UNIVERSIDADE TECNOLÓGICA FEDERAL DO PARANÁ
DEPARTAMENTO DE PÓS-GRADUAÇÃO *LATU SENSU*
ESPECIALIZAÇÃO EM DESENVOLVIMENTO WEB

MARCELO FERNANDO SILVA DOLCE

APLICATIVO PARA CAPTURA DE DADOS MÓVEIS DE POLUENTES ATMOSFÉRICOS EM AMBIENTES URBANOS UTILIZANDO WEB APIS

MONOGRAFIA DE ESPECIALIZAÇÃO

LONDRINA
2015

MARCELO FERNANDO SILVA DOLCE

APLICATIVO PARA CAPTURA DE DADOS MÓVEIS DE POLUENTES ATMOSFÉRICOS EM AMBIENTES URBANOS UTILIZANDO WEB APIS

Trabalho de Conclusão de Curso apresentado ao Departamento de Pós-Graduação da Universidade Federal do Paraná, como requisito parcial à obtenção do grau de especialista em Desenvolvimento Web.

Orientador: Prof. Thiago Prado de Campos

LONDRINA
2015

TERMO DE APROVAÇÃO

Título da Monografia

APLICATIVO PARA CAPTURA DE DADOS MÓVEIS DE POLUENTES ATMOSFÉRICOS EM AMBIENTES URBANOS UTILIZANDO WEB APIS

por

Marcelo Fernando Silva Dolce

Esta monografia foi apresentada às 08h30 do dia **14 de novembro de 2015** como requisito parcial para a obtenção do título de ESPECIALISTA EM DESENVOLVIMENTO WEB. O candidato foi arguido pela Banca Examinadora composta pelos professores abaixo assinados. Após deliberação, a Banca Examinadora considerou o trabalho APROVADO.

_____
Prof. Me. Thiago Prado de Campos
(UTFPR)

_____
Prof. Dr. Alessandro Botelho Bovo
(UTFPR)

_____
Prof. Me. Rodrigo Freese Gonzatto
(PUC-PR)

Visto da coordenação:

_____
Prof. Me. Thiago Prado de Campos
Coordenador da esp. em Desenvolvimento Web

_____
Prof. Me. José Luis Dalto
Coordenador de Pós-Graduação Lato Senso

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

Dedicatória

Dedico este trabalho a minha querida família e amigos.

AGRADECIMENTOS

Aos meus pais, que embora não puderam estudar muito, sempre me apoiaram e incentivaram, mostrando os caminhos corretos a seguir.

A minha namorada, pelo companheirismo, palavras de conforto e motivação.

Ao Prof° Thiago Prado de Campos, pelos conselhos e sábia orientação a respeito deste trabalho, e a todos os professores do curso de especialização em desenvolvimento Web pelos conhecimentos ministrados.

Aos colegas, pela amizade, experiências compartilhadas e a todas as pessoas que direta ou indiretamente contribuíram para a conclusão do curso.

Even more powerful than the obstacles around us, however, are the obstacles within us. The most potent forces that kill off new ideas are our own limitations. (Scott Belsky, Making Ideas Happen)

DOLCE, Marcelo Fernando Silva. **Aplicativo para captura de dados móveis de poluentes atmosféricos em ambientes urbanos utilizando Web APIs**. 2015. 63 f. Monografia (Especialização em Desenvolvimento Web) - Programa de Pós-Graduação Latu Sensu. Universidade Tecnológica Federal do Paraná. Londrina, 2015.

RESUMO

Este trabalho tem como objetivo demonstrar o desenvolvimento de um aplicativo móvel multiplataforma utilizando tecnologias Web. Este aplicativo foi criado para capturar e coletar determinadas informações sobre os percursos que serão executados, fornecendo dados mais sólidos e próximos à realidade das pessoas, ajudando assim a caracterizar o estudo referente a poluição atmosférica em ambientes urbanos. O aplicativo funciona em *smartphone* e faz parte de um conjunto de instrumentos acoplados a uma bicicleta, que é utilizada em diversos percursos executados na cidade de Londrina, Paraná, por pesquisadores do curso de Engenharia Ambiental. Como outro objetivo este projeto busca dar conhecimento sobre os padrões Web das novas APIs HTML5 e também de como funciona os aplicativos híbridos. Como resultado foi possível utilizar o aplicativo em situações reais de uso para caracterizar e fornecer dados ao estudo referente a poluição atmosférica em ambientes urbanos.

Palavras-chave: Web API, HTML5, móvel, app, aplicativo híbrido, Cordova, Crosswalk, Geolocation, GetUserMedia.

DOLCE, Marcelo Fernando Silva. **Application for mobile data capture of air pollutants in urban environments using web APIs**. 2015. 63 f. Monografia (Especialização em Desenvolvimento Web) - Programa de Pós-Graduação Latu Sensu. Universidade Tecnológica Federal do Paraná. Londrina, 2015.

ABSTRACT

This project has the objective to demonstrate the development of a multiplatform mobile application using Web technologies. This application was created to capture and collect certain information about the courses that will be run, providing more solid data and near real people, thus helping to characterize the study about air pollution in urban environments. The application works on smartphone and is part of a set of instruments attached to a bicycle, which is used in several paths in Londrina, Paraná, in Environmental Engineering course researchers. As another objective of this project seeks to inform about the new Web standards HTML5 APIs and also of how hybrid applications works. As a result it was possible to use the application in real situations of use to characterize and provide data to the study related to air pollution in urban environments.

Keywords: Web API, HTML5, mobile, app, hybrid app, Cordova, Crosswalk, Geolocation, GetUserMedia.

ÍNDICE DE FIGURAS

Figura 1: Etapas de recomendação de um documento do W3C ….18

Figura 2: Prioridade de Status dos documentos da W3C ……...…..19

Figura 3: Especificação de compatibilidade da API GetUserMedia - Can I Use ……....21

Figura 4: Especificação de compatibilidade da API IndexedDB - Can I Use ..……....22

Figura 5: Especificação de compatibilidade da API Geolocation - Can I Use ..……..23

Figura 6: Especificação de compatibilidade da API File - Can I Use …….……..24

Figura 7: Características do Framework Onsen UI …….……....29

Figura 8: Tela inicial do aplicativo …….……..…35

Figura 9: Tela inicial do aplicativo com menu exposto ……...36

Figura 10: Tela de coleta e captura dos dados referentes ao percurso ……..…...…37

Figura 11: Tela de coleta. Alternando entre câmera frontal e traseira ……...……..…38

Figura 12: Tela de coleta. Selecionando o tempo para captura ……..……..…38

Figura 13: Tela de listagem dos circuitos que já foram percorridos e armazenados …39

Figura 14: Tela de detalhamento de um determinado percurso selecionado ……....40

Figura 15: Tela de detalhamento de percurso com confirmação de exclusão ……...40

Figura 16: Tela de captura de percurso posicionada verticalmente ……....48

Figura 17: Tela de captura de percurso posicionada horizontalmente ……....49

Figura 18: Galeria do smartphone listando todas as fotos armazenadas do percurso..49

Figura 19: Sequência de fotos capturadas durante o percurso ……....50

Figura 20: Arquivos .html e .json gerados e exportados automaticamente ……51

Figura 21: Representação do arquivo .html atuando como um índice ……51

Figura 22: Representação do arquivo .json exportado ……..52

Figura 23: Tela conceito do sistema que representará os dados exportados ……...54

LISTA DE TABELAS

Tabela 1: Plataformas mobile e suas respectivas tecnologias……..….15

LISTA DE ABREVIATURAS E SIGLAS

API - Application Programming Interface

APP - Application

BC - Black Carbon

CCA - Cordova Chrome Apps

CSP - Content Security Policy

CSS - Cascade Style Sheets

DOM - Document Object Model

GPS - Global Positioning System

HTML5 - Hyper-text Markup Language, versão 5

IDE - Integrated Development Environment

JSON - JavaScript Object Notation

OS - Operation System

RIA - Rich Internet Application

SDK - Software Development Kit

SQL - Structured Query Language

UI - User Interface

W3C - World Wide Web Consortium

WWW - World Wide Web

XSS - Cross Site Scripting

SUMÁRIO

1 INTRODUÇÃO....11
2. FUNDAMENTAÇÃO TEÓRICA....13
2.1 Definições dos Gases Poluentes....13
2.1.1 Black Carbon (BC)....13
2.1.2 Ozônio Troposférico....14
2.2 Tecnologias Disponíveis....15
2.2.1 Aplicações Nativas....16
2.2.2 Aplicações Híbridas....17
2.2.3 Open Web Platform....19
2.2.3.1 HTML....19
2.2.3.2 API....20
2.2.3.3 Etapas de recomendação de um documento do W3C....21
2.2.3.4 Status dos documentos do W3C....22
2.2.4 APIs utilizadas no desenvolvimento....23
2.2.4.1 GetUserMedia....24
2.2.4.2 IndexedDB....25
2.2.4.3 Geolocation....26
2.2.4.4 File API....27
2.2.5 Apache Cordova....28
2.2.6 Android Webview....28
2.2.6.1 Por quê usar CROSSWALK....29
2.3 Bibliotecas Utilizadas....31
2.4 Trabalhos Relacionados....33
3. DESENVOLVIMENTO....34
3.1 Descrição....34
3.1.1 Requisitos do Sistema....34
3.1.2 Requisitos do Dispositivo....36
3.1.3 Interface....37
3.2 Como o aplicativo foi implementado....44
3.2.1 Algoritmos....45
3.2.1.1 Coleta de coordenadas (GPS)....45
3.2.1.2 Captura de foto....46
3.2.1.3 Armazenamento de coordenadas no banco de dados....48
3.2.1.4 Armazenamento de foto em disco....49
4. TESTES E RESULTADOS....51
5. CONSIDERAÇÕES FINAIS....56
5.1 Trabalhos futuros....57
5.2 Limitações nos Testes....58
5.3 Limitações no Desenvolvimento....60

1 INTRODUÇÃO

Mudanças climáticas recentes no planeta Terra são atribuídas ao aumento das concentrações atmosféricas de gases de efeito estufa (dióxido de carbono, metano, óxido nitroso, ozônio troposférico, etc.) e de material particulado com alto conteúdo de *black carbon* – substância altamente absorvedora de radiação solar. Diferente dos outros poluentes climáticos, o ozônio troposférico e o *black carbon* são os únicos agentes que contribuem simultaneamente para o aquecimento global e para a deterioração da qualidade do ar, com efeitos nocivos à saúde humana (Bond et al., 2013).

Os poluentes atmosféricos em ambientes urbanos apresentam uma grande variabilidade espaço-temporal, devido à dinâmica do trânsito nas cidades (por exemplo, ocorrem níveis mais altos de concentração em horários de pico), e aos atributos geométricos da cidade que definem padrões de circulação local (por exemplo, cânions urbanos, cruzamentos de avenidas, entre outros). Desta forma, monitorar a concentração de poluentes atmosféricos em uma única estação fixa, como é comumente feito, leva a incertezas quando as medições são extrapoladas para outras áreas da cidade, especialmente para regiões de geografia complexa.

Uma abordagem alternativa para o monitoramento de poluentes atmosféricos é utilizando equipamentos portáteis que podem ser facilmente acoplados a bicicletas ou outras plataformas móveis, permitindo, assim, uma cobertura espacial mais ampla da área de estudo (Kingham et al., 2013).

Um estudo piloto realizado na cidade de Londrina em 2013 utilizando uma bicicleta como plataforma móvel, revelou que essa abordagem é eficiente e relativamente barata. O estudo investigou a distribuição espacial de partículas totais em suspensão em três circuitos da cidade. Um limitação na técnica de coleta de dados foi a obtenção de informações sobre as características do circuito já que o ciclista fica impossibilitado de fazer anotações sobre, por exemplo, quantidade de tráfego ou veículos desregulados que podem ter implicações diretas nas medições. Uma solução

é utilizar um telefone celular que possa registrar fotos do circuito em tempo real, com a ajuda de um aplicativo móvel instalado em um *smartphone* acoplado junto à bicicleta.

Como objetivo geral, este projeto propôs desenvolver um aplicativo que seja capaz de coletar informações referente a percursos determinados possibilitando a caracterização do estudo de poluição atmosférica, fornecendo dados do percurso, tais como data e hora do trajeto, fotos capturadas a uma quantidade de intervalo de tempo determinado e orientações de GPS (Latitude e Longitude). Tudo isto através da utilização de tecnologias Web existentes hoje no mercado.

Este trabalho visou, portanto, desenvolver e testar esta possível solução para aprimorar a coleta de dados utilizando uma plataforma móvel e tecnologias abertas da Web, HTML5, novas APIs e tecnologias adjacentes.

2. FUNDAMENTAÇÃO TEÓRICA

Este capítulo tem como objetivo explicar alguns conceitos relacionados ao problema mencionado para o qual a pesquisa de poluentes atmosféricos pretende resolver e também às tecnologias e abordagens utilizadas para criar ou chegar à solução.

2.1 Definições dos Gases Poluentes

Este projeto propôs caracterizar através de um aplicativo o estudo referente a poluição atmosférica em ambientes urbanos que determinará a distribuição espaço-temporal da concentração de poluentes climáticos de vida curta (*black carbon* e ozônio troposférico) e avaliar seus impactos sobre a saúde humana.

