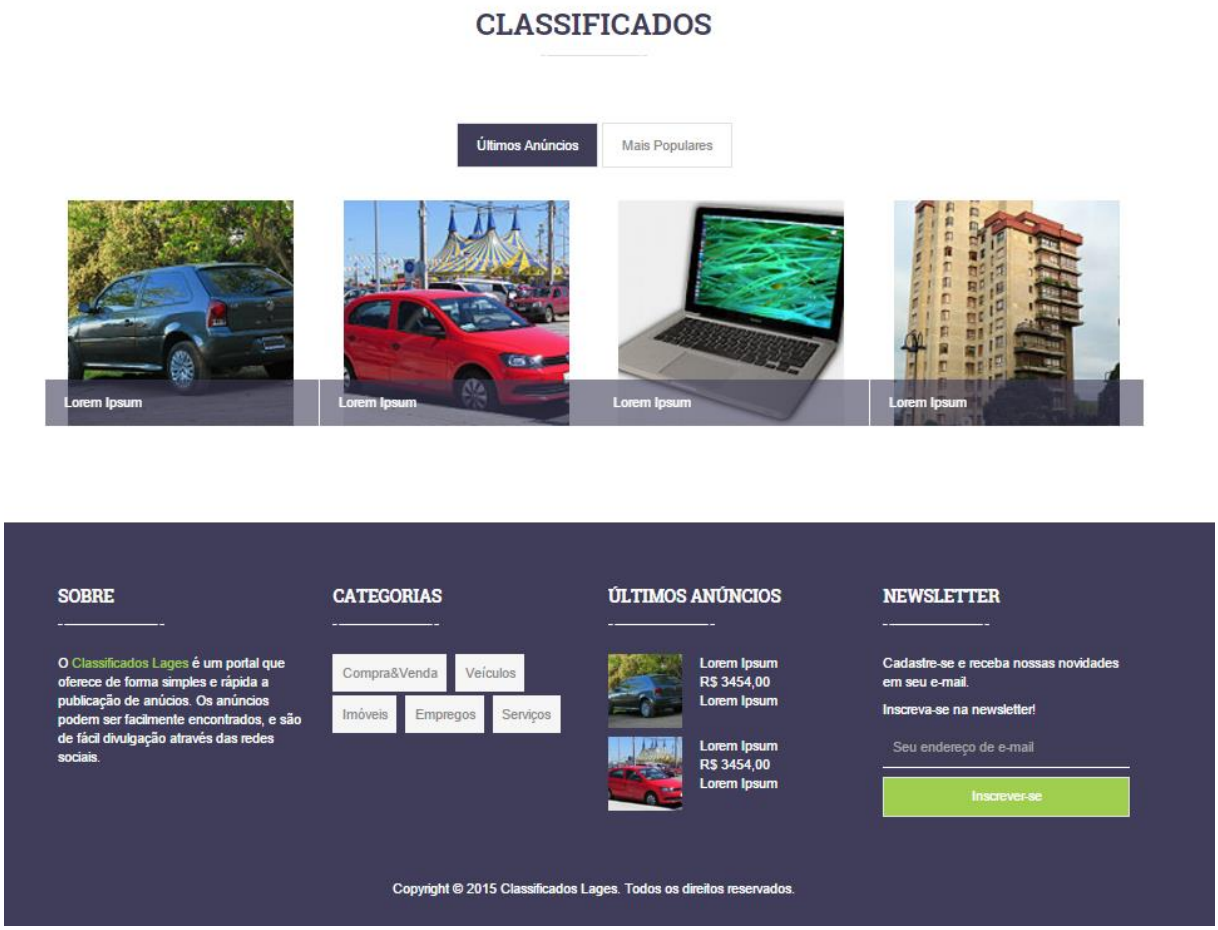

Figura 54 – Rodapé da página inicial
Fonte: Próprio autor.

Nessa segunda metade é possível visualizar a lista de anúncios (últimos e mais populares) e o rodapé da página.

