UNIVERSIDADE DO VALE DO RIO DOS SINOS - UNISINOS
UNIDADE ACADÊMICA DE EDUCAÇÃO CONTINUADA
CURSO DE ESPECIALIZAÇÃO EM DESENVOLVIMENTO DE APLICATIVOS PARA DISPOSITIVOS MÓVEIS

MARCIO GUSTAVO GUSMAO SCHERER

APLICATIVO DETETIVE CIDADÃO
SOCIEDADE E TECNOLOGIA JUNTAS NO COMBATE AO CRIME

PORTO ALEGRE
2017

Marcio Gustavo Gusmão Scherer

APLICATIVO DETETIVE CIDADÃO:

Sociedade e tecnologia juntas no combate ao crime

Trabalho de Conclusão de Curso de Especialização apresentado como requisito parcial para obtenção do título de Especialista pelo Curso de Desenvolvimento de Aplicativos para Dispositivos Móveis da Universidade do Vale do Rio dos Sinos – UNISINOS

Orientador(a): Prof(a). MS. Anderson da Cruz

Porto Alegre

2017

RESUMO

Segurança pública tem se tornado uma grande prioridade para a sociedade e os governantes da maioria das cidades brasileiras, devido aos alarmantes e crescentes índices de crimes de todos os tipos no país. Especificamente no que se trata de roubo e furtos de veículos, a cidade de Porto Alegre e região metropolitana registram índices preocupantes. O combate a esses crimes é especialmente importante, pois além da transgressão em si, muitas vezes os veículos são utilizados para cometer outros delitos. Esse trabalho propõe o desenvolvimento de um aplicativo para dispositivos móveis, chamado Detetive Cidadão, que permite que a própria população auxilie as forças de segurança efetuando denúncias anônimas a respeito de veículos suspeitos nas vias públicas. O diferencial do aplicativo é o fato do mesmo já estar interligado com os sistemas de segurança, emitindo alertas em tempo real para que os agentes de segurança pública possam tomar as ações cabíveis em tempo hábil.

Palavras-chave: Segurança Pública. Tecnologia OCR. OpenALPR. IONIC2.

LISTA DE FIGURAS

LISTA DE SIGLAS

API	Application Programming Interface
DENATRAN	Departamento Nacional de Transito
DETRANRS	Departamento Estadual de Trânsito do Rio Grande do Sul
EJB	Enterprise JavaBeans
EPTC	Empresa Pública de Transporte e Circulação
GPS	Global Position System
JDBC	Java Database Connectivity API
JMS	Java Message Service
PROCEMPA	Companhia de Processamento de Dados do Município de Porto Alegre
PROCERGS	Companhia de Processamento de Dados do Estado do RGS
REST	Representational State Transfer
SERPRO	Serviço Federal de Processamento de Dados
SSPRS	Secretaria de Segurança Pública do Rio Grande do Sul
WEB	World Wide Web

SUMÁRIO

1 INTRODUÇÃO

Segurança pública tem se tornado uma grande prioridade para a sociedade e os governantes da maioria das cidades brasileiras, devido aos alarmantes e crescentes índices de crimes de todos os tipos no país. Especificamente no que se trata de roubo e furtos de veículos, a cidade de Porto Alegre e região metropolitana registram índices preocupantes. O combate a esses crimes é especialmente importante, pois além da transgressão em si, muitas vezes os veículos são utilizados para cometer outros delitos. Em 2015, Porto Alegre registrou a maior taxa de roubo e furto de veículos entre as capitais brasileiras, conforme dados divulgados pelo 10º Anuário Brasileiro da Segurança Pública [G1ASPB]. No primeiro trimestre do ano de 2017, Porto Alegre registrou a marca de 3377 ocorrências de furto e roubo de veículos, quase 38 veículos em média por dia, conforme dados da SSPRS [SSPRGS].

Com o objetivo de diminuir esses índices, a prefeitura de Porto Alegre está investindo pesado em tecnologia. Desde o início do ano, através da PROCEMPA e em parceria com o DETRAN, PROCERGS e os órgãos de segurança pública do estado e do município, está desenvolvendo e constantemente aprimorado um sistema de monitoramento eletrônico de tráfego (chamado ALPR), já em funcionamento, que objetiva principalmente identificar em tempo real veículos trafegando com ocorrências de roubo e furto, bem como os com indícios de clonagem. Uma vez o veículo identificado pelo sistema, em alguma via pública, uma operação é montada pela SSPRS para tentar localizar e recuperar o mesmo. As informações dos veículos são capturadas por câmeras, espalhadas em diversos pontos da cidade.

Com a finalidade de suporte a essa iniciativa, esse trabalho propõe um aplicativo para dispositivos móveis chamado Detetive Cidadão. Esse aplicativo permite o registro e envio de informações sobre veículos suspeitos que estejam circulando ou estacionados na cidade. Permite fotografar a placa do veículo, a partir daí as informações são compartilhadas com o sistema de monitoramento de tráfego (ALPR). Os dados são então analisados pelos agentes públicos, responsáveis por tomar as devidas providências. A ideia do aplicativo é empoderar o cidadão para que ele ajude no combate à violência e à criminalidade, o objetivo da ferramenta é transformar as pessoas em uma fonte de informação valiosa que possa atuar em qualquer lugar, a qualquer tempo, ajudando na recuperação dos veículos e coibindo os crimes. O aplicativo pode ser utilizado também

por agentes públicos (Brigada Militar, EPTC, Polícia Civil, etc) para abordagens e verificações de rotina.

1.1 OBJETIVOS

Desenvolver um aplicativo para dispositivos móveis chamado Detetive Cidadão, que possa ser utilizado pelos cidadãos de Porto Alegre e agentes públicos para fotografar e alertar as autoridades a respeito de veículos suspeitos de roubo / furto, abrindo assim um canal direto e totalmente anônimo entre o cidadão e as autoridades de segurança pública. O objetivo principal do aplicativo é reduzir a criminalidade na cidade, coibindo o roubo e furto de veículos.