A seguir veremos algumas das definições dos poluentes atmosféricos inseridos no contexto desse trabalho.

2.1.1 Black Carbon (BC)

É o componente que mais fortemente absorve luz de partículas em suspensão (PM), e é formado pela combustão incompleta de combustíveis fósseis, biocombustíveis, e biomassa. BC é emitida diretamente para a atmosfera sob a forma de partículas finas (PM 2,5). BC é a forma mais eficaz de PM, em massa, em absorver a energia solar: por unidade de massa na atmosfera, BC pode absorver um milhão de vezes mais energia do que o dióxido de carbono (CO2). BC é um componente importante de "fuligem", uma mistura de absorventes de luz complexo que também contém uma parte do *black carbon*. (EPA, 2015)

2.1.2 Ozônio Troposférico

O ozônio (O3) é um gás oxidante, presente na troposfera ao nível traço, sendo formado por reações fotoquímicas na atmosfera. Na estratosfera é um dos gases-traço mais importantes, sendo formado a aproximadamente 30 km de altitude quando a radiação solar (comprimento de onda menor que 242 nm) dissocia a molécula de oxigênio (O2), mantendo-se em concentrações da ordem de 10 ppmv. O ozônio estratosférico tem a função benéfica de filtrar os raios ultravioletas do sol que chegam à superfície terrestre e que são prejudiciais à saúde humana e ao ecossistema. O mesmo ozônio presente na estratosfera também é encontrado naturalmente na troposfera em baixas concentrações (10-20 ppbv), quando em equilíbrio com seus precursores. No entanto, a influência das atividades antrópicas vem alterando consideravelmente esse equilíbrio e também os níveis globais de concentração de fundo. O ozônio, por seu caráter oxidante quando na troposfera, exerce diversos efeitos nocivos à saúde humana e ao ecossistema terrestre. Além disso, há suspeitas de que o aumento do ozônio troposférico pode ter um significativo impacto na qualidade do ar e nas mudanças climáticas, uma vez que são gases do efeito estufa (IPCC, 2001).

Os poluentes Black Carbon (BC) e Ozônio (O3), estão diretamente envolvidos no estudo de poluentes atmosféricos em ambientes urbanos do Prof° Dr. Admir Créso de Lima Targino. Pois o objetivo deste trabalho é oferecer uma plataforma que colete os dados que irão caracterizar esse estudo. A partir da coleta realizada pelo aplicativo, os dados resultantes de cada percurso serão somados aos dados gerados pelos aparelhos acoplados às bicicletas do estudo de poluentes. Esses dados possibilitarão uma maior visão dos danos causados à atmosfera por cada tipo de poluente, e através deles indicar por representação fotográfica quais os pontos que tem maior influência nesses resultados e que mais favoreceram para a aglomeração desses gases.

2.2 Tecnologias Disponíveis

Para o desenvolvimento de um aplicativo móvel que funcione em uma plataforma mobile que seja capaz de caracterizar o estudo de poluentes atmosféricos em ambientes urbanos, necessitamos de tecnologias que sirvam para captura de imagem, armazenamento de dados, escrita de arquivos, interface de uso e de comunicação com sistemas servidores que sejam compatíveis com os dispositivos móveis aos quais queremos suportar.

Os padrões abertos da Web oportunamente cobrem essas necessidades e foram escolhidos por serem um conjunto de soluções livres, pelo fato das APIs utilizadas neste projeto serem um tanto quanto recentes e por quê gostaríamos de mostrar que com a evolução da Web e os novos padrões resultantes dela e suas tecnologias adjacentes é possível um aplicativo híbrido ser capaz de atender os mesmos requisitos e solucionar os mesmos problemas que um aplicativo nativo.

Em contrapartida, o aplicativo desenvolvido neste trabalho também poderia ser feito utilizando outras tecnologias de mercado. Descrevemos algumas das tecnologias disponíveis e suas respectivas plataformas e linguagens a seguir.

2.2.1 Aplicações Nativas

Uma aplicação nativa é uma aplicação cujas funcionalidades e bibliotecas são dependentes de uma plataforma.

Em se tratando de aplicações móveis (*apps* para *smartphones*), são aplicações criadas especialmente para as plataformas Android, iOS, Windows Phone ou Blackberry, por exemplo. As aplicações nativas para estas plataformas são dependentes das funcionalidades e bibliotecas oferecidas pela plataforma e, em geral, cada plataforma costuma possuir um kit de desenvolvimento de software (SDK) e um ambiente integrado de desenvolvimento (IDE), conforme listamos na tabela abaixo.

Tabela 1: Plataformas mobile e suas respectivas tecnologias

Plataforma	Linguagem	SDK	IDE
Android	Java / Dalvik / ART	Android API	Android Studio
iOS	Objective C / Swift (iOS 8+)	Cocoa Touch	XCode
Windows Phone	C#	Windows Phone 8 SDK	Visual Studio
Blackberry OS	Java	Native SDK	Momentics IDE

Vantagens de uma aplicação nativa:

- Velocidade: Aplicativos nativos são mais rápidos por terem acesso direto ao sistema operacional e serem programados específicos na linguagem nativa do dispositivo.
- Funcionamento offline: Um aplicativo nativo é melhor se for necessário funcionar sem internet. Cache no navegador é possível no HTML5, mas ainda é limitado comparado ao nativo.

Entretanto, uma aplicação nativa necessita de:

- Curva de aprendizado maior, para ter domínio da linguagem e sua IDE;
- Respeitar as diretrizes de interface de usuário de cada plataforma, pois há uma experiência de uso distinta entre elas;
- Tempo para desenvolver o código diferente para cada plataforma, caso seja interesse atingir um maior número de usuários;
- Depender da loja de aplicativos da plataforma (como por exemplo, Apple Store ou Play Store) e do usuário para receber atualizações do aplicativo.
- Tempo para dar manutenção e suporte à diferentes plataformas: as mudanças devem ser enviadas individualmente a cada loja. Enquanto isso, web apps e híbridos podem ser atualizados com a frequência necessária, como uma página na internet.

2.2.2 Aplicações Híbridas

O conceito clássico de híbrido refere-se no ato de misturar. Na mistura de dois ou mais elementos diferentes o resultado é a oposição da ordem natural das coisas. Os aplicativos móveis híbridos surgiram a partir desse conceito, entre a junção do nativo e da Web.

Os híbridos são aplicações projetadas para serem executadas em *browsers* de dispositivos móveis que se comportam como um aplicativo nativo, mas na verdade são páginas web, tendo parte ou total de seu conteúdo carregado pela internet. Sua interface gráfica é adaptada para dispositivos com telas menores, utilizando conceitos como o *responsive* (são páginas com visualização e experiência adaptável à plataformas e resoluções diferentes).

Esses aplicativos, são instalados a partir da loja de aplicativos, criam um ícone na tela principal e têm acesso completo às capacidades do dispositivo como qualquer outro aplicativo.

São populares devido a sua característica multiplataforma e redução na curva de aprendizagem, isto é, se desenvolve apenas uma vez e executa em diferentes plataformas, permitindo redução dos custos de produção.

Aplicativos híbridos vão funcionar da mesma forma que os aplicativos nativos, contudo:

- São baseados em HTML5(estruturação das páginas), CSS3(estilização da interface) e JavaScript(programação), sendo essas as principais tecnologias;
- Exige um menor custo no desenvolvimento comparado com os nativos, isso pelo fato de manter apenas um código fonte não precisando desenvolver um código para cada plataforma;
- Para efetuar manutenções e aplicar atualizações periódicas nos aplicativos, há uma enorme vantagem sobre as nativas, realizando a manutenção no código apenas uma única vez para todas as plataformas.

Hoje em dia uma grande quantidade de soluções para plataformas móveis gira em torno de aplicativos híbridos, pois escrever uma vez e executar em qualquer plataforma torna este modelo cada vez mais utilizado no desenvolvimento.

2.2.3 Open Web Platform

Open Web Platform, ou, Plataforma Aberta da Web é a coleção de tecnologias abertas (royalty-free), que incrementam a Web. Utilizando a Open Web Platform, todos tem o direito de implementar um componente de software da Web sem a necessidade de quaisquer aprovações ou despesas com taxas de licença.

2.2.3.1 HTML

HTML é uma linguagem de marcação para estruturação e apresentação de conteúdo para a World Wide Web. A quinta versão do HTML, traz consigo importantes mudanças quanto ao seu papel no mundo da Web, através de novas funcionalidades como semântica e acessibilidade. Nesta versão é ampliada de forma surpreendente as funcionalidades da HTML, alterando de maneira significativa como se desenvolve para a web. Trata-se da mais extensa especificação para a HTML focada em criar funcionalidades para desenvolvimento não só de sites, mas também de aplicações de internet rica (RIA).

A utilização de marcação HTML5 possibilita ao desenvolvedor web alcançar os seguintes objetivos:

- Criar um código totalmente semântico empregando os novos elementos da linguagem ou mesmo os elementos já existentes que tiveram sua semântica redefinida.
- Usar os novos atributos da linguagem para criar elementos gráficos ricos no desenvolvimento de aplicações Web.
- Inserir mídia, tais como áudio e vídeo e a funcionalidade canvas de uma forma nativa, sem dependência de *plugins* de terceiros ou extensões proprietárias.
- Desenvolver formulários altamente interativos com validação no lado do cliente com uso de atributos criados especialmente para essas finalidades.

- Usar as API Geolocation, Web Storage, Web Messaging, Web Sockets e Web Workers.
- Usar mecanismos, como Microdados e, ARIA para incrementar a acessibilidade.(HTML5 a linguagem de marcação que revolucionou a Web, 2011)

HTML5 também é um candidato em potencial para aplicações multiplataforma móveis. Muitos recursos do HTML5 tem sido construídos com a consideração de ser capaz de executar em dispositivos de baixa potência como *smartphones* e *tablets*.

HTML5 não se trata apenas de uma linguagem de marcação, mas também de um conjunto de novas funcionalidades encapsuladas em APIs que podem ser acessadas via JavaScript.

2.2.3.2 API

Application Programming Interface, ou Interface de Programação de Aplicativos é um conjunto de rotinas e padrões de programação para acesso a um aplicativo de software ou plataforma e acessíveis por meio de uma linguagem de programação, para interação entre sistemas diferentes, sem a necessidade de conhecer a engenharia da implementação de cada sistema.

Na prática, uma API é a exposição de uma série de ferramentas, métodos de programação e protocolos, com o objetivo de facilitar a programação de uma aplicação.

Uma API documenta os métodos e propriedades disponíveis sobre um objeto ou conjunto de objetos de um sistema e como acessá-los para obter os resultados desejados.

No W3C quem vem trabalhando nisso é o *Device APIs Working Group*, que tem como principal objetivo criar as chamadas *cliente-side* APIs que permitem o

desenvolvimento de aplicações e Web Widgets que interagem com serviços de dispositivos, tais como geolocalização, acelerômetro, câmera, vibração, contatos, etc.

Todos os recursos necessários para o desenvolvimento do aplicativo deste trabalho estão atualmente consistentes ou em processo de padronização pelo W3C.

2.2.3.3 Etapas de recomendação de um documento do W3C

Estão descritos a seguir, para um maior entendimento, os processos e status de documentos elegidos pelo W3C até se transformarem em uma recomendação.

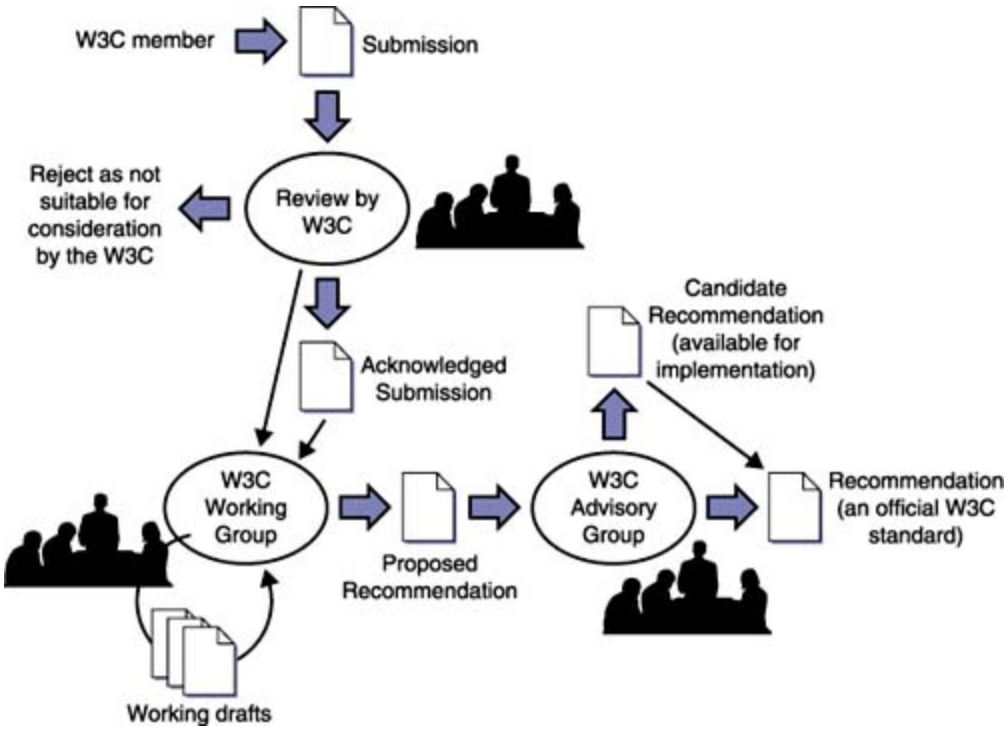

Figura 1: Etapas de recomendação de um documento do W3C

2.2.3.4 Status dos documentos do W3C

- **Rascunho** (*Working Draft*) - documento ainda em fase de escrita e discussão por um grupo de trabalho e pela comunidade.
    - **Última Chamada do Rascunho** (*Last Call Working Draft*) - quando os membros do grupo de trabalho acreditam já terem discutido o suficiente sobre o documento, dá-se um prazo para últimas contribuições.
- **Candidata a Recomendação** (*Candidate Recommendation*) - documento já revisado após últimas contribuições e que satisfaz os requisitos técnicos do grupo de trabalho. Nesta versão a tecnologia é colocada a disposição para experiências de implementações. Geralmente inclui uma seção com riscos que indicam partes da recomendação poderão não estar na versão final.
- **Proposta de Recomendação** (*Proprosed Recommendation*) - documento maduro, escrito após ampla revisão e experiências de implementação. Encaminhado ao Comitê Consultivo do W3C para aprovação.
- **Recomendação do W3C** (*W3C Recommendation*) - documento aprovado pelo Comitê Consultivo do W3C.