A imagem a seguir é o trecho de código fonte responsável por obter as informações necessárias para a página inicial.

```php
public function index()
{
    $anunciosPopulares = \AnuncioHelper::getAnunciosPopulares(12);
    $anunciosRecentes = \AnuncioHelper::getUltimosAnuncios(12);

    $subcategoriasCompraVenda = Subcategoria::with(['anuncios'])->Geral()
    ->take(5)->get();
    $subcategoriasVeiculos = Subcategoria::with(['anuncios'])->Veiculo()
    ->take(5)->get();
    $subcategoriasImoveis = Subcategoria::with(['anuncios'])->Imovel()
    ->take(5)->get();
    $subcategoriasEmpregos = Subcategoria::with(['anuncios'])->Emprego()
    ->take(5)->get();
    $subcategoriasServicos = Subcategoria::with(['anuncios'])->Servico()
    ->take(5)->get();

    $urlCategoriaCompraVenda = action('CategoriasController@geral');
    $urlCategoriaVeiculos = action('CategoriasController@veiculo');
    $urlCategoriaImoveis = action('CategoriasController@imovel');
    $urlCategoriaEmpregos = action('CategoriasController@emprego');
    $urlCategoriaServicos = action('CategoriasController@servico');

    $subcategorias = Subcategoria::geral()->lists('nome', 'id');
    array_unshift($subcategorias, "Subcategoria");

    return view('site.index', compact(['subcategorias', 'anunciosPopulares',
        'anunciosRecentes', 'subcategoriasCompraVenda', 'subcategoriasVeiculos',
        'subcategoriasImoveis', 'subcategoriasEmpregos', 'subcategoriasServicos',
        'urlCategoriaCompraVenda', 'urlCategoriaVeiculos', 'urlCategoriaImoveis',
        'urlCategoriaEmpregos', 'urlCategoriaServicos']));
}
```

Figura 55 – Código fonte página inicial
Fonte: Próprio autor.

4.5.2 Cabeçalho

No cabeçalho da página ficam os *links* de acesso às redes sociais, além de botões que podem variar, de acordo com o estado de autenticação do usuário. Caso ele esteja autenticado, são mostrados os botões “minha conta” e “sair”, caso contrário são mostrados os botões “Cadastre-se” e “Entrar”.

Figura 56 – Topo da página inicial
Fonte: Próprio autor.

A imagem a seguir é um trecho de código fonte responsável por criar dinamicamente esse cabeçalho conforme o estado de autenticação do usuário:

```
@if(Auth::check())
<div class="col-sm-6 responsive-width-top">
    <div class="links text-right">
        <a href="{{ action("AnunciosController@create") }}"><i
        class="fa fa-bullhorn"></i> Anunciar</a>
        <a href="{{ action("UsuariosController@conta") }}">
        Minha Conta</a>
        <a class="sair" href="{{ action("Auth\AuthController@
        getLogout") }}"><i class="fa fa-sign-out"></i> Sair</a>
    </div>
</div>
@else
<div class="col-sm-6 responsive-width-top">
    <div class="links text-right">
        <a href="{{ action("AnunciosController@create") }}"><i
        class="fa fa-bullhorn"></i> Anunciar</a>
        <a href="{{ action("Auth\AuthController@getRegister") }}
        ">Cadastre-se</a>
        <a href="{{ action("Auth\AuthController@getLogin") }}"><
        i class="fa fa-sign-in"></i> Entrar</a>
    </div>
</div>
@endif
```

Figura 57 – Topo da página inicial
Fonte: Próprio autor.

4.5.3 Rodapé

No rodapé da página fica uma descrição resumida do sistema. Além disso ficam *links* de acesso fácil para as páginas de cada categoria. Ainda no rodapé estão os dois últimos anúncios cadastrados e um campo para que o usuário possa se inscrever na *newsletter* do sistema.

Figura 58 – Rodapé da página inicial
Fonte: Próprio autor.