De forma a complementar o objetivo geral proposto, apresentam-se os seguintes objetivos específicos:

- Detalhar o funcionamento do sistema de monitoramento de tráfego desenvolvido pela Procempa e integrado com o aplicativo Detetive Cidadão;
- Desenvolver o aplicativo Detetive Cidadão, detalhar o seu funcionamento cobrindo os perfis de Cidadão e Agente Público, arquitetura tecnológica e frameworks / tecnologias e ferramentas utilizadas no desenvolvimento da solução;
- Demonstrar o resultado de um estudo de aplicativos similares ao Detetive Cidadão, já existentes no mercado, bem como apontar as vantagens do mesmo sobre esses aplicativos;
- Demonstrar os resultados da integração do Detetive Cidadão com o aplicativo para dispositivos móveis EUFACOPOA e a colaboração do mesmo na recuperação de veículos e redução dos índices de criminalidade;

1.2 ORGANIZAÇÃO DO TRABALHO

Buscando a compreensão dos objetivos a que se propõe este estudo, é indicada a seguir a estrutura do presente trabalho:

O Capítulo 2 apresenta em detalhes o funcionamento do Sistema de Monitoramento de Tráfego ALPR, integrado com o Detetive Cidadão;

O Capítulo 3 cobre o funcionamento, arquitetura tecnológica e os detalhes de desenvolvimento do aplicativo do Detetive Cidadão. Apresenta também um estudo de comparação e benefícios do aplicativo com outros aplicativos similares no mercado; O Capítulo 3 aborda também os resultados obtidos na integração do Detetive Cidadão com o aplicativo para dispositivos móveis EUFACOPOA, apresentando os impactos positivos que a incorporação do aplicativo trouxe, assim como as estatísticas de veículos que foram analisados e recuperados através do aplicativo.

No Capítulo 4 serão tratadas as considerações finais desse trabalho.

2 SISTEMA DE MONITORAMENTO DE TRÁFEGO (ALPR)

O sistema de monitoramento de tráfego ALPR, desenvolvido pela PROCEMPA, entre outros objetivos, visa identificar a placa dos veículos em circulação nas avenidas da cidade, verificando, em tempo real, se os veículos possuem registros de furto ou roubo. O sistema utiliza a tecnologia de OCR (Optical character recognition) para detectar os caracteres das placas dos veículos, para que o veículo possa então ser consultado junto a base de dados do DETRAN. De acordo com Chaudhuri et. al. [OCRD16], o reconhecimento ótico de caracteres é uma das áreas mais populares de pesquisa voltadas a reconhecimento de padrões. Segundo os autores, a técnica consiste basicamente em traduzir caracteres impressos ou inseridos em imagens em caracteres codificados (textos) que possam ser reconhecidos por um sistema de computador. Após o veículo ser consultado na base de dados, caso ocorram ocorrências de roubo ou furto, um alerta é emitido na interface do sistema para que possa ser analisado. O alerta contém todas as informações para que o veículo possa eventualmente ser encontrado e recuperado pelos órgãos públicos de segurança, como por exemplo: a placa, o horário que foi capturado, o modelo e cor do veículo e onde foi encontrado (Figura 1). Alguns dados nas figuras que serão apresentadas foram ofuscados para manter o sigilo das informações.

Figura 1 – Alerta no sistema de Monitoramento de Tráfego ALPR

O processo de análise de um alerta consiste em efetuar a triagem do mesmo, que envolve verificar se a placa do veículo condiz com o OCR gerado pelo sistema. Esse processo pode envolver uma confirmação da ocorrência em outros sistemas também, a critério da operação feita pelo órgão público de segurança. Após a confirmação do alerta, o próximo passo é verificar se existe uma viatura próximo ao local e despachar a ocorrência no sistema, esse processo visa a tentativa de captura / recuperação do veículo suspeito. Por fim, o sistema permite que o agente público finalize o alerta, informando para o sistema qual foi o desfecho da ocorrência, por exemplo, se o veículo foi recuperado, se não foi localizado, etc. O resultado da operação é importante principalmente para alimentação dos sistemas de inteligência da SSPRS, assim como os locais onde os veículos são localizados. Essas informações contribuem para que a secretaria possa realizar operações nos principais focos de ocorrência (elaboração de mapas de calor, por exemplo). Uma visão completa do sistema pode ser vista na Figura 2.

Figura 2 – Sistema de Monitoramento de Tráfego ALPR

3 APLICATIVO PARA DISPOSITIVOS MÓVEIS DETETIVE CIDADÃO

O aplicativo Detetive Cidadão visa empoderar a população, fornecendo a possibilidade aos cidadãos de colaborar com as autoridades por meio de denúncias envolvendo veículos suspeitos estacionados ou trafegando nas vias da cidade. A ideia do aplicativo é que o próprio cidadão ajude no combate à violência e à criminalidade, o objetivo da ferramenta é transformar as pessoas em uma fonte de informação valiosa que possa atuar em qualquer lugar, a qualquer tempo, ajudando na recuperação dos veículos e coibindo a criminalidade. A denúncia, feita de forma totalmente anônima, inicia com o usuário baixando o aplicativo e entrando no mesmo com seu perfil de cidadão (Figura 3). O próximo passo é tirar uma foto do veículo considerado suspeito. O cidadão - que conhece principalmente o movimento no seu bairro e pode facilmente perceber veículos que nunca foram vistos circulando ou pernoitando nas redondezas - após a captura da foto, faz a conferência da placa do veículo e do endereço antes de submeter as informações para análise.

Figura 3 – Interface de Autenticação do Aplicativo

Essas informações, placa e endereço, são capturadas automaticamente pelo aplicativo através da foto do veículo (OCR) e a posição GPS do usuário (Figura 4). Uma vez as informações conferidas, o cidadão deve submeter o veículo para análise.

Figura 4 – Interface de Captura de Informações do Veículo

Na interface final do aplicativo, o mesmo avisa o usuário se o veículo possui alguma ocorrência de Roubo/Furto ou se está em situação legal (Figura 5).

Figura 5 – Aviso de Ocorrência de Roubo/Furto

Se o veículo possuir alguma ocorrência, o aplicativo avisa o usuário para não tomar nenhuma ação e afastar-se do veículo, para sua própria segurança. Nesse caso, o próprio aplicativo cria um alerta no sistema de monitoramento de tráfego ALPR, automaticamente. O alerta é recebido pelos órgãos públicos que são responsáveis por fazer a triagem, a análise do caso e dependendo da situação enviar as autoridades para averiguação/captura do veículo. O alerta é tratado pelo sistema ALPR e pelos agentes públicos como se fosse qualquer outro alerta de roubo/furto, com a mesma prioridade dos alertas gerados pela captura de imagens fornecidas pelas câmeras de monitoramento da cidade (Figura 6).

Figura 6 – Alerta automaticamente criado no sistema ALPR

O aplicativo também pode ser acessado por agentes públicos (agentes de tráfego, de segurança etc.) mediante autenticação do usuário. Nesse caso, o aplicativo gera alerta

e retorno ao usuário mediante qualquer ocorrência pertencente ao veículo, não somente ocorrências de furto ou roubo. O agente é avisado se o veículo possuir qualquer restrição a circulação, tiver licenciamento vencido, ocorrência de busca e apreensão ou ainda se estiver baixado na base de dados do DETRAN.