Figura 2: Prioridade de Status dos documentos da W3C

2.2.4 APIs utilizadas no desenvolvimento

As APIs do HTML5 que foram utilizadas no desenvolvimento deste aplicativo móvel estão descritas a seguir juntamente com sua tabela de especificações de compatibilidade indicando em qual versão e plataforma serão necessários para seu pleno funcionamento.

2.2.4.1 GetUserMedia[1]

A API getUserMedia acrescenta fontes dinâmicas de captura, como o uso de microfones e câmeras. As características dessas fontes pode mudar em resposta às necessidades da aplicação. As especificações dessa API encontram-se Em Rascunho.

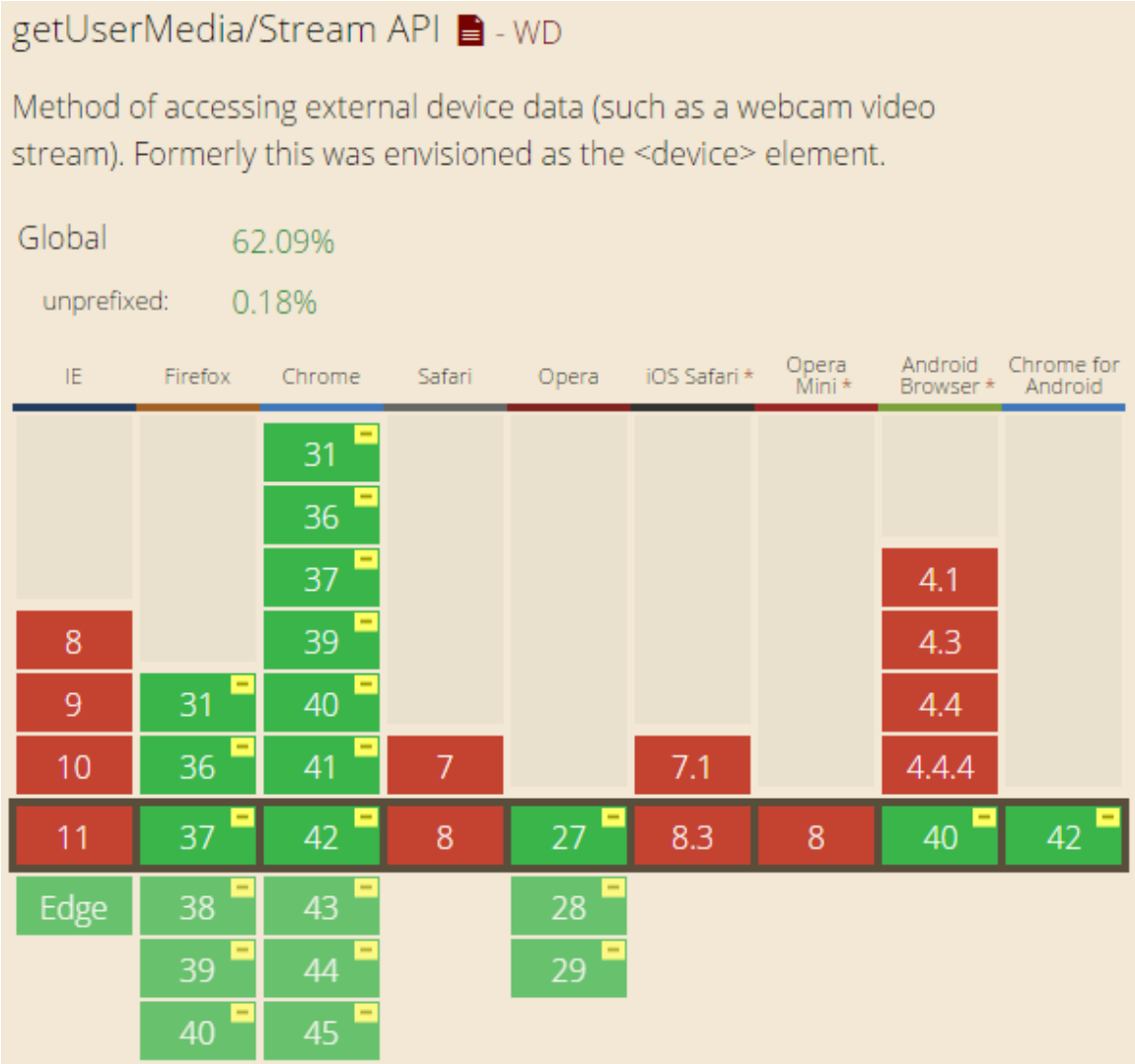

Figura 3: Especificação de compatibilidade da API GetUserMedia - Can I Use

[1] http://www.w3.org/TR/mediacapture-streams/#dom-mediadevices-getusermedia

2.2.4.2 IndexedDB[2]

IndexedDB é uma API para armazenamento de dados no navegador do usuário, de quantidades significantes de informações e buscas com alta performance por índices. Com ele você pode criar aplicações web com possibilidade de fazer *query* sem necessidade de conexão, suas aplicações podem funcionar tanto *online* quanto *offline*.

IndexedDB é um banco de dados de registros que prendem valores simples e objetos hierárquicos. Cada registro consiste em uma chave e algum valor. Além disso, o banco de dados mantém índices sobre registros de que ele armazena. As especificações dessa API encontram-se Em Recomendação.

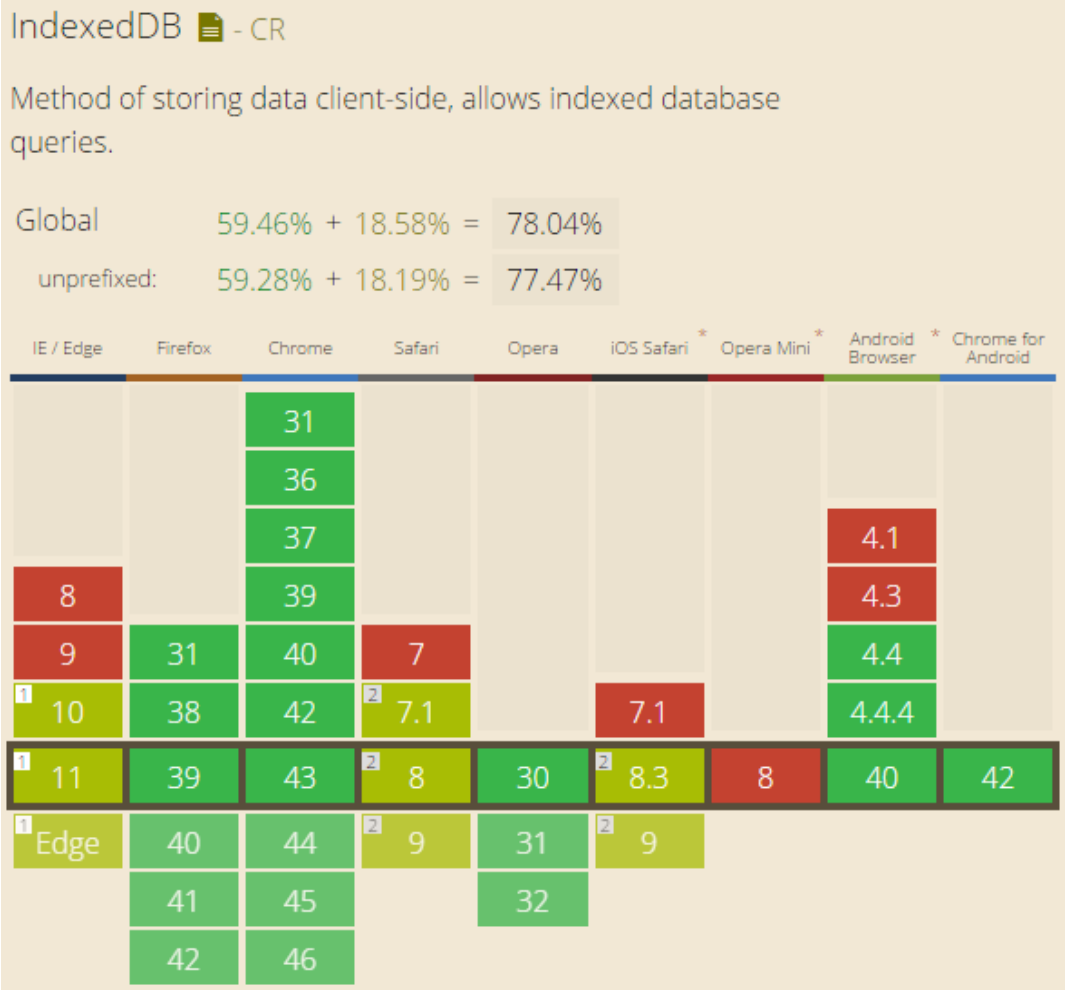

Figura 4: Especificação de compatibilidade da API IndexedDB - Can I Use

[2] http://www.w3.org/TR/IndexedDB/

2.2.4.3 Geolocation[3]

A API Geolocation ou Geolocalização define uma interface de alto nível para a informação de localização associada apenas com o dispositivo que hospeda a implementação, tais como latitude e longitude. Essas informações podem ser então coletadas e armazenadas como complemento das informações de localização do trajeto percorrido. As especificações dessa API encontram-se Em Rascunho.

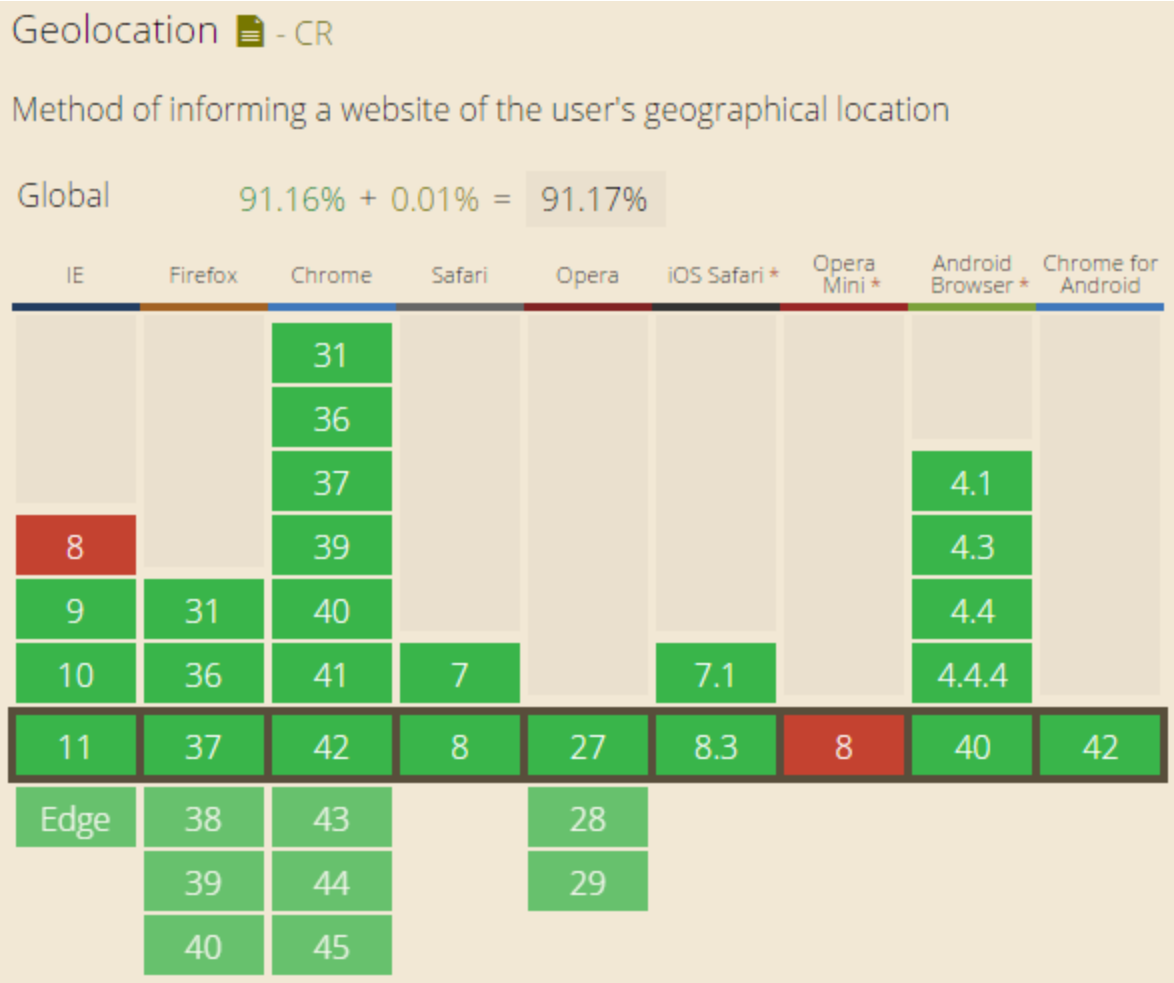

Figura 5: Especificação de compatibilidade da API Geolocation - Can I Use

[3] http://dev.w3.org/geo/api/spec-source.html

2.2.4.4 File API[4]

Aplicações Web são atualmente bastante limitadas em como elas podem gravar arquivos. É possível fornecer um *link* para que seja realizado o *download*, mas a criação e gravação de arquivos de tipo arbitrário, ou modificar arquivos baixados em seu caminho para o disco, é difícil ou impossível. Esta especificação define uma API através do qual usuário podem permitir que aplicativos possam gravar arquivos ou gerá-los. (W3C, 2015).