A imagem a seguir é um trecho de código fonte responsável por criar dinamicamente esse cabeçalho conforme o estado de autenticação do usuário:

```
@if(Auth::check())
<div class="col-sm-6 responsive-width-top">
    <div class="links text-right">
        <a href="{{ action("AnunciosController@create") }}"><i
        class="fa fa-bullhorn"></i> Anunciar</a>
        <a href="{{ action("UsuariosController@conta") }}">
        Minha Conta</a>
        <a class="sair" href="{{ action("Auth\AuthController@
        getLogout") }}"><i class="fa fa-sign-out"></i> Sair</a>
    </div>
</div>
@else
<div class="col-sm-6 responsive-width-top">
    <div class="links text-right">
        <a href="{{ action("AnunciosController@create") }}"><i
        class="fa fa-bullhorn"></i> Anunciar</a>
        <a href="{{ action("Auth\AuthController@getRegister") }}
        ">Cadastre-se</a>
        <a href="{{ action("Auth\AuthController@getLogin") }}"><
        i class="fa fa-sign-in"></i> Entrar</a>
    </div>
</div>
@endif
```

Figura 59 – Topo da página inicial
Fonte: Próprio autor.

4.5.4 Conta do usuário

A imagem a seguir é referente à página “minha conta”. Nela tem os dados do usuário autenticado e uma opção para ir até a página de alteração dos dados. Logo abaixo é mostrada uma lista de anúncios ativos do usuário, onde ele pode gerenciá-los.