3.1 ARQUITETURA

O aplicativo para dispositivos móveis Detetive Cidadão foi construído com o suporte do framework IONIC 2. De acordo com Phan [IONIC16] dentre as principais vantagens do framework, está a possibilidade de se programar utilizando HTML 5, CSS, Angular e Type Script, gerando as versões para IOS e Android automaticamente por meio do próprio framework. Utilizando-se o Cordova, é possível acessar recursos nativos do dispositivo, como GPS e câmera, recursos amplamente utilizados pela solução. O aplicativo se comunica com os serviços através de uma API REST, criada com componentes Java/JEE e implantada em um servidor de aplicação Redhat Wildfly. A API Java contém os serviços necessários para a aplicação, como o serviço de OCR, o serviço de autenticação do usuário (que autentica o usuário no AD através do protocolo LDAP) e o serviço da PROCERGS que analisa se o veículo possui ou não ocorrências na base de dados do DETRAN. O serviço de OCR é feito por meio da utilização do framework Openalpr. A arquitetura completa da solução pode ser melhor visualizada na Figura 7.

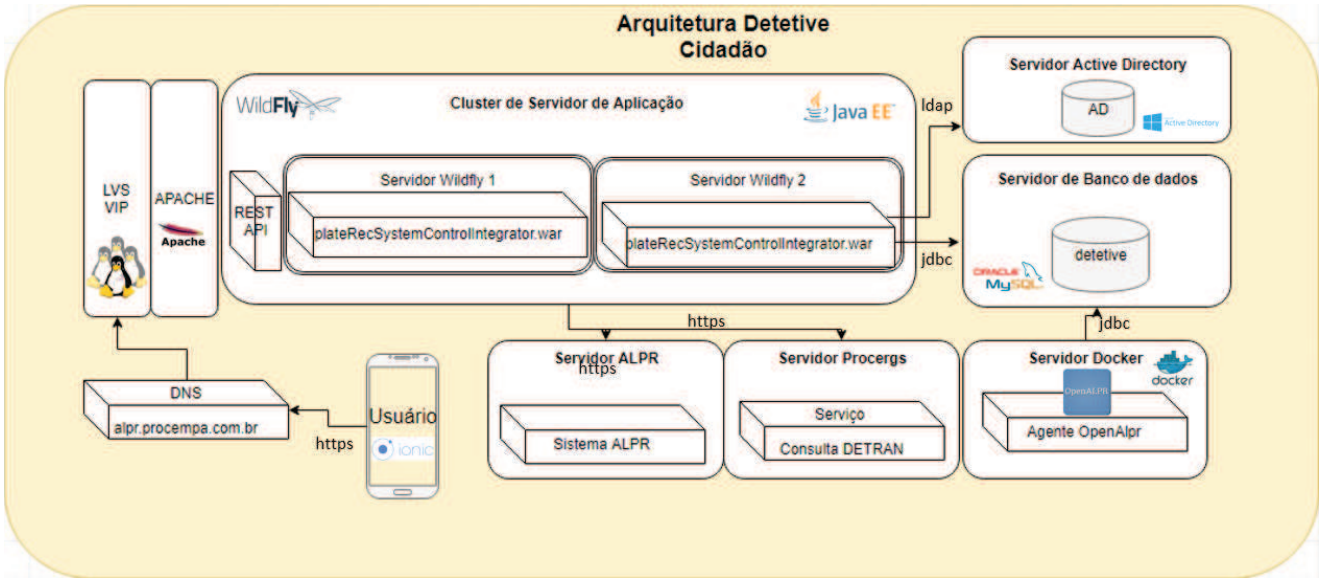

Figura 7 – Diagrama de arquitetura

3.2 FRAMEWORKS, LINGUAGENS E TECNOLOGIAS UTILIZADAS

Essa seção apresenta os detalhes técnicos da solução Detetive Cidadão, incluindo os frameworks, as tecnologias e linguagens utilizadas na construção dos serviços de suporte (*backend*) e no desenvolvimento do próprio aplicativo.

3.2.1 **IONIC2**

No que se refere à desenvolvimento mobile, o *framework* IONIC vem ganhando bastante visibilidade e vem sendo com frequência a escolha dos desenvolvedores. O mundo mobile é fragmentado, com várias plataformas, *frameworks* e tecnologias. O IONIC foi criado para suprir essa lacuna. A ideia principal é permitir que os desenvolvedores construam aplicações essencialmente nativas (*native-feeling)* utilizando tecnologias WEB como HTML, CSS e AngularJS. Utilizando os plug-ins do Cordova, é possível até mesmo acessar recursos nativos do dispositivo [IONIC16]. O aplicativo Detetive Cidadão foi desenvolvido com o suporte do IONIC2, utilizando o Cordova para acesso aos recursos nativos necessários, como a câmera para tirar a foto do veículo a ser averiguado e o GPS, para obter a localização do usuário. O endereço da localização do usuário é obtido através de um serviço REST, desenvolvido em Java e implantado no servidor de aplicação Wildfly. Esse serviço, quando requisitado, acessa o serviço de localização de endereços do Google [GOGAPI], o servico recebe as coordenadas de geo-posicionamento passadas pelo dispositivo do usuário e retorno o endereco relativo às coordenadas.

3.2.2 **OpenALPR**

O OpenALPR é um *framework opensource* que fornece suporte à análise de imagens e vídeos de veículos. Por meio da imagem do veículo, com a placa nítida, o *framework* pode fornecer informações de OCR da placa (conversão da imagem em caracteres), o modelo e a cor do veículo (funcionalidades ainda em caráter restrito e experimental). O *framework* é escrito em C++, mas possui conectores para diversas linguagens de programação, sendo possível, assim, integrá-lo também com Java, C#, Node.js e Python [OPALPR]. O aplicativo Detetive Cidadão utiliza o OpenALPR para tentar extrair os caracteres da placa do veículo, para que a mesma possa ser analisada junto a base de dados do DETRAN. O OpenALPR, é executado na arquitetura em um Container Docker, com todas as dependências e configurações necessárias para execução do

framework. A utilização do Docker engloba o OpenALPR, suas várias configuração e dependências em um ambiente isolado, facilitando assim sua manutenção, migração e analise de problemas (*troubleshooting*).

3.2.3 **Active Directory (AD) e LDAP**

O Active Directory é uma implementação de serviço de diretório no protocolo LDAP que armazena informações sobre objetos em rede de computadores e disponibiliza essas informações a usuários e administradores desta rede. O AD é bastante utilizado como repositório único de usuários fornecendo uma forma simples e integrada de autenticação que pode ser usada por todos os sistemas de uma mesma corporação. Dessa forma, os usuários podem possuir uma conta e senha única para acesso a esses sistemas. A comunicação com o AD é geralmente feita por meio do protocolo LDAP, padrão de indústria para acessar e manter serviços de informação de diretório distribuído sobre uma rede. O aplicativo utiliza o AD e o LDAP para autenticar o usuário, quando o mesmo entra na aplicação utilizando o perfil do agente público. Nesse caso, o aplicativo acessa um serviço REST para autenticação, construído com o Java EE e implantado em um servidor de aplicação Wildfly. Esse serviço, por sua vez, autentica o usuário junto à base de dados Active Directory da Prefeitura de Porto Alegre, através do uso do protocolo LDAP.