A file-system-api e file-writer-api no momento são recursos apenas do Chrome e não um padrão W3C. Isto porque, a Google não convenceu MS, Mozilla e Apple de que isso era necessário.

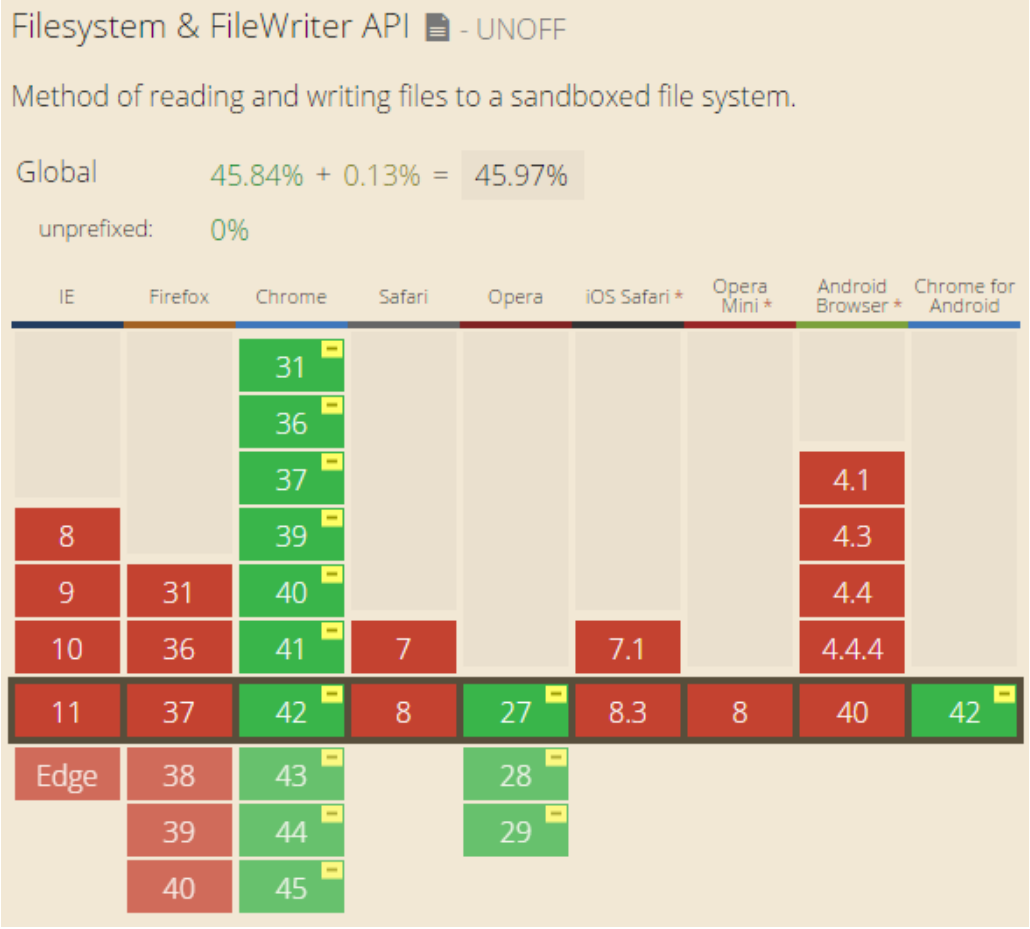

Figura 6: Especificação de compatibilidade da API File - Can I Use

[4] http://w3c.github.io/filesystem-api/

2.2.5 Apache Cordova

Apache Cordova é um conjunto de APIs de dispositivos que permitem com que um desenvolvedor de aplicativo móvel possa acessar as funções nativas do dispositivo, como a câmera, microfone, GPS, notificações, acelerômetro e etc. Combinado com um *framework* de interface do usuário, tais como jQuery Mobile, Dojo Mobile ou Sencha Touch, entre outros. Isto permite que um aplicativo de smartphone possa ser desenvolvido apenas com HTML, CSS e JavaScript. (APACHE, 2015, tradução nossa)

Ao usar as APIs do Cordova, um aplicativo pode ser construído sem qualquer código nativo (Java, Objective-C, etc) a partir do desenvolvedor do aplicativo. Em vez disso, são utilizadas tecnologias Web, e eles estão hospedados na própria aplicação localmente (em geral, não em um servidor Web remoto). (APACHE, 2015, tradução nossa)

2.2.6 Android Webview

Android WebView é um componente do sistema com tecnologia Chrome que permite aos aplicativos Android apresentarem conteúdo da Web.

A partir da versão 4.4 do Android (KitKat) o componente WebView inclui uma versão atualizada do motor V8 JavaScript e suporte para padrões modernos da web que estavam faltando na antiga WebView. Nesta versão ele também compartilha o mesmo motor de renderização do Chrome para Android versão 30, tornando assim uma fidelidade na apresentação de conteúdo muito maior e consistente entre o WebView e Chrome. (ANDROID, 2015, tradução nossa).

2.2.6.1 Por quê usar CROSSWALK

No Android existe uma desvantagem no uso de aplicativos híbridos. isto porque, no componente Webview, em versões antigas do Android, as aplicações podem ficar um pouco limitadas, pela fragmentação do sistema operacional em si. Cada versão do Android possui um Webview diferente acoplado.

O Chrome ganhou várias melhorias no Android, no entanto, executar antigos Webviews significa o não aproveitamento desses aprimoramentos e melhorias.

Com as variações de plataforma do mercado você é forçado a assumir que cada WebView Android funciona de forma diferente. Há diferenças em:

- Quais APIs JavaScript estão disponíveis;
- Suporte e sintaxe para propriedades CSS;
- Como a interface do seu aplicativo é processado;

É aí que entra o projeto Crosswalk de código aberto da Intel. O projeto fornece um componente WebView Chromium duplo embutida para versões Android como a versão Ice Cream Sandwich (4.0). Usando o Crosswalk, os aplicativos híbridos podem executar a versão mais recente do Chromium independente das versões do Android. Já para o sistema iOS, o WebView teve grandes atualizações desde a versão 6, e a partir da versão 8 do iOS, o Webview corresponderá às capacidades completas do navegador Safari. Isso tudo significa que é possível criar aplicativos híbridos para usar uma moderna Webview e executá-los em quase todos os dispositivos iOS e Android lançados dentro dos últimos três anos.

Com Crosswalk da Intel, o Webview Android é atualizado juntamente com a última versão do Google Chrome. Assim, incluindo o *runtime* Web Crosswalk ao seu Projeto de aplicativo híbrido, Cordova ou PhoneGap, os usuários sempre verão ele através de um WebView previsível, baseado no Google Chromium.

Vantagens de um projeto com Crosswalk:

- Obter um comportamento consistente, previsível, reduzindo a fragmentação dos dispositivo Android.

- Utiliza as últimas inovações da web e APIs. Fornece uma rica experiência de recurso em todos os dispositivos com sistema Android 4.0+.
- Fornece fácil depuração através do Chrome DevTools.
- Melhora o desempenho do seu HTML, CSS e JavaScript.

Atualmente, os aplicativos híbridos podem usar tecnologias como WebRTC, WebGL, WebAudio, Canvas 2D acelerado e kits de ferramentas web, como o Polymer, com desempenho máximo, sem a necessidade de *polyfills* de navegador.

Os aplicativos híbridos não são limitados apenas ao uso de APIs web. Como esses apps são também nativos, tem-se acesso completo a todas as APIs de dispositivos iOS e Android, mensagem por nuvem, notificações, pagamentos, Bluetooth, *sockets*, câmeras e muito mais.

Com a soma dessas tecnologias, Cordova, Android Webview e Crosswalk é possível encapsular e compilar toda uma aplicação que utilize apenas de tecnologias abertas da web (HTML, CSS e Javascript) com suporte as mais modernas APIs e que funcione em cima de diversas plataformas móveis, tais como Android e iOS.

O aplicativo depois de desenvolvido, utilizando dos recursos das APIs do Cordova ou Web Apis, é compilado pelo conjunto de ferramentas do Cordova, sendo assim capaz de ser executado sob o componente Webview do Android (respeitando suas especificações de compatibilidade).

2.3 Bibliotecas Utilizadas

Onsen UI

É um *framework* de interface de usuário baseado em elementos personalizados do HTML5 especialmente projetado para funcionar sobre aplicativos PhoneGap & Cordova. É composto por uma grande seleção de componentes de interface do usuário com base na Web, fornece total suporte ao *layout* responsivo, incluindo *smartphones* e *tablets*. (ASIAL, 2015, tradução nossa)

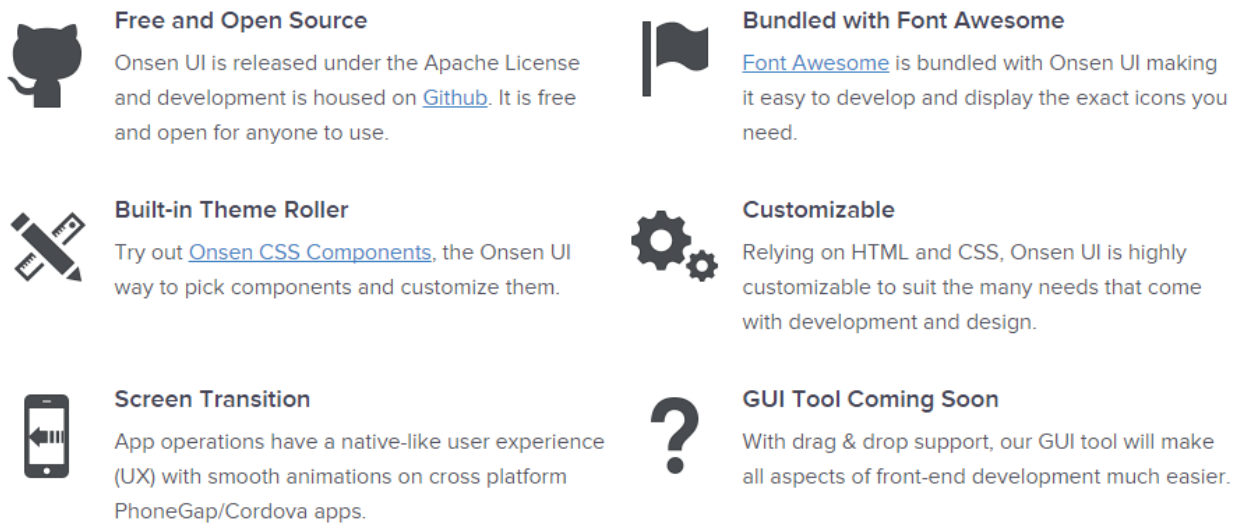

Figura 7: Características do Framework Onsen UI

AngularJS

É um *framework* para aplicações Web dinâmicas. Ele permite que você use HTML como linguagem de declaração de documentos estáticos e permite estender sua sintaxe. A ligação de dados entre o Angular e o HTML é através de injeção de dependência reduzindo a quantidade de código que seria escrito tradicionalmente.

É totalmente extensível e funciona bem com outras bibliotecas, como jQuery, Sencha, KendoUI, etc. Cada recurso pode ser modificado ou substituído de acordo com suas necessidades de fluxo de trabalho de desenvolvimento e de características exclusivas. (GOOGLE, 2015).

2.4 Trabalhos Relacionados

Uma das soluções de coleta de informações de campos atualmente disponível no mercado é o aplicativo Collector for ArcGIS da *ArcGIS*.