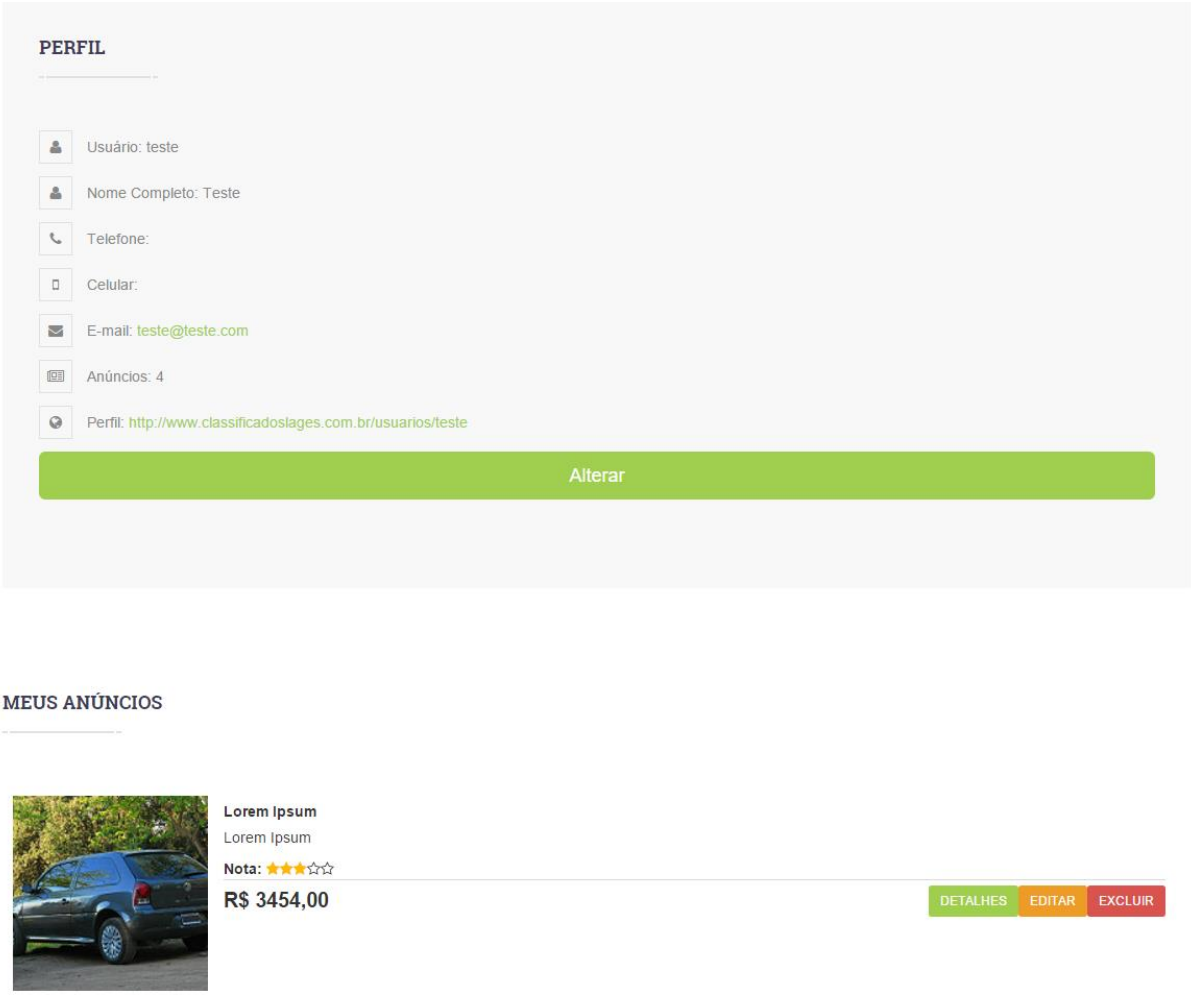

Figura 60 – Página conta do usuário
Fonte: Próprio autor.

A imagem a seguir é um trecho de código fonte responsável por obter a lista de anúncios ativos do usuário autenticado.

```
public function conta()
{
    $anuncios = Auth::user()->Anuncios()->ativo()->orderBy('id','desc')->paginate(5);
    return view('site.usuarios.minha-conta', compact(['anuncios']));
}
```

Figura 61 – Código fonte “meus anúncios”
Fonte: Próprio autor.

4.5.5 Publicar anúncio

A imagem a seguir é referente à página de criação e edição de anúncios. O cabeçalho e o rodapé são semelhantes a página inicial. Os campos são dinâmicos, de acordo com a categoria e subcategorias selecionadas, os campos necessários são mostrados e desnecessários ocultados. O exemplo abaixo se refere ao cadastro de um anúncio de motocicleta.