3.2.4 **Java Enterprise Edition**

O Java Enterprise Edition é uma plataforma de programação para construção de aplicações que podem ser implantadas em servidores que suportem a linguagem Java. O JEE possui uma série de especificações, componentes e recursos extremamente uteis para o desenvolvimento de aplicações coorporativas, robustas e altamente escaláveis [JEE713]. A quantidade de recursos do JEE é bastante ampla, sendo que vamos focar nos recursos que foram utilizados para criação dos serviços de suporte (*backend*) da aplicação Detetive Cidadão. A aplicação war (*web application archive*) plateRecSystemControlIntegrator, implantada no servidor de aplicação Wildfly, contém uma API REST - construída com o suporte do JAXRS - que é acessada pela aplicativo para analisar a imagem do veículo e localizar, baseado na posição GPS do usuário, o endereço do mesmo. A API também provê suporte para a análise da placa do veículo junto ao serviço do DETRAN e para criação do alerta no Sistema de Monitoramento de Tráfego

ALPR, no caso de o veículo possuir alguma ocorrência de roubo ou furto. A API REST, quando acessada, delega o processamento das chamadas para um EJB *Stateless*, componente útil da arquitetura JEE que conta com todo suporte a acesso de recursos do servidor de aplicação (como mensageria JMS, e *datasources*) e controles como o de segurança e transação. O acesso ao banco de dados ocorre através da API JDBC do Java, a imagem é persistida no banco para que o processo de reconhecimento da placa (sendo executado em um container Docker) possa ser realizado. Existe na API REST também um serviço específico para que o usuário possa se autenticar na aplicação, esse serviço utiliza o protocolo LDAP para que o usuário possa ser autenticado no AD da empresa no caso do aplicativo estar sendo acesso pelo perfil agente público.

3.2.5 **Tecnologia de Container**

Para encapsular o framework OpenALPR foi utilizado a tecnologia de Container Docker. O Docker é um projeto de código aberto que automatiza a implantação de aplicativos dentro de recipientes de software, fornecendo uma camada adicional de abstração e automação de virtualização de sistema a nível operacional em Linux, Mac OS e Windows. Esses recipientes são chamados no Docker de imagens e containers.
Vitalino et. al. [DOCK16] coloca que um container é um agrupamento de uma aplicação junto com suas dependências, que compartilham o kernel do sistema operacional do host, ou seja, da máquina (seja ela virtual ou física) onde está rodando. Containers são similares a maquinas virtuais, porém mais leves, visto que as imagens (as estruturas dos containers) geralmente são bem enxutas e carregam somente o essencial para que a aplicação em questão possa ser executada. Outra grande vantagem, além do desempenho, é a portabilidade. Não somente a aplicação em questão fica encapsulada, mas todo o ambiente necessário para que a mesma execute corretamente, assim, a aplicação e todo o ambiente são idênticos em tempo de desenvolvimento / testes e em produção. No aplicativo Detetive Cidadão, o Docker é utilizado para encapsular o OpenALPR e todas as suas várias dependências e configurações. O *framework*, a aplicação e todos os outros recursos necessários são isolados em uma Imagem docker, posteriormente enviada a um registro Docker. Dessa forma, ao invés de instalar todo o framework e configurá-lo em cada ambiente, se faz necessário somente baixar a versão específica da imagem e executá-la, gerando assim um container, onde efetivamente o framework OpenALPR e a aplicação serão executados. O container no caso, faz de tempos em tempos uma

varredura no banco para verificar se existem imagens de veículos a ser analisadas. Para cada imagem, o container/aplicação executa o OpenALPR para tentar capturar o OCR, o OCR é então salvo no banco e fica pronto para ser retornado ao usuário.

3.3 DESENVOLVIMENTO DO APLICATIVO

Essa seção visa descrever alguns dos principais componentes criados no framework IONIC para geração do aplicativo para dispositivo móvel Detetive Cidadão. Para cada tela será abordado o código principal da interface (*.html*) e a respectiva classe de manipulação de dados (*.ts*). No final da seção, será mostrado o código da classe loginProvider, responsável por encapsular a codificação de acesso aos serviços externos. Os recursos não serão explicados em detalhes, assim como o código da classe dados.ts e da última interface do aplicativo por critérios de simplicidade, para maiores informações o site oficial da IONIC pode ser acessado em [IONICFW]. Alguns trechos de código menos importantes também foram removidos por critérios de simplificação.

3.3.1 **Interface inicial de Autenticação**

A interface é bastante simples, como pode ser visto na Figura 8, possui somente um botão de acesso do cidadão ao sistema e um formulário com os dados para autenticação, para o caso de o acesso ser feito através do perfil do agente público.

```
1.  <ion-header>
2.  <ion-navbar>
3.     <ion-title text-center>Detetive Cidadão</ion-title>
4.  </ion-navbar>
5.  </ion-header>
6.  <ion-content padding>
7.  <ion-row >
8.     <ion-col text-center>
9.        <h1> Detetive Cidadão</h1>
10.    </ion-col>
11. </ion-row>
12. <ion-row class="logo-row">
13.    <ion-col></ion-col>
14.    <ion-col width-100>
15.       <img src="assets/brasao.png" />
16.    </ion-col>
17.    <ion-col></ion-col>
```

```
18. </ion-row>
19. <form (ngSubmit)="login()" #registerForm="ngForm">
20. <ion-row>
21.    <ion-col class="signup-col">
22.       <button ion-button class="submit-btn" full type="submit">Entrar como Cidadão</button>
23.    </ion-col>
24. </ion-row>
25. <ion-card>
26.    <ion-card-content>
27.       <ion-list no-lines>
28.          <ion-item>
29.             <ion-label stacked>Usuário</ion-label>
30.             <ion-input type="text" placeholder="Usuário" name="usuario"
31.             [(ngModel)]="usuario"></ion-input>
32.          </ion-item>
33.          <ion-item>
34.             <ion-label stacked>Senha</ion-label>
35.             <ion-input type="password" placeholder="Senha" name="senha" [(ngModel)]="senha"></ion
36.             input>
37.          </ion-item>
38.       </ion-list>
39.       <ion-row>
40.          <ion-col class="signup-col">
41.            <button ion-button class="submit-btn" full type="submit" [disabled]="!usuario  != '
42.            || !senha != ''">Entrar como Agente Público</button>
43.          </ion-col></ion-row>
44.       </ion-card-content>
45.    </ion-card>
46. </form>
47. </ion-content>
```