Collector for ArcGIS é um aplicativo móvel que tem como objetivo coletar e atualizar as informações no campo, registrar sua posição atual e inserir os dados que foram capturados. Com ele é possível estender o alcance da plataforma *ArcGIS* para o campo e melhorar a precisão e numerário dos seus dados espaciais. Entre os recursos disponíveis, é possível: (ESRI, 2015).

- Criar e compartilhar mapas para coletar dados;
- Criar dados a partir de relatórios;
- Incluir imagens e vídeos aos dados;
- Baixar mapas no seu dispositivo e trabalhe offline;
- Utilizar GPS para criar e atualizar dados do mapa;
- Coletar feições de área, linhas e pontos;
- Preencher formulários de mapas dinâmicos de fácil uso;
- Localizar lugares e obter direções;
- Rastrear e informar áreas que você visitou.

Para utilizar seus mapas no Collector, é necessário uma conta organizacional do ArcGIS. (ESRI, 2015).

O Collector tem como objetivo atualizar informações e organizá-las a fim de caracterizar dados referentes a projetos e pesquisas em campo. Entretanto este aplicativo foca principalmente no gerenciamento de conteúdo posterior a coleta e não a captura de dados em si. Mas sua utilização ainda sim é válida, pois agregaria nos resultados finais da pesquisa de poluentes atmosféricos, visando a atualização e organização de conteúdo. Assim, processando dados pós coletados pelo aplicativo proposto neste trabalho.

3. DESENVOLVIMENTO

3.1 Descrição

A aplicação tem como objetivo principal funcionar como uma ferramenta de captura de imagem e coleta de dados ao usuário que percorrerá um determinado circuito, utilizando APIs do HTML5. Para atingir esse objetivo foram levantados os requisitos do sistema e realizada uma pesquisa de ferramentas, tecnologias e técnicas, além da criação de protótipos das telas necessárias para o funcionamento da aplicação.

3.1.1 Requisitos do Sistema

Os requisitos definidos para o desenvolvimento deste aplicativo foram pensados e elaborados em conjunto com o responsável pelo grupo de pesquisa, Prof° Dr. Admir Créso de Lima Targino, através de uma reunião e explicou quais eram as limitações do processo de coleta de dados de sua pesquisa. O intuito é suprir o básico das funcionalidades necessárias para o aplicativo ser capaz de caracterizar visualmente um percurso, já que o ciclista fica impossibilitado de fazer anotações.

O sistema deve:

1. Funcionar em aparelhos móveis como smartphones, que serão utilizados acoplados às bicicletas, responsáveis pela condução nos circuitos.
2. Apresentar na tela um mapa com o percurso atualmente percorrido e atualizado em tempo real.
3. Operar em modo on-line e off-line.
4. Ser capaz de capturar uma foto a cada período de tempo determinado (em segundos), sendo possível a alteração desse tempo pelo usuário quando necessário.
5. Salvar automaticamente em memória as fotos capturadas a fim de caracterizar todo o circuito percorrido.
6. Ser capaz de sincronizar e vincular à foto capturada do percurso com a exata coordenada geográfica em que foi capturada.
7. Armazenar informações de todo o percurso e exportá-las quando for necessário.
8. Conter alguns controle de percurso como iniciar/parar, tempo de captura de foto, escolha de câmera fronta ou traseira.
9. Utilizar tecnologias abertas, explicitamente APIs do HTML5.

3.1.2 Requisitos do Dispositivo

- Sistema Android
    - O aplicativo foi compilado especificamente para este sistema operacional.

- Android 4.0 (Ice Cream Sandwich) ou posterior
    - É nesta versão do Android ou superior que é suportado a versão mínima do Android Webview disponibilizado pelo projeto Crosswalk.

- Processador **ARMv7** ou x86
    - Esta é a arquitetura do processador do dispositivo suportada pelo aplicativo quando compilado.

- Suporte para localização precisa (baseado em GPS e rede)
    - Através da tecnologia GPS que as coordenadas de localização são capturadas.

3.1.3 Interface

O protótipo de *layout* do aplicativo foi desenvolvido antes mesmo da programação das funcionalidades. As telas foram pensadas levando em consideração todos os recursos que o aplicativo deveria oferecer. Por este motivo foi utilizado o Onsen UI, um framework de HTML5 para aplicações móveis, este oferece diversos componentes que facilitam a customização e caracterização do *layout* resultando em maior produtividade.

O aplicativo não utilizou um padrão específico de Interface. O único padrão seguido foi das diretrizes de composições de interface do framework Onsen UI.

Após o desenvolvimento do protótipo, o aplicativo foi apresentado ao Prof° Dr. Admir Créso de Lima Targino, e consequentemente aprovado. E assim foi dado sequência no desenvolvimento.

O aplicativo foi projetado para ser composto por quatro telas principais, sendo três delas acessíveis através do menu lateral. Caracterizam-se Tela inicial com a **apresentação** do aplicativo, **coleta** e captura dos dados por GPS e das fotos e listagem dos **percursos** armazenados. E a quarta tela, visível somente quanto um determinado percurso é escolhido. Esta é responsável por detalhar mais as informações do circuito armazenado.

A primeira tela é a apresentação. Seu principal objetivo é exibir a descrição do aplicativo e ser a ponte entre as outras funcionalidades existentes do sistema. O menu fica alojado verticalmente a esquerda sempre disponível ao usuário com um movimento simples na tela de, deslizar para a direita.

Figura 8: Tela Inicial do Aplicativo

Figura 9: Tela Inicial do Aplicativo com menu exposto

A segunda tela, nomeada de “Coletar” fica responsável por toda a coleta e captura dos dados que irão caracterizar o circuito. Esta tela é composta por controles de iniciar/parar ao topo, controle de captura manual sobre a imagem transmitida da câmera, visualização do mapa em tempo real no rodapé com seus respectivos controles de escolha de câmera e tempo de captura sob o mapa.

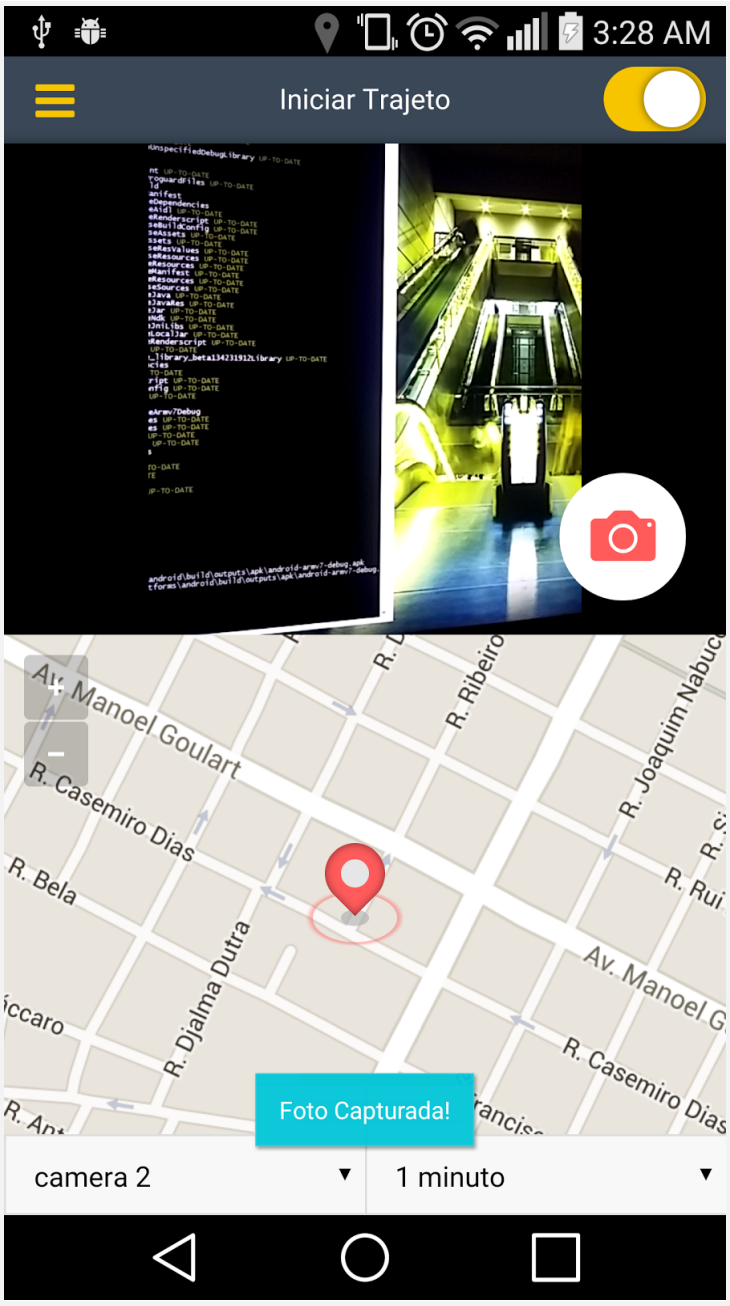

Figura 10: Tela de coleta e captura dos dados referentes ao percurso

Na tela de Coleta de dados, o usuário tem acessível dois controles para configurar, o seletor de Câmera, permitindo alternar entre a câmera frontal ou traseira, e o seletor de tempo, que permite escolher entre 3, 5, 10 segundos e 1 minuto.

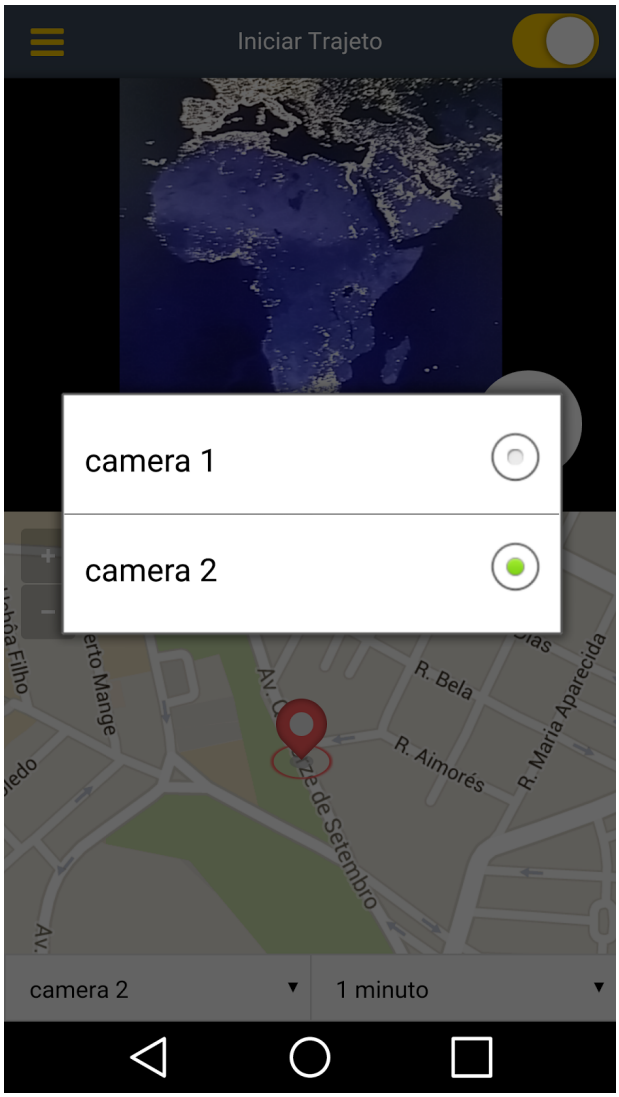

Figura 11: Tela de coleta. Alternando entre câmera frontal e traseira

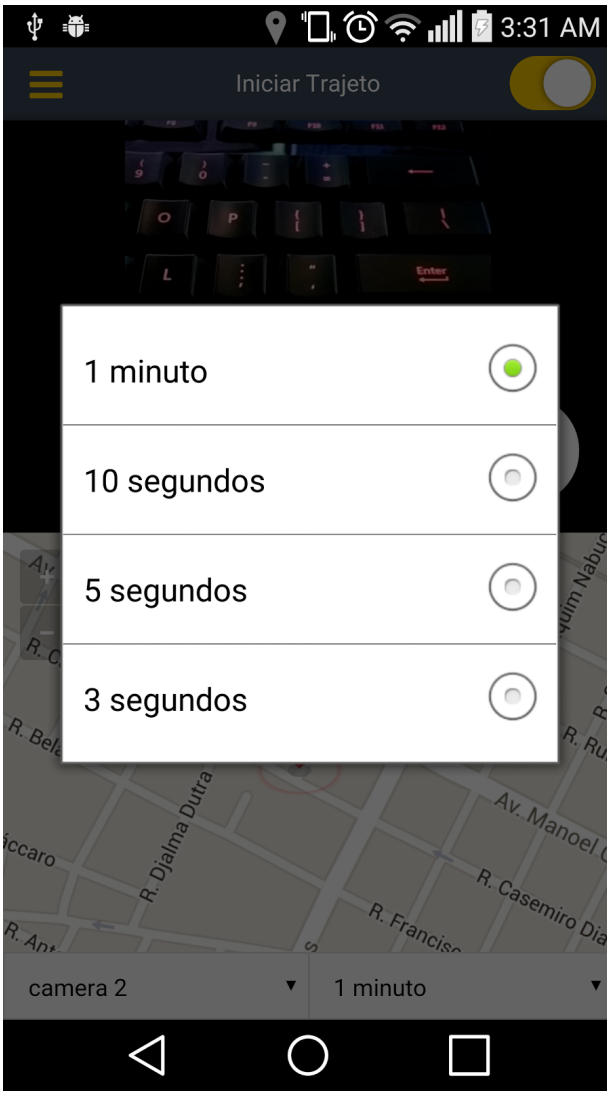

Figura 12: Tela de coleta. Selecionando o tempo para captura

A terceira tela, nomeada de “Percursos” é responsável pela listagem de todos os últimos percursos realizados e armazenados. A lista é ordenada de forma crescente, inserindo o último percurso ao final da lista. Essa listagem consiste em pré exibir ao usuário informações de data, duração e quantidade de coordenadas salvas, facilitando a busca de um determinado item.