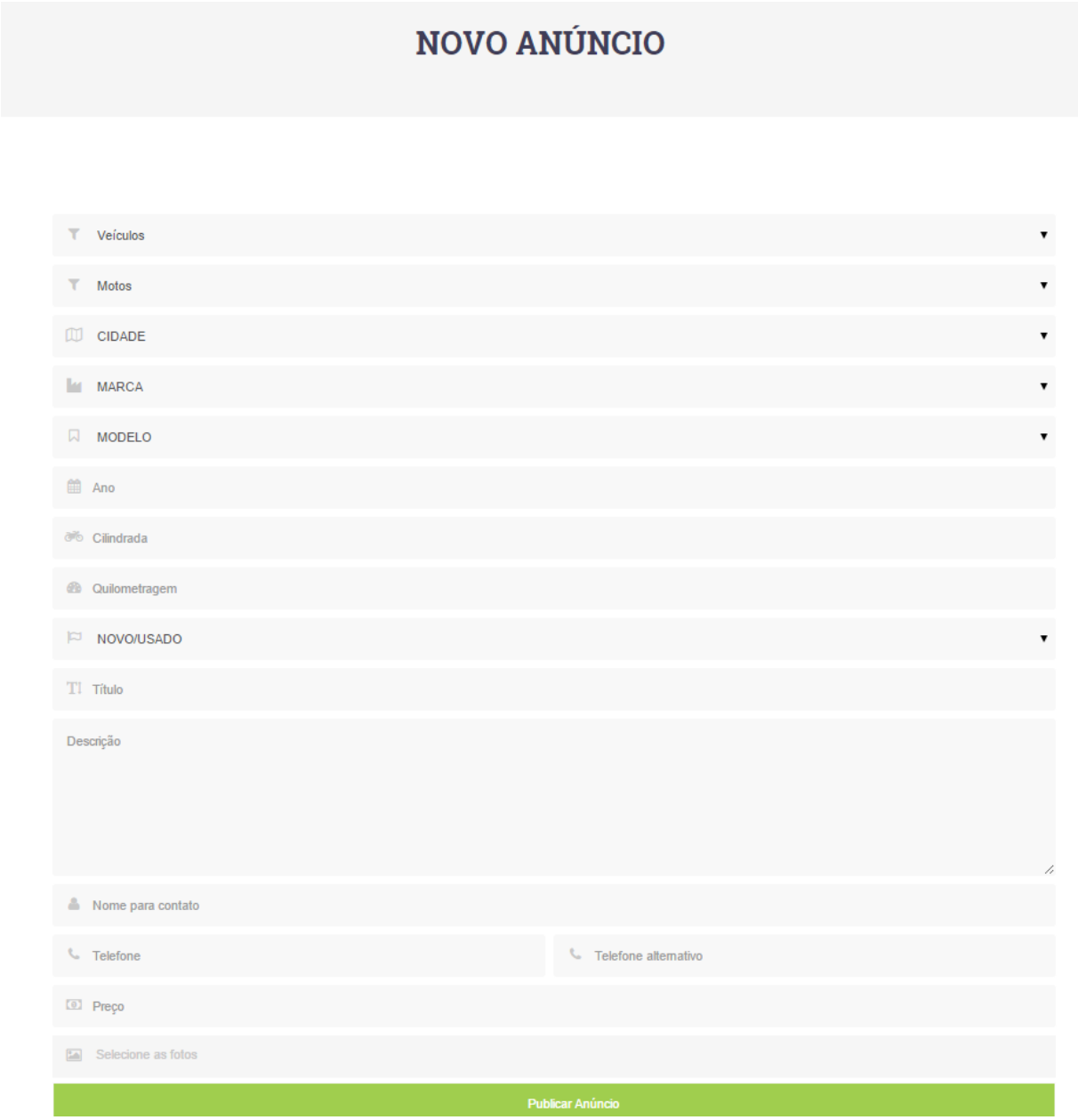

Figura 62 – Página de anúncio.
Fonte: Próprio autor.

A imagem a seguir é um trecho de código fonte responsável por carregar a página e gravar o anúncio.

```php
public function create()
{
    $anuncio = new Anuncio;

    $subcategorias = Subcategoria::geral()->lists('nome', 'id');
    array_unshift($subcategorias, "Selecione uma subcategoria...");
    $cidades = Cidade::lists('nome', 'id');
    array_unshift($cidades, "CIDADE");
    $categorias = array('geral'=>'Compra&Venda', 'veiculo'=>'Veículos',
        'imovel'=>'Imóveis', 'emprego'=>'Empregos','servico'=>'Serviços');
    $estados = array('0'=>'NOVO/USADO', 'novo'=>'Novo', 'usado'=>'Usado');

    return view('site.anuncios.create',compact(['anuncio', 'subcategorias',
     'categorias', 'cidades', 'estados']));
}

public function store(AnunciosRequest $request)
{
    $anuncio = \AnuncioHelper::salvarAnuncio($request,
        $request->input('categoria'));
    return Response::json('success', 200);

}
```

Figura 63 – Código fonte da criação de um anúncio
Fonte: Próprio autor.

Antes de chegar ao método “store”, os dados são validados na classe “AnunciosRequest”. O trecho de código fonte abaixo é responsável por validar as informações básicas do anúncio. Além dessas regras existem regras específicas para cada categoria de anúncio.

```php
public function basico()
{
    $categoria = $this->input('categoria');
    $regras = [
        'subcategoria_id'=>'required|integer|exists:subcategorias,id,categoria,'
        .$categoria,
        'cidade_id'=>'required|integer|exists:cidades,id',
        'categoria'=>'required|in:geral,veiculo,imovel,emprego,servico',
        'titulo'=>'required|string|min:2|max:255',
        'descricao'=>'required|string|min:10|max:500',
        'nome_contato'=>'required|string|min:2|max:255',
        'telefone_contato'=>'required|
        regex:[^\([1-9]{2}\) [2-9][0-9]{3,4}\-[0-9]{4}$]',
        'telefone_contato_alternativo'=>
        'regex:[^\([1-9]{2}\) [2-9][0-9]{3,4}\-[0-9]{4}$]',
    ];
    return $regras;

}
```

Figura 64 – Código fonte da validação básica de um anúncio.
Fonte: Próprio autor.