Figura 8 – Código ionic da interface login.html

A classe de suporte da interface pode ser vista na Figura 9. A importação dos componentes na linha 2 é necessária para possibilitar a navegação entre as páginas da aplicação. Nas linhas 3 e 4, são importadas respectivamente a classe de acesso a dados remotos (serviços) de autenticação e a página de análise de veículos. A classe de serviços será explicada nas próximas seções, é basicamente para validar o acesso do agente público, quando a aplicação é acessada nesse perfil. O método “login” (linha 33), pode ser analisado para verificar como a autenticação é feita. O recurso “Storage“ do Ionic, é utilizado para persistir a informação do usuário, após ser autenticado, o código pode ser visto na linha 25. Uma vez que o usuário entra com perfil de cidadão ou autentica com as

credenciais de agente público, a aplicação direciona a interface para a página de análise de veículos (linha 39).

```
1.  import { Component } from '@angular/core';
2.  import { NavController, NavParams, AlertController, LoadingController, Loading } from 'ionic-
    angular';
3.  import { LoginProvider } from '../../providers/login';
4.  import { AnalisePage } from '../analise/analise';
5.  import { Storage } from '@ionic/storage';
6.  import * as numeral from 'numeral';
7.  @Component({
8.     selector: 'page-login',
9.     templateUrl: 'login.html'
10. })
11.
12. export class LoginPage {
13.    public usuario:string;
14.    public senha:string;
15.    public perfil:string;
16.    private loading: Loading;
17.
18. constructor(public navCtrl: NavController,
19.    public navParams: NavParams,
20.    private loginService:LoginProvider,
21.    private alertCtrl: AlertController,
22.    private loadingCtrl: LoadingController,private storage: Storage) {}
23.
24. ionViewDidLoad() {
25.    this.storage.ready().then(() => {
26.    this.storage.get('usuario').then((val) => {
27.    if (val) {
28.       this.navCtrl.setRoot(AnalisePage, { usuario: val });
29.    }})
30. });
31. }
32.
33. public login(){
34.    this.showLoading();
35.    if (!this.usuario  || this.usuario == '') {
36.    this.loading.dismiss();
37.    this.navCtrl.setRoot(AnalisePage, { user: this.usuario });
38. } else {
39.    this.loginService.login(this.usuario, this.senha).subscribe(
40.    result => {
41.    if (result) {
42.       setTimeout(() => {
43.          this.loading.dismiss();
44.          this.storage.set('usuario', this.usuario);
45.          this.navCtrl.setRoot(AnalisePage, { usuario: this.usuario });
46.       });
47.    } else {
48.       this.showError('O serviço demorou para responder, tente mais tarde!');
49.    }
50. }, error => {
51.    this.showError(error);
52. })}
```

Figura 9 – Código da classe login.ts

3.3.2 **Interface de Análise de Veículos**

A interface de análise de veículos possibilita ao usuário tirar uma foto do veículo a ser analisado, submeter a foto para captura do OCR e do endereço relacionado a localização do dispositivo (utilizando a posição GPS). Esses dados são então processados

no serviço e retornados para a tela para revisão do usuário. É importante que o usuário revise os dados antes de submeter o veículo à consulta. A Figura 10 mostra o código da interface em detalhes.

```
1.  <ion-header>
2.     <ion-navbar>
3.        <ion-title text-center>Detetive Cidadão
4.           <p class="loggedUser" *ngIf="usuario">Usuário: {{usuario}}</p>
5.        </ion-title>
6.        <ion-buttons end *ngIf="usuario">
7.           <button ion-button icon-only  (tap)="logout()">
8.              <ion-icon name="log-out" color="light"></ion-icon>
9.           </button>
10.       </ion-buttons>
11.    </ion-navbar>
12. </ion-header>
13.
14. <ion-content>
15.    <form (ngSubmit)="enviar($event)" #registerForm="ngForm">
16.       <div class="img">
17.          <img class="img" [src]="imagem" *ngIf="imagem">
18.          <img class="img" src="assets/carsample.jpg" *ngIf="!imagem"/>
19.       </div>
20.       <div class="simple-label">Utilize o botao da câmera para tirar um foto. A placa deve ser
21.        preferencialmente nítida como a foto acima.</div>
22.          <div class="item row">
23.          <div class="col" item-width="10%">
24.             <label class="item-input" item-width="10%">
25.             </label>
26.          </div>
27.          <div class="col" item-width="25%">
28.             <label class="item-input" item-width="25%">
29.                <ion-input class="input-label2" text-center  maxlength="3" type="text"
30.                placeholder="ABC"
31.                name="confirmedPlateLetters" [(ngModel)]="confirmedPlateLetters"></ion-input>
32.             </label>
33.          </div>
34.          <div class="col" item-width="10%">
35.             <label class="item-input" item-width="10%">
36.                <ion-input class="input-label" text-center  type="text" data-ng-disabled="true" data-ng
37.                readonly="true" ng-readonly="true" value="-" placeholder="-" name="hifen" ></ion-input>
38.          </label>
39.       </div>
40.       <div class="col" item-width="25%">
41.          <div class="item-input-inset" item-width="25%">
42.             <label class="item-input">
```

```
43.                <ion-input class="input-label2"  text-center maxlength="4"  type="tel"
44.                placeholder="1234"
45.                name="confirmedPlateNumbers" [(ngModel)]="confirmedPlateNumbers"></ion-input>
46.              </label>
47.           </div>
48.        </div>
49.        <div class="col" item-width="10%">
50.           <label class="item-input" item-width="10%">
51.           </label>
52.        </div>
53.     </div>
54.     <div class="item row">
55.        <ion-input  class="input-label3" text-center type="text" placeholder="Local da Ocorrência"
56.          Name="confirmedAddress" [(ngModel)]="locationAddress"></ion-input>
57.     </div>
58.     <div class="simple-label">Antes de submeter a pesquisa, verifique os dados</div>
59.        <ion-row>
60.           <ion-col class="signup-col">
61.              <button ion-button class="submit-btn" full type="submit"
62.              [disabled]="!confirmedPlateNumbers
63.              != '' || !confirmedPlateLetters  != '' || !locationAddress != ''">Verificar veículo
64.            </button>
65.           </ion-col>
66.        </ion-row>
67.        <ion-fab right bottom>
68.           <button ion-fab color="ercabecalho" (tap)="capturar()"><ion-icon name="camera"></ion-
69.           icon></button>
70.        </ion-fab>
71.     </form>
72. </ion-content>
```