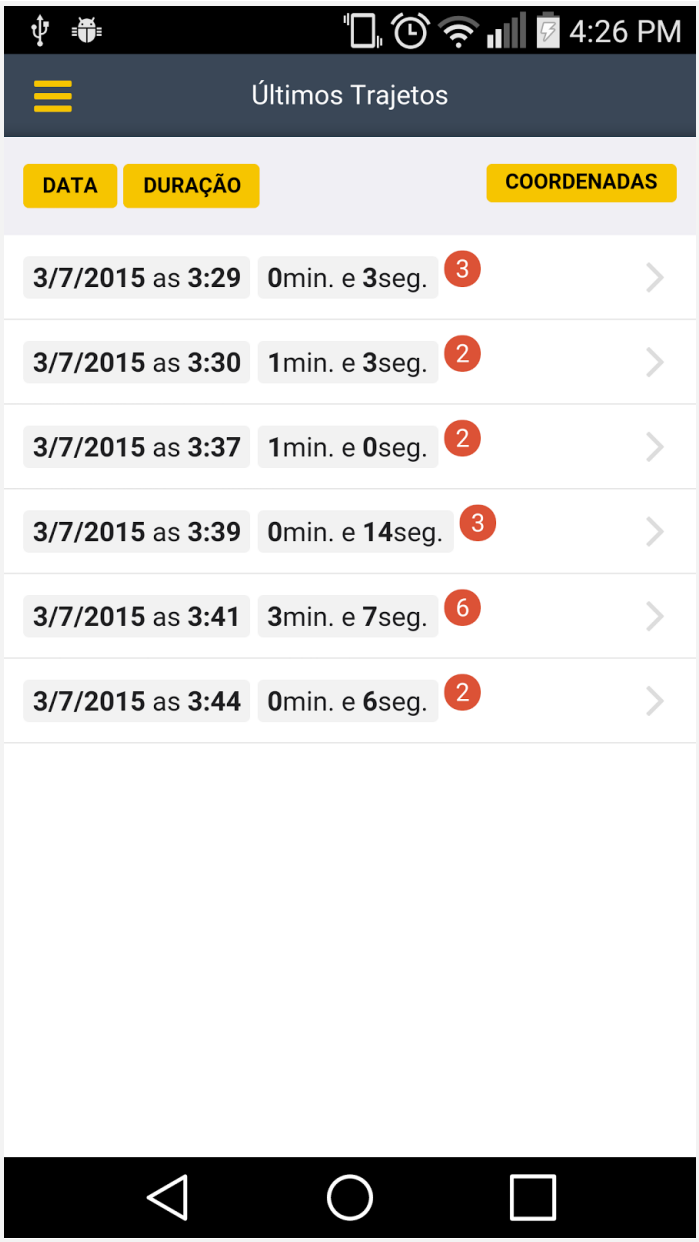

Figura 13: Tela de listagem dos circuitos que já foram percorridos e armazenados

A última tela é encarregada de exibir todas as informações de um determinado percurso. Aqui estão contidos dados como a data em que foi registrado, período em que foi iniciado, duração do circuito e quantidade de coordenadas registradas em todo o trajeto. Além de disponibilizar dois controles ao usuário: os comandos “Exportar” e “Remover”.

- O botão Exportar, tem a função de salvar as informações desta tela incluindo todas as coordenadas registradas em formato JSON, possibilitando a replicação dos pontos registrados a partir do mapa na tela de captura em outras plataformas caso haja necessidade.
- O botão Remover, tem a função de remover o circuito selecionado, incluindo todas as suas coordenadas. ao ser pressionado o usuário se depara com uma confirmação da exclusão, evitando a exclusão não desejada.

Figura 14: Tela de detalhamento de um determinado percurso selecionado.

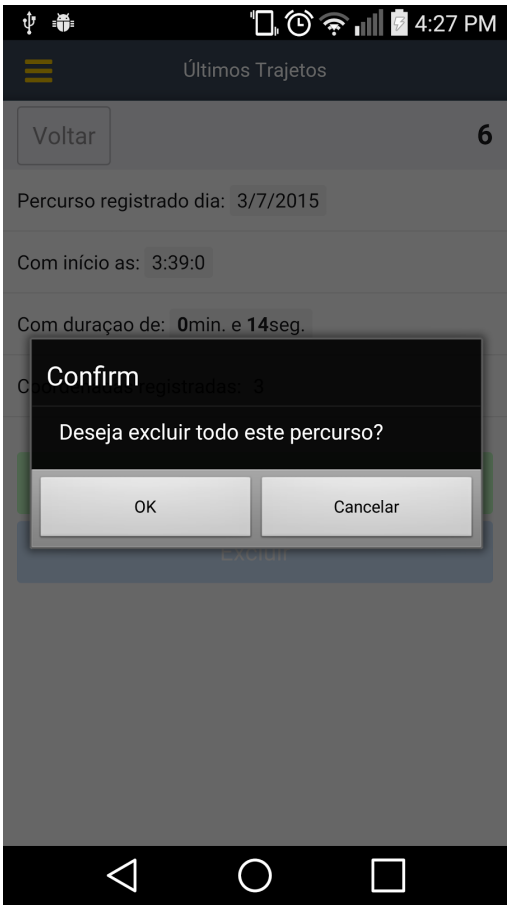

Figura 15: Tela de detalhamento de percurso com confirmação de exclusão.

3.2 Como o aplicativo foi implementado

A aplicação se baseia na utilização das APIs do HTML5 que fazem a comunicação direta com o *hardware* do dispositivo. Essas APIs são responsáveis por todas as funcionalidades do aplicativo, como captura de imagem, coleta dados, coordenadas e armazenagem de informações no dispositivo. A coleta de coordenadas e captura de fotos poderá ser realizada tanto on-line como off-line. Esses dados ficarão armazenados no dispositivo em um banco de dados através da API IndexedDB e em armazenamento em disco, através da API File e posteriormente essas informações poderão ser exportados para um arquivo no padrão JSON.

Para atingir tais objetivos, foi utilizado como linguagem de programação de acesso as APIs, o Javascript, a linguagem de marcação HTML e de estilização da interface do aplicativo o CSS.

O *framework* base de componentes e interface de usuário utilizado foi o Onsen UI, que funciona juntamente com a biblioteca AngularJS.

A biblioteca Angular JS funciona em conjunto com o Onsen UI e fica responsável por gerenciar as requisições do *framework*, controles de componentes de interface, alterações e organizações do conteúdo e gerenciamento de telas de forma dinâmica, tornando-as independentes umas das outras.

No próximo tópico está descrito o funcionamento dos algorítimos de acesso às APIs.

3.2.1 Algoritmos

3.2.1.1 Coleta de coordenadas (GPS)

A Api Geolocation possui dois métodos principais para coleta de coordenadas de localização, *getCurrentPosition()* e *watchPosition()*.

Através do método *getCurrentPosition* no objeto *navigator.geolocation* obtemos as coordenadas através de um *callback* assíncrono, já que a obtenção dos dados pode demorar um pouco. Utilizando o método *watchPosition*, será registrado uma função do manipulador sendo possível monitorar o deslocamento do usuário informando a posição do dispositivo toda vez que a mesma for alterada, útil para uma aplicação de navegação como o desenvolvido nesse trabalho.

O método *watchPosition* retorna um valor *watchID*, que em seguida, poderá ser usado para cancelar o registro do manipulador quando passado para o método *Geolocation.clearWatch().*

Especificar uma função de manipulação de erro de retorno de chamada é opcional.

```
function onError(error) {
    console.log('code: ' + error.code + '\n' + 'message: ' + error.message + '\n');
}

// Método watchPosition é configurado os parâmetros da API
watchID = navigator.geolocation.watchPosition(onSuccessGeolocation, onError, {
    frequency: 3000,
    maximumAge: 5000,
    enableHighAccuracy: true
});

// Callback de Sucesso, é retornado o objeto position com as coordenadas
function onSuccessGeolocation(position) {
    latitude = position.coords.latitude;
    longitude = position.coords.longitude;
}
```

3.2.1.2 Captura de foto

Para a captura de foto utilizamos o objeto *navigator.getUserMedia*, que requisita a permissão do usuário para utilizar um dispositivo multimídia, como uma câmera ou um microfone.

Se o usuário conceder permissão, o *successCallback* é chamado na aplicação que originou o pedido, recebendo um objeto LocalMediaStream como argumento.

```
// Verificando qual prefixo da API getUserMedia é reconhecido (compatível) pelo
navegador
navigator.getUserMedia = navigator.getUserMedia || navigator.webkitGetUserMedia
|| navigator.mozGetUserMedia;

// Função para iniciar o streaming de vídeo no elemento <video>
function startStreaming(){
    if (!!window.stream) {
        videoElement.src = null;
        window.stream.stop();
    }
    var videoSource = videoSelect.value;
    var constraints = { // Configurações da API, escolhendo somente streaming de
vídeo
        audio: false,
        video: {
            optional: [{sourceId: videoSource}]
        }
    };
    navigator.getUserMedia(constraints, successCallback, errorCallback);

    // Callback de sucesso
    function successCallback(LocalMediaStream) {
        window.stream = LocalMediaStream; // make stream available to console
        videoElement.src = window.URL.createObjectURL(stream);
        videoElement.play(); // Inicia streaming
    }
```

```
    // Callback de erro
    function errorCallback(error){
        alert("navigator.getUserMedia error: ", error);
    }
    // Inicia o temporizador de captura de foto
    takePictureInterval();
}

// Função temporizador de captura
function takePictureInterval(){
    clearInterval(intervalMap);
    intervalMap = setInterval(function(){
        takePicture(); // Chama a função de captura
    }, timerSelect.value); // Tempo de captura da imagem
}

// Função captura de foto
function takePicture(){
    takePictureInterval(); // Chama o temporizador de captura, baseado no tempo
no select

    context.drawImage(video, 0, 0, videoWidth, videoHeight);
    var dataUri = canvas.toDataURL(); // Aqui é retornado a foto em Base64
}
```

3.2.1.3 Armazenamento de coordenadas no banco de dados

O IndexedDB encoraja o uso do seguinte padrão:

1. Abrir um banco de dados.
2. Criar um *ObjectStore* ao atualizar o banco.
3. Iniciar uma transação e fazer um *request* (pedido) para fazer alguma operação no banco, como inserir ou obter dados.
4. Esperar a operação ser completada ouvindo algum evento *DOM*.
5. Fazer algo com o resultado da *query* (que pode ser obtida pelo objeto request).

```
// Função IndexedDB para inserção das coordenadas no banco
insertCoordenada: function(rotaid, lat, long, arquivo, callback){

    // inicia transação
    var transaction = db.transaction(["coordenadas"],"readwrite");

    transaction.oncomplete = function(event) {
        console.log("Coordenadas inseridas");

        if (callback && typeof(callback) === "function") {
            callback(); // Executa o callback quando a transação for concluída
        }
    };
    transaction.onerror = function(event) {
        console.error("Erro ao inserir Coordenadas");
    };
    // Cria um objectStore
    var objectStore = transaction.objectStore("coordenadas");

    // Insere os dados no objectStore
    objectStore.add({rotaid: rotaid, data: DB.getDatetime(), lat: lat, long:
long, arquivo: arquivo});

}
```

3.2.1.4 Armazenamento de foto em disco

- **LocalFileSystem** tem dois métodos globais que lhe permite solicitar o acesso a um sistema de arquivos em modo seguro: *requestFileSystem*() e *resolveLocalFileSystemURL*(). Os métodos são implementados pelo objeto *window*, por tanto são instanciados globalmente.
- **FileSystem** representa um sistema de arquivos. O objeto é a sua porta de entrada para toda a API.
- **Entry** representa uma entrada em um sistema de arquivos. A entrada pode ser um arquivo ou um diretório.
- **DirectoryEntry** representa um diretório em um sistema de arquivos.
- **DirectoryReader** permite que você leia arquivos e diretórios em um diretório.
- **FileEntry** representa um arquivo em um sistema de arquivos.
- **FileError** é um erro que você pode encontrar ao usar os métodos assíncronos.

```
// Funções WebSQL para inserção das coordenadas no banco
saveFile("UTFPR", datetimeNow+"_"+latitude+"_"+longitude+".png", BINARY_ARR,
"Foto Capturada!")


// Função de salvar aquivos
function saveFile(diretorio, arquivo, conteudo, mensagem){

    if (typeof LocalFileSystem != 'undefined'){
        window.webkitRequestFileSystem(PERSISTENT,3000*1024*1024,
function(grantedBytes) {
            window.requestFileSystem(LocalFileSystem.PERSISTENT,0,gotFS, fail);
        }, function(e) {
            console.error('Error', e);
        });
    } else {
        console.log("Não foi possível salvar o arquivo.");
    }
```

```
    function gotFS(fileSystem) {
        fileSystem.root.getDirectory(diretorio, {create: true}, gotDir);
    }


    function gotDir(dirEntry) {
        dirEntry.getFile(arquivo, {create: true, exclusive: false}, gotFile);
    }


    function gotFile(fileEntry) {
        fileEntry.createWriter(gotFileWriter, fail);
    }

    function gotFileWriter(writer) {

        writer.onwriteend = function(writer) {
            if( mensagem != "" && mensagem != null){
                toast(mensagem); // Chama a função de mensagem (toast)
            }
        };
        writer.write(conteudo);
    }

    function fail(error) {
        alert(error.code);
    }
}
```

4. TESTES E RESULTADOS

O aplicativo foi experimentado em campo sob condições reais. As fotos foram capturadas num período entre 5 e 10 segundos, e o trajeto percorrido foi marcado automaticamente no mapa. No final do percurso foi gerado e exportado automaticamente no armazenamento do dispositivo os arquivos no formato *.json e .html* permitindo posteriormente a utilização dos dados coletados.