4.5.6 Visualizar anúncio

A imagem a seguir é referente à página de visualização detalhada de anúncio. O cabeçalho e o rodapé são semelhantes a página inicial. Os campos são dinâmicos, de acordo com a categoria e subcategorias do anúncio. Mais abaixo existe a opção de enviar uma mensagem diretamente ao autor do anúncio.

O exemplo abaixo se refere a visualização de um anúncio de um videogame.

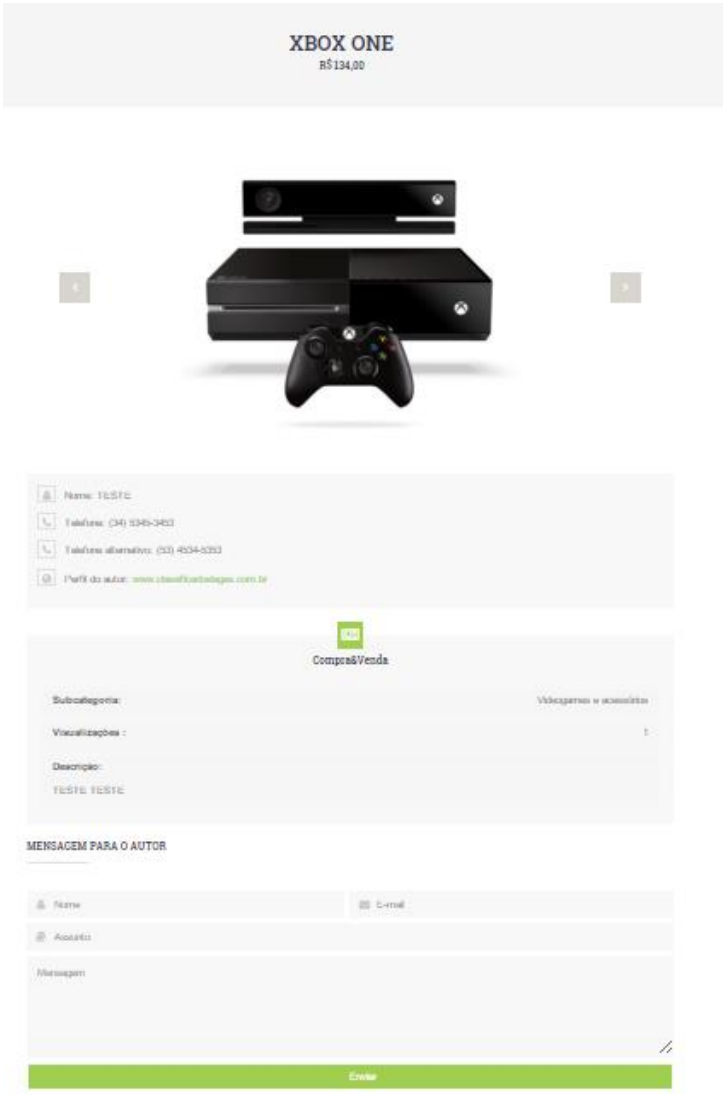

Figura 65 – Visualizar anúncio.
Fonte: Próprio autor.

A imagem a seguir é um trecho de código fonte responsável por carregar a página de visualização do anúncio.

```php
public function showAnuncio($id)
{
    $anuncio = Anuncio::ativo()->with(['usuario', 'fotos',
        'anuncioveiculo', 'anuncioimovel', 'anuncioemprego', 'anuncioservico'])
    ->find($id);
    if(!is_null($anuncio))
    {
        $anuncio->visualizacoes = $anuncio->visualizacoes+1;
        $anuncio->save();
    }
    return view('site.anuncios.show', compact(['anuncio']));
}
```

Figura 66 – Código fonte visualização do anúncio.
Fonte: Próprio autor.

4.5.7 Página de administração

A imagem a seguir é referente à página de administração. Na barra lateral estão as opções para acessar o gerenciamento de cada uma das entidades. A página abaixo se refere ao gerenciamento de usuários.