Figura 10 – Código ionic da interface analise.html

Os principais detalhes de codificação da classe de suporte da interface podem ser vistos na Figura 11. Além dos componentes padrões de importação, já explicados na seção anterior, essa classe importa os componentes do IONIC “Geolocation” (linha 8) e “Camera” (linha 6) para capturar a posição GPS do dispositivo do usuário e as imagens dos veículos (a codificação dessas funcionalidades pode ser vista nas linhas 26 e 44 respectivamente).

```
1.  import { API_BASE_URL } from './../../_constants';
2.  import { ConfirmacaoPage } from './../confirmacao/confirmacao';
3.  import { DadosProvider } from './../../providers/dados';
4.  import { Component, NgZone } from '@angular/core';
5.  import { NavController, NavParams, AlertController, LoadingController } from 'ionic-angular';
6.  import { Camera, CameraOptions } from '@ionic-native/camera';
```

```
7.  import { Toast } from '@ionic-native/toast';
8.  import { Geolocation } from '@ionic-native/geolocation';
9.  import { ModalController } from 'ionic-angular';
10. import { PhotoViewer } from '@ionic-native/photo-viewer';
11. import * as numeral from 'numeral';
12. import { LoginProvider } from '../../providers/login';
13. import * as _ from 'lodash';
14. import { Storage } from '@ionic/storage';
15. import { LoginPage } from '../login/login';
16.
17. @Component({
18.    selector: 'page-analise',
19. templateUrl: 'analise.html'
20. })
21.
22. export class AnalisePage {
23.    ionViewDidLoad() {
24.       let user: string = this.navParams.get('usuario');
25.       this.usuario = user;
26.       this.geolocation.getCurrentPosition().then((resp) => {
27.          this.lat = resp.coords.latitude;
28.          this.long = resp.coords.longitude;
29.       }).catch((error) => {
30.          alert('Ocorreu um erro tentando obter sua posição GPS!');
31.       });
32.       this.imagemAlterada = false;
33.    }
34.
35.    enviar(event: Event) {
36.       event.preventDefault();
37.       let loading = this.loadingCtrl.create({
38.       content: 'Verificando ocorrências, por favor aguarde...'});
39.       if (this.imagemAlterada == false){
40.          alert('Você precisa tirar uma foto do veículo antes de submeter a pesquisa!');
41.          return;
42.       }
43.       loading.present();
44.       this.dadosProvider.verificarAlertas(this.confirmedPlateLetters+
45.       this.confirmedPlateNumbers).subscribe(
46.       (resposta: any) => {
47.          if (resposta) {
48.             this.descricao_veiculo = resposta.modelo;
49.             this.descricao_alerta = resposta.descricao_alerta;
50.          }
51.          if (this.descricao_alerta && this.descricao_alerta == 'Furtado/Roubado')  {
52.             this.dadosProvider.inserirAlerta(this.imagem, this.confirmedPlateLetters + '-' +
53.             this.confirmedPlateNumbers, this.locationAddress, this.descricao_veiculo).subscribe(
54.             (resposta: any) => {
55.                loading.dismiss();
56.                this.navCtrl.push(ConfirmacaoPage, { placa: this.confirmedPlateLetters + '-' +
57.                this.confirmedPlateNumbers,  detalhesVeiculo: this.descricao_veiculo, ocorrencias:
58.                this.descricao_alerta, usuario: this.usuario });
59.             },
60.                err => {
61.                loading.dismiss();
62.                alert('Ocorreu um erro tentando criar o alerta!');
63.             }
64.          );
65.    } else { // all clear
66.       loading.dismiss();
67.       this.navCtrl.push(ConfirmacaoPage, { placa: this.confirmedPlateLetters + '-' +
68.       this.confirmedPlateNumbers,  detalhesVeiculo: this.descricao_veiculo, ocorrencias:
69.       this.descricao_alerta, usuario: this.usuario });
70.    }
71.    }, err => {
72.       loading.dismiss();
73.        alert('Erro obtendo dados do serviço, tente mais tarde:'+err);
74.    });}
75.    capturar() {
76.       let loading = this.loadingCtrl.create({
77.       content: 'Analisando o veículo, por favor aguarde...'
78.       });
79.       const options: CameraOptions = {
80.          quality: 80,
```

```
81.           destinationType: this.camera.DestinationType.DATA_URL,
82.           encodingType: this.camera.EncodingType.JPEG,
83.           mediaType: this.camera.MediaType.PICTURE,
84.           correctOrientation: true,
85.           targetWidth: 1024,
86.           targetHeight: 768
87.       }
88.       loading.present();
89.       this.camera.getPicture(options).then((imageData) => {
90.          this.imagem = `${imageData}`;
91.          this.imagemAlterada = true;
92.          this.dadosProvider.buscarDetalhesVeiculo_cidadao(this.imagem, this.lat,
93.          this.long).subscribe(
94.          (resposta: any) => {
95.             loading.dismiss();
96.             if (resposta.placa && resposta.placa != '' && resposta.placa != 'null') {
97.                this.confirmedPlateLetters = resposta.placa.substring(0, 3);
98.                this.confirmedPlateNumbers = resposta.placa.substring(4, 8);
99.                this.originalCarPlate = resposta.placa;
100.            } else {
101.               alert('O sistema não conseguiu ler a placa, digite a placa manualmente!');
102.            }
103.            this.locationAddress = resposta.cameraEndereco;
104.       }, err => {
105.          loading.dismiss();
106.          alert('Erro obtendo dados do serviço!');
107.       }); }, function(err) {
108.          loading.dismiss();
109.          alert('A chamada ao recurso do dispositivo foi cancelada pelo usuário!');
110.       }
111.   );
112.}
```

Figura 11 – Código ionic da interface analise.html

Por fim, na Figura 12, é possível ver o código resumido da classe LoginProvider, que como já mencionado, essa classe encapsula toda a lógica de acesso aos serviços externos do aplicativo. O acesso ao serviço REST, escrito em Java e implantado no servidor de aplicação Wildfly, é feito com a utilização do componente http do próprio Angular, como pode ser observado na linha 2. O código de acesso ao serviço e o tratamento das respostas de sucesso ou erro pode ser visto na linha 25.

```
1.  import { Injectable } from '@angular/core';
2.  import { Headers, Http } from '@angular/http';
3.  import { Observable, Observer, ReplaySubject } from 'rxjs';
4.  import { NavController, NavParams} from 'ionic-angular';
5.  import * as _ from 'lodash';
6.  import 'rxjs/add/operator/map';
7.  import { API_BASE_URL } from './../_constants';

8.  export class Usuario {
9.     public static PERFIL_CIDADAO = "PERFIL_CIDADAO";
10.    public static PERFIL_AGENTE = "PERFIL_AGENTE";
11.    constructor(
12.       public login: string,
13.       public token: string,
14.       public perfil: string) {
15.       }
16. }
17. @Injectable()
18. export class LoginProvider {
```

```
19.    public usuarioLogado: ReplaySubject<Usuario>;
20.    private TIMEOUT: number = 30*1000;
21.    constructor(public http: Http) {
22.       this.usuarioLogado = new ReplaySubject(1);
23.    }
24.    public login(usuario: string, senha: string): Observable<boolean> {
25.       if (usuario && senha) {
26.          return Observable.create((observer: Observer<boolean>) => {
27.             let headers = new Headers({
28.             'Content-Type': 'application/x-www-form-urlencoded'});
29.             let body = 'user=' + usuario+'&password='+senha;
30.             this.http
31.             .post(`${API_BASE_URL}user/authenticate`, body, { headers: headers })
32.             .timeout(this.TIMEOUT)
33.             .subscribe(
34.                data => {
35.                   this.usuarioLogado.next(new Usuario(usuario, data.toString(),
36.                   Usuario.PERFIL_AGENTE));
37.                   observer.next(true);
38.                   observer.complete();
39.                },
40.                error => {
41.                   observer.error("Identificação do usuário e/ou senha inválidos!");
42.                   observer.complete();
43.                });
44.            });
45.       } else {
46.          return Observable.throw('Identificação do usuário e/ou senha inválidos!');
47.       }
48.    }
49.    public logout() {
50.       this.usuarioLogado.next(null);
51.    }
52. }
```