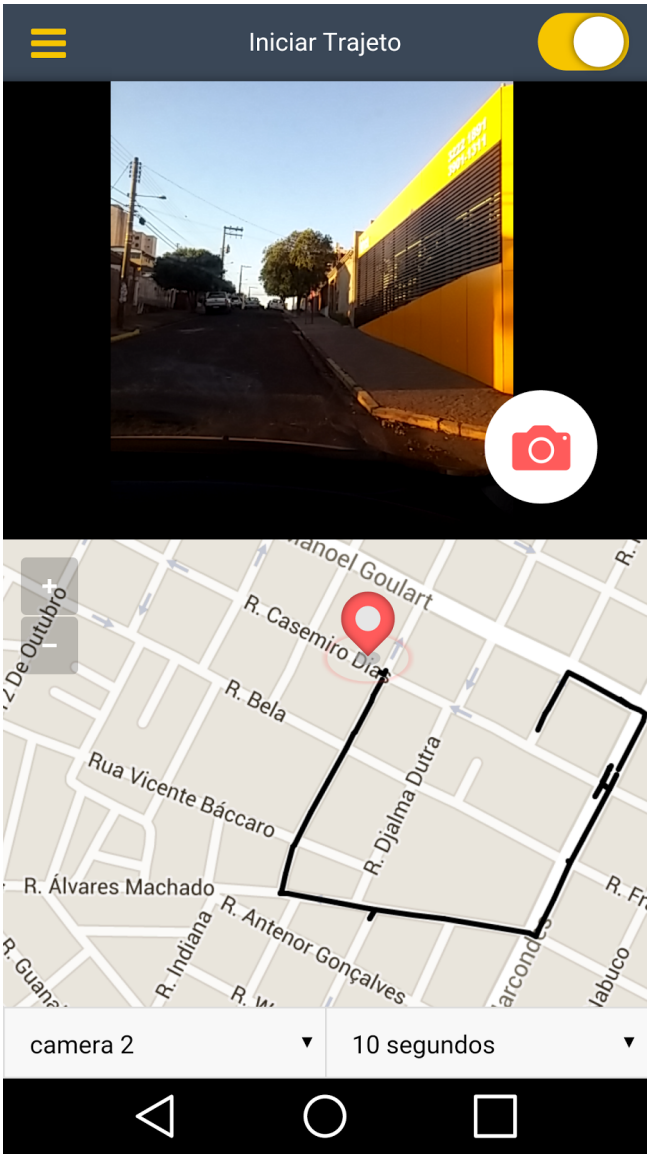

Figura 16: Tela de captura de percurso posicionada verticalmente

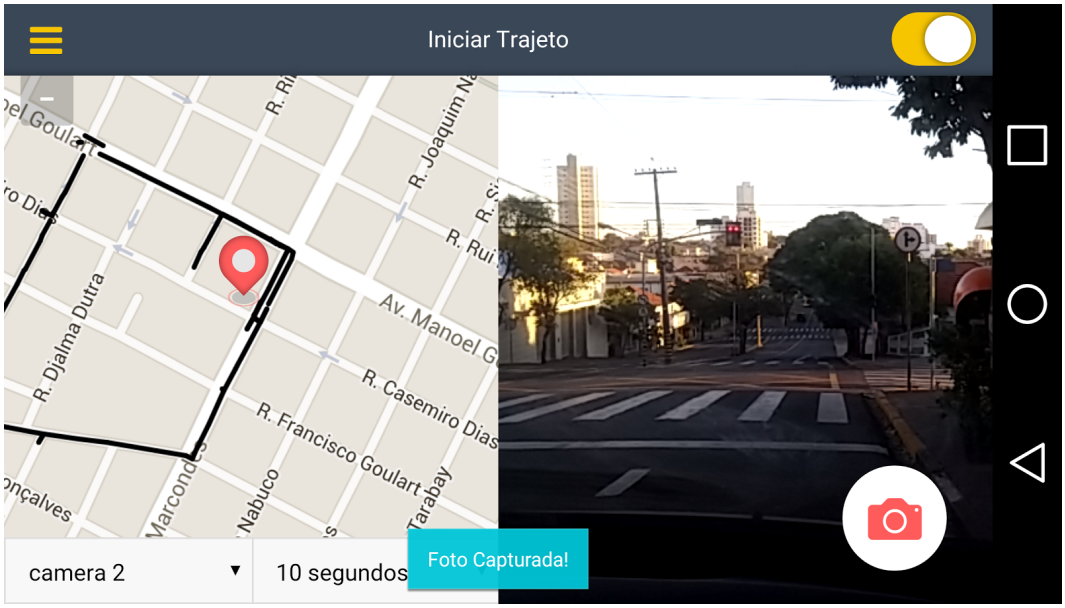

Figura 17: Tela de captura de percurso posicionada horizontalmente

Figura 18: Galeria do smartphone listando todas as fotos armazenadas do percurso

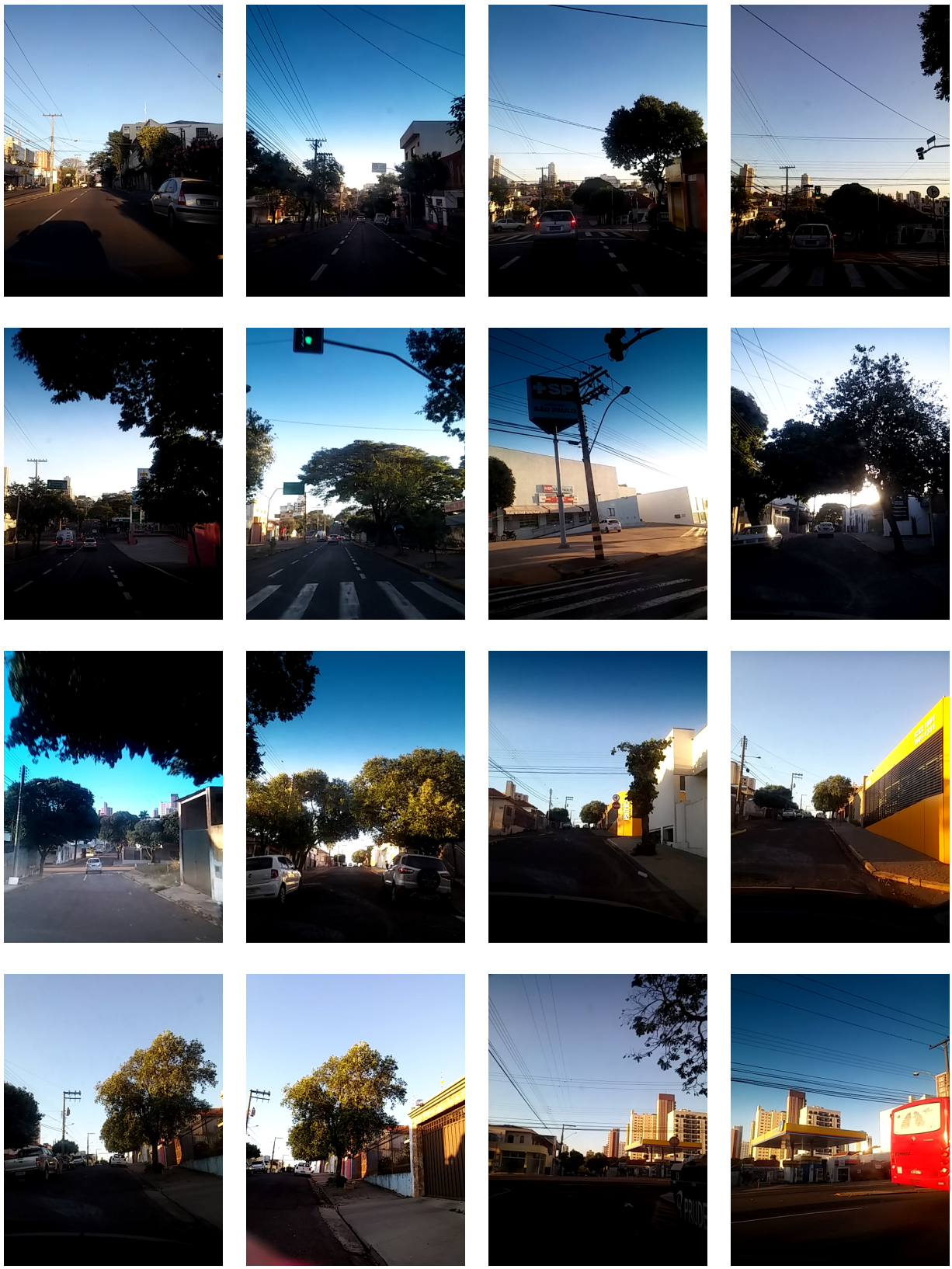

Figura 19: Sequência de fotos capturadas durante o percurso

Figura 20: Arquivos .html e .json gerados e exportados automaticamente

Registrado dia: 2/8/2015 as 17:40:39

Data	Latitude	Longitude	Foto
Sun Aug 02 2015 17:30:47 GMT-0300 (BRT)	-22.1251876	-51.39067	2015-8-2_17.30.47_-22.1251876_-51.39067.png
Sun Aug 02 2015 17:31:08 GMT-0300 (BRT)	-22.1250687	-51.39074409	2015-8-2_17.31.8_-22.1250687_-51.39074409.png
Sun Aug 02 2015 17:31:18 GMT-0300 (BRT)	-22.12475179	-51.39051229	2015-8-2_17.31.18_-22.12475179_-51.39051229.png
Sun Aug 02 2015 17:31:28 GMT-0300 (BRT)	-22.12491954	-51.39005326	2015-8-2_17.31.28_-22.12491954_-51.39005326.png
Sun Aug 02 2015 17:31:39 GMT-0300 (BRT)	-22.12499316	-51.3899394	2015-8-2_17.31.39_-22.12499316_-51.3899394.png
Sun Aug 02 2015 17:31:49 GMT-0300 (BRT)	-22.12493624	-51.39002719	2015-8-2_17.31.49_-22.12493624_-51.39002719.png
Sun Aug 02 2015 17:31:59 GMT-0300 (BRT)	-22.1249466	-51.38984104	2015-8-2_17.31.59_-22.1249466_-51.38984104.png
Sun Aug 02 2015 17:32:08 GMT-0300 (BRT)	-22.12516806	-51.38969394	2015-8-2_17.32.8_-22.12516806_-51.38969394.png
Sun Aug 02 2015 17:32:18 GMT-0300 (BRT)	-22.12565056	-51.39006663	2015-8-2_17.32.18_-22.12565056_-51.39006663.png
Sun Aug 02 2015 17:32:24 GMT-0300 (BRT)	-22.12566045	-51.39007574	2015-8-2_17.32.24_-22.12566045_-51.39007574.png
Sun Aug 02 2015 17:32:30 GMT-0300 (BRT)	-22.12568365	-51.3901005	2015-8-2_17.32.30_-22.12568365_-51.3901005.png
Sun Aug 02 2015 17:32:33 GMT-0300 (BRT)	-22.12566873	-51.39010175	2015-8-2_17.32.33_-22.12566873_-51.39010175.png
Sun Aug 02 2015 17:32:43 GMT-0300 (BRT)	-22.12571983	-51.39008129	2015-8-2_17.32.43_-22.12571983_-51.39008129.png
Sun Aug 02 2015 17:32:53 GMT-0300 (BRT)	-22.12578009	-51.39004514	2015-8-2_17.32.53_-22.12578009_-51.39004514.png
Sun Aug 02 2015 17:33:00 GMT-0300 (BRT)	-22.12572055	-51.39013048	2015-8-2_17.33.0_-22.12572055_-51.39013048.png
Sun Aug 02 2015 17:33:03 GMT-0300 (BRT)	-22.12581682	-51.39022669	2015-8-2_17.33.3_-22.12581682_-51.39022669.png
Sun Aug 02 2015 17:33:06 GMT-0300 (BRT)	-22.12599292	-51.39026938	2015-8-2_17.33.6_-22.12599292_-51.39026938.png
Sun Aug 02 2015 17:33:11 GMT-0300 (BRT)	-22.12645564	-51.39046354	2015-8-2_17.33.11_-22.12645564_-51.39046354.png
Sun Aug 02 2015 17:33:15 GMT-0300 (BRT)	-22.12668451	-51.39063648	2015-8-2_17.33.15_-22.12668451_-51.39063648.png
Sun Aug 02 2015 17:33:20 GMT-0300 (BRT)	-22.12709116	-51.39086446	2015-8-2_17.33.20_-22.12709116_-51.39086446.png
Sun Aug 02 2015 17:33:25 GMT-0300 (BRT)	-22.1274533	-51.39101681	2015-8-2_17.33.25_-22.1274533_-51.39101681.png
Sun Aug 02 2015 17:33:28 GMT-0300 (BRT)	-22.12750184	-51.39110933	2015-8-2_17.33.28_-22.12750184_-51.39110933.png
Sun Aug 02 2015 17:33:35 GMT-0300 (BRT)	-22.1274188	-51.39149317	2015-8-2_17.33.35_-22.1274188_-51.39149317.png
Sun Aug 02 2015 17:33:45 GMT-0300 (BRT)	-22.12739377	-51.39215463	2015-8-2_17.33.45_-22.12739377_-51.39215463.png
Sun Aug 02 2015 17:33:55 GMT-0300 (BRT)	-22.1273715	-51.39291573	2015-8-2_17.33.55_-22.1273715_-51.39291573.png
Sun Aug 02 2015 17:34:02 GMT-0300 (BRT)	-22.12715007	-51.39341497	2015-8-2_17.34.2_-22.12715007_-51.39341497.png
Sun Aug 02 2015 17:34:12 GMT-0300 (BRT)	-22.12708096	-51.39401369	2015-8-2_17.34.12_-22.12708096_-51.39401369.png
Sun Aug 02 2015 17:34:13 GMT-0300 (BRT)	-22.12710121	-51.39407368	2015-8-2_17.34.13_-22.12710121_-51.39407368.png
Sun Aug 02 2015 17:34:15 GMT-0300 (BRT)	-22.12700443	-51.39407633	2015-8-2_17.34.15_-22.12700443_-51.39407633.png
Sun Aug 02 2015 17:34:19 GMT-0300 (BRT)	-22.12681281	-51.39415649	2015-8-2_17.34.19_-22.12681281_-51.39415649.png
Sun Aug 02 2015 17:34:29 GMT-0300 (BRT)	-22.12653481	-51.39395036	2015-8-2_17.34.29_-22.12653481_-51.39395036.png
Sun Aug 02 2015 17:34:30 GMT-0300 (BRT)	-22.12650885	-51.39393634	2015-8-2_17.34.30_-22.12650885_-51.39393634.png