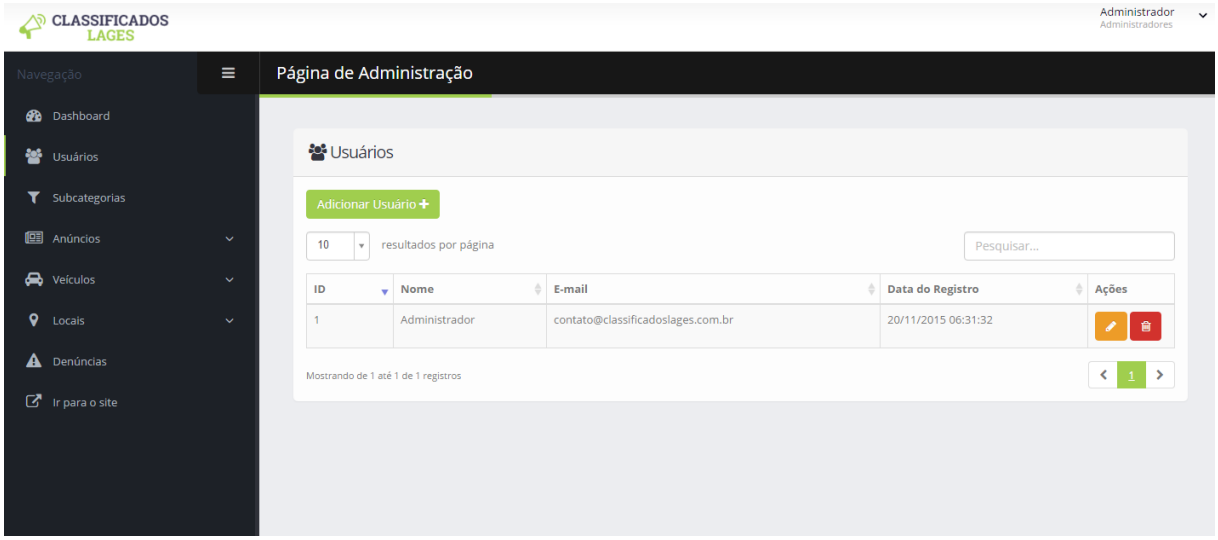

Figura 67 – Página de administração.
Fonte: Próprio autor.

A imagem a seguir é do código fonte responsável pelo gerenciamento de usuários.

```php
<?php namespace App\Http\Controllers\Admin;
use App\Http\Controllers\Controller;
use App\Http\Requests\UsuariosRequest;
use App\Usuario;
use Response;

class UsuariosController extends Controller {

    public function __construct()
    {

    }

    public function create()
    {
        $panelTitle = "Adicionar";
        $usuario = new Usuario;
        return view('admin.usuarios.form',compact(['usuario', 'panelTitle']));
    }

    public function store(UsuariosRequest $request)
    {
        $usuario = $request->all();
        Usuario::create($usuario);
        return Response::json('success', 200);
    }

    public function index()
    {
        $usuarios = Usuario::all();
        return view('admin.usuarios.index', compact('usuarios'));
    }

    public function getAll()
    {
        $usuarios = Usuario::all();
        return Response::json($usuarios);
    }

    public function edit($id)
    {
        $usuario = Usuario::find($id);
        $panelTitle = "Editar";
        return view('admin.usuarios.form',compact(['usuario', 'panelTitle']));
    }

    public function update($id, UsuariosRequest $request)
    {
        $usuarioUpdate = $request->all();
        $usuario = Usuario::find($id);
        $usuario->update($usuarioUpdate);
        return Response::json('success', 200);
    }

    public function destroy($id)
    {
        $usuario = Usuario::find($id);
        $usuario ->delete();
        return Response::json('success', 200);
    }
}
```

Figura 68 – Código fonte gerenciamento de usuários
Fonte: Próprio autor.

5 CONCLUSÃO

O desenvolvimento de software é extremamente importante para entregar a população ferramentas e sistemas úteis que possam ser utilizados no dia a dia. Por esse motivo, este trabalho foi desenvolvido. Com o propósito de entregar um sistema que a população local pudesse se tornar referência no meio de sistemas de classificados já existentes.