Figura 12 – Código ionic da classe loginProvider.ts

3.4 ANÁLISE DE APLICATIVOS SIMILARES NO MERCADO

Essa seção visa abordar e descrever alguns aplicativos similares à proposta do aplicativo Detetive Cidadão efetuando uma comparação entre os aplicativos.

3.4.1 **Sinesp Cidadão**

O aplicativo Sinesp Cidadão é um módulo do Sistema Nacional de Informações de Segurança Pública, Prisionais e sobre Drogas, o Sinesp (Lei 12.681/2012), o qual permite acesso direto pelo cidadão aos serviços da Secretaria Nacional de Segurança Pública do Ministério da Justiça. Desenvolvido pelo SERPRO, o módulo de consulta de veículos permite ao cidadão consultar a situação de roubo ou furto de qualquer veículo no país. As informações são consultadas diretamente no banco de dados do DENATRAN, conforme parceria entre este órgão e o Ministério da Justiça [SINESPC].

O aplicativo pode ser acessado diretamente pela WEB e está disponível também para as plataformas Android, Windows Phone e IOS. O funcionamento do aplicativo é simples, basta o cidadão digitar a placa do veículo a ser pesquisado e em poucos segundos o sistema informa se o veículo possui alguma ocorrência de roubo ou furto. No caso de ocorrências, o próprio cidadão deve informar o fato às autoridades através do telefone 190.

Figura 13 – Sistema Sinesp Cidadão, desenvolvido pelo SERPRO

3.4.2 **Alerta veículos**

O sistema Alerta Veículos foi desenvolvido pela PROCERGS com o objetivo de difundir rapidamente, em todo o Estado do Rio Grande do Sul, as consultas por placa dos veículos envolvidos em ocorrências de furto e/ou roubo [ALEVEIC]. O sistema, acessível unicamente pela WEB, realiza as consultas de veículos na própria base do DETRANRS. Caso ocorram ocorrências positivas, o sistema avisa o usuário para que o mesmo denuncie o caso aos órgãos competentes, através do telefone 190. O sistema também fornece alguns detalhes da infração, onde e quando ela ocorreu, juntamente com as informações de recuperação e devolução do veículo. Uma vez encontrado alguma ocorrência de roubo ou furto, o próprio cidadão deve alertar os órgãos de segurança pública através do telefone 190.

Figura 14 – Sistema Alerta Veículos, desenvolvido pela PROCERGS

3.4.3 Comparativo Entre os Aplicativos Existentes no Mercado

O aplicativo Detetive Cidadão é de funcionamento bastante similar aos existentes no mercado, tento o mesmo propósito dos aplicativos comparados no que se refere a funcionalidade de análise de ocorrências de roubo e furto dado a placa do veículo. O diferencial do aplicativo está na integração do mesmo com o sistema de monitoramento de tráfego ALPR, desenvolvido pela PROCEMPA. Diferente dos aplicativos comparados, o Detetive Cidadão cria automaticamente um alerta no sistema de monitoramento de tráfego possibilitando uma ação mais rápida e eficiente das autoridades públicas no que se refere principalmente a ação de recuperação do veículo. A própria questão do anonimato na hora de efetuar a denúncia é um ponto a favor, visto que o próprio aplicativo alerta as autoridades, nenhuma exposição por parte do cidadão que efetua a denúncia é feita. Conforme colocado, o aplicativo pode também ser utilizado pelos agentes públicos do estado, não somente para averiguar veículos com suspeita de roubo e furto, mas também com outros tipos de ocorrências, como veículos com restrição a circulação e licenciamento vencido.

3.5 INCORPORAÇÃO AO APLICATIVO EUFACOPOA E RESULTADOS

Após o desenvolvimento da aplicativo Detetive Cidadão, o código do mesmo foi doado a PROCEMPA e incorporado como uma funcionalidade do aplicativo

EUFACOPOA, aplicativo oficial da cidade de Porto Alegre. As características do aplicativo foram mantidas e passaram a ser largamente utilizadas pela população. A popularidade do aplicativo EUFACOPOA foi afetada positivamente pela inclusão da funcionalidade do Detetive Cidadão, que passou a ter uma quantidade significativamente maior de downloads e acessos (Figura 15).

Figura 15 – Downloads do Aplicativo EUFACOPOA

A quantidade de análises de veículos suspeitos pela população, num período de 60 dias (entre 12 de Junho e 12 de Agosto de 2017) foi de aproximadamente 2.400, tendo como resultado até o momento (até 12/08) 3 veículos roubados recuperados em operações orquestradas pela SSPRS e algumas dezenas de veículos flagrados com outros tipos de ocorrências. Verificando a Figura 16, é possível concluir que a utilização do aplicativo foi bastante significativa no lançamento do mesmo, quando foi amplamente divulgado pelos meios de comunicação [BAND17] [GAUC17] [YOUT17], tendo mais de 500 verificações de placas entre os dias 14 e 16 de Junho. Após o lançamento, a utilização do aplicativo se estabilizou em uma média de 36 análises por dia.