No decorrer do trabalho, foram realizadas pesquisas sobre conceitos e tecnologias que seriam utilizadas. Dando embasamento teórico para as etapas mais avançadas.

Com este trabalho, pode-se concluir que um bom planejamento, documentação e implementação seguindo boas práticas, é possível entregar um trabalho estável e com qualidade.

Este trabalho também proporcionou a agregação de conhecimentos possibilitando a aprendizagem e aperfeiçoamento, incentivando a exploração de tecnologias distintas, que culminaram na entrega do trabalho.

REFERÊNCIAS

APPOLINÁRIO, Fábio. **Dicionário de metodologia científica**: um guia para a produção do conhecimento científico. 2. ed. São Paulo: Atlas, 2011. 320p.

BALDUINO, Plínio. **Dominando JavaScript com jQuery**. São Paulo: Casa do Código, 2012. 180p.

BELEM, Thiago. **Gerenciando dependências com o Composer**, 2012. Disponível em: http://blog.thiagobelem.net/gerenciando-dependencias-com-o-composer/.Acesso em: 07/03/2015.

BUSCHMANN, F. et al. **Pattern-Oriented Software Architecture:** a system of patterns. John Wiley & Sons, England, 1996. 467p.

COMPOSER. **Getting Started**, 2015. Disponível em: https://getcomposer.org/doc/00-intro.md. Acesso em: 07/03/2015.

CAELUM. **Desenvolvimento Web com HTML, CSS e JavaScript.** São Paulo: Caelum, 2013. 261p.

DIAS, Oscar. **Laravel: Rotas e Controllers**, 2014. Disponível em: http://magazine.softerize.com.br/tutoriais/php/laravel/laravel-rotas-e-controllers. Acesso em: 20/03/2015.

EIS, Diego et al. **HTML5 e CSS3 com farinha e pimenta.** São Paulo: Tableless, 2012. 218p.

FAYAD, M. E., SCHMIDT, D. C. **Object-oriented Application frameworks. Communications of the ACM**, Vol. 40, 1997. 10 p.

FERRARI, F. A. **Crie banco de dados em MySQL**. São Paulo: Digerati Books, 2010. 123p.

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

GILMORE, W. Jason. **Easy Laravel 5**. Leanpub.com, 2015. 212p.

K19 TREINAMENTOS. **Desenvolvimento Web com HTML, CSS e Javascript.** São Paulo, 2013. 464p.

MAZZA, Lucas. **HTML5 e CSS3 Domine a web do futuro**. São Paulo: Casa do Código, 2012. 195p.

MCLAUGHLIN, B. **Use a Cabeça!:** Iniciação Rápida Ajax. Rio de Janeiro: Alta Books, 2006. 317p.

MYSQL. **MySQL 5.7 Reference Manual**, 2015. Disponível em: http://dev.mysql.com/doc/refman/5.7/en/. Acesso em: 15/01/2015.

LARMAN, Craig. **Utilizando UML e padrões:** uma introdução à análise e ao projeto orientados a objetos e ao Processo Unificado. 3. ed. Porto Alegre: Bookman, 2005. 685p.

OTWELL, Taylor. **Laravel Documentation**, 2015. Disponível em: http://laravel.com/docs/. Acesso em: 10/01/2015.

PRESSMAN, R. **Engenharia de Software**: Uma Abordagem Profissional. 7. ed. São Paulo: McGraw-Hill, 2011. 771p.

REES, Dayle. **Basic Routing,** 2014. Disponível em: http://daylerees.com/codebright/basic-routing. Acesso em: 15/03/2015.

SKVORC, Bruno. **Best PHP Framework for 2015,** 2015. Disponível em: http://www.sitepoint.com/best-php-framework-2015-sitepoint-survey-results/. Acesso em: 28/03/2015.

SOMMERVILLE, Ian. **Engenharia de Software**. 8. ed. São Paulo: Addison-Wesley Brasil, 2007. 552p.

THE PHP GROUP. **O que é o PHP?**, 2015. Disponível em: http://php.net/manual/pt_BR/intro-whatis.php. Acesso em: 10/02/2015.