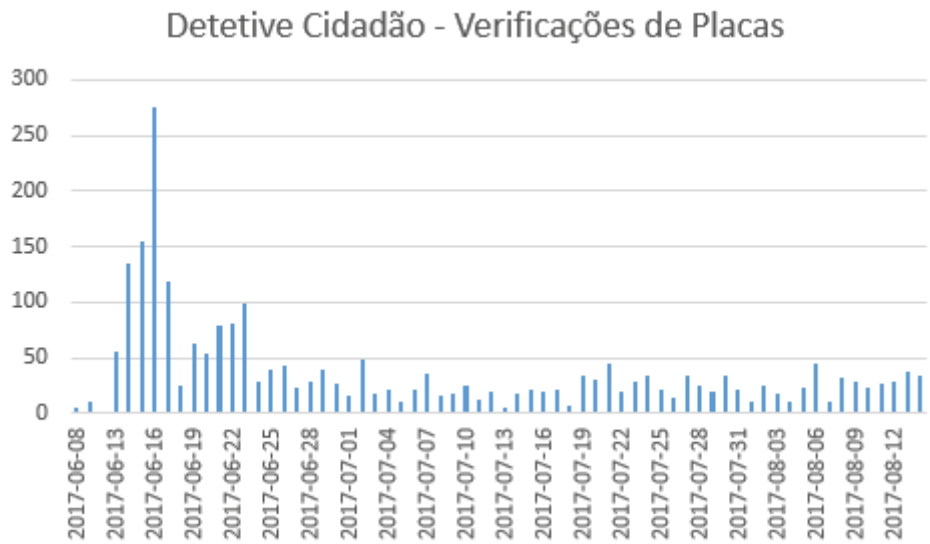

Figura 16 – Detetive Cidadão: utilização pela população

A iniciativa de piloto do cercamento eletrônico de Porto Alegre, que conta também com o Detetive Cidadão como fonte de denúncia, foi citado pelo Sindicato das Seguradoras do RS (Sindsergs) como iniciativas que, entre outras, colaboraram na queda de quase 20% nos índices de roubo e furto de veículos no mês de Junho de 2017 [CORP17].

A interface do aplicativo também foi melhorada pela PROCEMPA, que adicionou uma camada de imagem na câmera para que cidadão pudesse ter uma referência de onde encaixar a placa para adquirir uma boa imagem para análise (Figura 17). A camada adicionada sobre a câmera foi desenvolvida com a utilização do Plugin Cordova Camera Preview [CORCPR].

Figura 17 – Detetive Cidadão como parte do EUFACOPOA

4 CONSIDERAÇÕES FINAIS

Esse trabalho visou prover meios tecnológicos para empoderar o cidadão que, a partir de um aplicativo pode efetuar, de forma anônima, denúncias sobre veículos suspeitos na cidade. Os resultados do trabalho foram avaliados como positivos para área de segurança pública, já se mostrando efetivo no que se diz respeito a colaboração da população no envio de denúncias e na capacidade dos órgãos públicos de efetuar operações de recuperação dos veículos reportados usando os meios tecnológicos propostos. O objetivo do trabalho também foi demonstrar as tecnologias e frameworks utilizados no desenvolvimento do aplicativo abordando brevemente o desenvolvimento do mesmo utilizando o framework IONIC.

REFERÊNCIAS BIBLIOGRÁFICAS

[ALEVEIC] Sistema de Consulta de Veículos Alerta Veículo, acessado em 28/05/2017. Disponível em: http://www.alertaveiculos.rs.gov.br/sav/

[BAND17] Reportagem da Band divulgando o lançamento do aplicativo Detetive Cidadão. Acessado em 21/06/2017. Disponível em: http://noticias.band.uol.com.br/cidades/rs/noticia/100000863081/suspeitas-de-furto-e-roubo-de-carro-podem-ser-denunciados-por-app.html

[CORCPR] Plugin Cordova Camera Preview, acessado em 12/07/2017. Disponível em: https://github.com/cordova-plugin-camera-preview/cordova-plugin-camera-preview

[CORP17] Nóticia de redução de crimes de Roubo e Furto de veículos no site do Correio do Povo, acessado em 07/08/2017. Disponível em: http://www.correiodopovo.com.br/Noticias/Policia/2017/7/624039/RS-e-quarto-estado-com-mais-roubo-de-veiculos-no-primeiro-semestre

[DOCK16] Vitalino, J; Castro, M. Descomplicando o Docker. Brasport, 2016. 120p.

[EUPOAA] Aplicativo oficial da cidade de Porto Alegre EUFACOPOA, versão Android na loja Google Play, acessado em 12/07/2017. Disponível em: https://play.google.com/store/apps/details?id=br.gov.rs.portoalegre.app&hl=pt_BR

[EUPOAI] Aplicativo oficial da cidade de Porto Alegre EUFACOPOA, versao para IOS na loja ITunes, acessado em 12/07/2017. Disponível em: https://itunes.apple.com/br/app/eufa%C3%A7opoa/id1245871620?l=en&mt=8

[IONIC16] Phan, H. Ionic 2 Cookbook, 2nd Eddition. Packt Publishing Ltd, 2016. 320p.

[IONICFW] Página oficial de documentação do framework IONIC. Acessado em 26/07/2017. Disponivel em: https://ionicframework.com/docs/

[GAUC17] Reportagem da Gaucha divulgando o lançamento do aplicativo Detetive Cidadão. Acessado em 21/06/2017. Disponível em: http://gaucha.clicrbs.com.br/rs/noticia-aberta/aplicativo-permite-envio-de-fotos-para-consulta-de-carros-roubados-na-capital-198452.html

[G1ASPB] Reportagem do G1 sobre Roubo e Furto de veículos em Porto Alegre, de acordo com o Décimo Anuário Brasileiro de Seguranca Publica. Acessado em 11/06/2017. Disponível em: http://g1.globo.com/rs/rio-grande-do-sul/noticia/2016/11/porto-alegre-tem-maior-taxa-de-roubo-e-furto-de-veiculos-do-pais-diz-anuario.html

[GOGAPI] Documentação do serviço de localização de endereços da Google. Acessado em 19/07/2017. Disponível em: https://developers.google.com/maps/documentation/geocoding/start

[JEE713] Gupta, A. Java EE 7 Essentials: Enterprise Developer Handbook. O'Reilly Media, Inc. 2013. 362p.

[OCRD16] Chaudhuri, A.; Mandaviya, K.; Badelia, P.; Ghosh, S. Optical Character Recognition Systems for Different Languages with Soft Computing. Springer, 2016. 248p.

[OPALPR] Documentação oficial do framework OpenALPR, acessado em 19/07/2017. Disponível em http://doc.openalpr.com

[SINESPC] Sistema de Consulta de Veículos Sinesp Cidadao, acessado em 28/05/2017. Disponível em: https://www.sinesp.gov.br/sinesp-cidadao

[SSPRGS] Site da Secretaria de Segurança Pública do Estado do Rio Grande do Sul, acessado em 02/06/2017. Disponível em: https://www.ssp.rs.gov.br

[YOUT17] Vídeo de divulgacao do Detetive Cidadão feito pela Prefeitura de Porto Alegre no Youtube , acessado em 22/06/2017. Disponível em: https://www.youtube.com/watch?v=tM30V58130g