Figura 21: Representação do arquivo .html atuando como um índice

VALID


```
{
  "percurso":{
    "dia":"2015-08-02T20:40:39.132Z",
    "hora":17,
    "rotas":[
      {
        "data":"2015-08-02T20:30:47.000Z",
        "lat":-22.1251876,
        "long":-51.39067,
        "arquivo":"2015-8-2_17.30.47_-22.1251876_-51.39067.png"
      },
      {
        "data":"2015-08-02T20:31:08.000Z",
        "lat":-22.1250687,
        "long":-51.39074409,
        "arquivo":"2015-8-2_17.31.8_-22.1250687_-51.39074409.png"
      },
      {
        "data":"2015-08-02T20:31:18.000Z",
        "lat":-22.12475179,
        "long":-51.39051229,
        "arquivo":"2015-8-2_17.31.18_-22.12475179_-51.39051229.png"
      },
      {
        "data":"2015-08-02T20:31:28.000Z",
        "lat":-22.12491954,
        "long":-51.39005326,
        "arquivo":"2015-8-2_17.31.28_-22.12491954_-51.39005326.png"
      },
      {
        "data":"2015-08-02T20:31:39.000Z",
        "lat":-22.12499316,
        "long":-51.3899394,
        "arquivo":"2015-8-2_17.31.39_-22.12499316_-51.3899394.png"
      },
      {
        "data":"2015-08-02T20:31:49.000Z",
        "lat":-22.12493624,
        "long":-51.39002719,
        "arquivo":"2015-8-2_17.31.49_-22.12493624_-51.39002719.png"
      },
      ...
      ⊞
      ...
    ]
  }
}
```


Figura 22: Representação do arquivo .json exportado

5. CONSIDERAÇÕES FINAIS

Este trabalho descreveu um estudo de tecnologias abertas e padrões web que visa facilitar o *workflow* dos desenvolvedores e trazer novas possibilidades ao processo de desenvolvimento de um aplicativo híbrido utilizando APIs do HTML5.

Através da aplicação criada, foi possível solucionar a necessidade de caracterizar um circuito, esperado na pesquisa de poluentes atmosféricos em ambientes urbanos.

Além do recurso de coleta de dados para a caracterização de um determinado percurso, agora será possível um trabalho mais rico posterior a essa coleta. Através do desenvolvimento do aplicativo, os dados que foram capturados poderão ser exportados num padrão internacional *JSON* trabalhados e inseridos em pesquisas futuras.

Demonstramos ser possível criar uma aplicação que se comunique com o hardware do *smartphone* e execute as mais diversas tarefas, como a utilização da câmera, optando pela frontal ou traseira, captura de coordenadas por meio do GPS, gravação das informações em um *storage* interno e a gravação de arquivos na memória do *smartphone*. Além de ser possível o seu funcionamento tanto *online* quanto *offline*, possibilitando a coleta de coordenadas geográficas e captura de fotos até mesmo sem conexão com a internet. Deste modo, alcançamos um ambiente propício e de fácil utilização, sem perder a flexibilidade das páginas de internet, que podem funcionar em qualquer dispositivo móvel atual.

5.1 Trabalhos futuros

Sugerimos a implementação de um sistema que faça o trabalho de ler e interpretar os arquivos (.JSON) de percurso exportados pelo aplicativo desenvolvido neste projeto. A partir desses dados e juntamente com os dados de poluentes climáticos coletados pelos aparelhos do estudo de poluição urbana, é possível também gerar um mapa que permita enxergar e interpretar todo o trajeto percorrido com seus respectivos pontos e informações, indicando todas os pontos em que foram capturadas as fotos do percurso, representados através de um balão com uma determinada cor respectiva ao índice de poluição coletada ali.

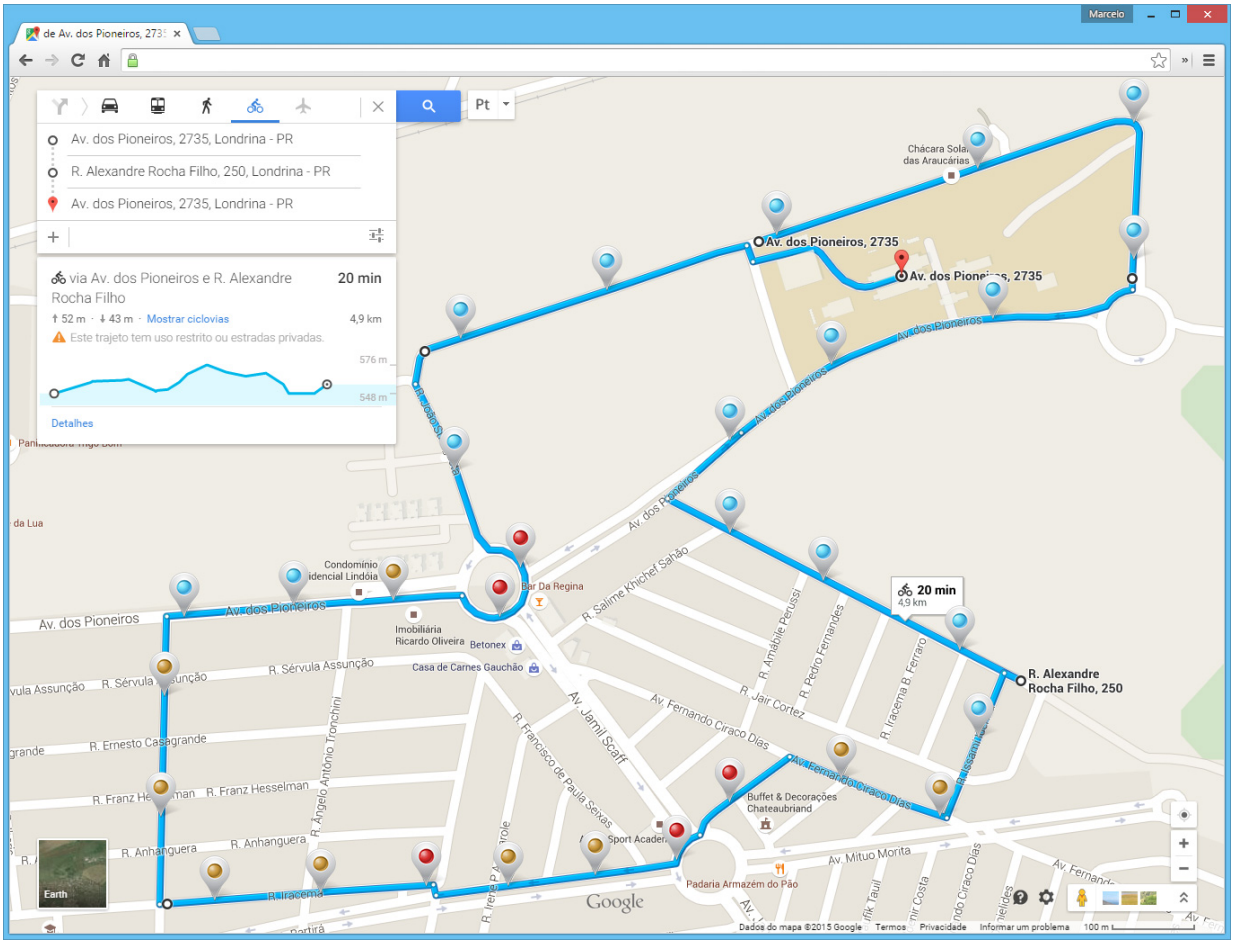

Figura 23: Tela conceito do sistema que representará os dados exportados

5.2 Limitações nos Testes

A maior limitação do projeto foi não ter conseguido realizar o teste junto ao grupo de pesquisa, por contratempos na data em que estava marcada para apresentação e desencontros com o responsável pelo projeto. Entretanto apesar dos contratempos a execução dos testes em campo não foram prejudicados e foram realizados seguindo o roteiro de desenvolvimento. A seguir veremos as limitações encontradas no dispositivo.

- Bateria do dispositivo

  A bateria de um smartphone é um módulo de hardware indispensável. Mais importante ainda quando o seu desempenho é confiada à um aplicativo como o desenvolvido neste trabalho, pois é um recurso altamente consumido, sendo por um período de relativamente alto de tempo e em um ambiente externo não controlado, podendo favorecer o aquecimento do aparelho.

- Foco da Câmera

  Como o aplicativo será utilizado especialmente em ambiente externo e em alto movimento, a precisão e a qualidade do foco processado pelo *Smartphone* é um fator muito importante na hora da captura, pois é ele quem define qual será a qualidade final da foto.

- Internet

  A qualidade de conexão durante a utilização do aplicativo influencia diretamente no uso e exibição do mapa localizado na tela de coleta e captura. Caso a conexão esteja lenta ou temporariamente indisponível, a representação do mapa e trajeto congelará.

- Armazenamento em disco

  Não é regra mas, cada foto capturada durante o percurso fica entre 200kb e 500kb. Então se a execução de coleta de um percurso for realizada por horas, é possível que o smartphone tenha sua capacidade de armazenagem totalmente limitada. O que fará diferença será a quantidade de armazenamento que o dispositivo suportará.

- Tempo de captura de foto

  Quando o tempo de captura de foto é configurado para menos de 5 segundos é possível que a foto do percurso seja comprometida, levando a replicação da foto armazenada anteriormente, pois para que o arquivo seja armazenado em disco é necessário alguns segundos de processamento.

5.3 Limitações no Desenvolvimento

- Content-Security-Policy (CSP)

  É uma camada extra de segurança que ajuda a detectar e mitigar certos tipos de ataques, incluindo Cross Site Scripting (XSS) e ataques de injeção de dados. Esses ataques são usados para tudo, desde roubo ou danificação de dados e até mesmo distribuição de *malware*.

  A partir da versão 5 do Apache Cordova essas diretivas de segurança foram requeridas e necessárias para que alguns recursos e comportamento de mídia, *scripts*, *styles* e etc, no aplicativo sejam executados. Isso é feito pela especificação dos domínios que o navegador (webview) deve considerar como fontes válidas de *scripts* executáveis. Um navegador compatível com a CSP somente executará *scripts* carregados em arquivos recebidos dos domínios especificados, ignorando quaisquer outros *scripts* (incluindo *scripts inline* e atributos de eventos no HTML).

  A seguir, a *Metatag* adicionada ao arquivo *index.html* do aplicativo deste trabalho, para interpretação do CSP:

```
<meta http-equiv="Content-Security-Policy" content="default-src *;
style-src 'self' 'unsafe-inline' chrome-extension:; script-src 'self'
'unsafe-eval' 'unsafe-inline' chrome-extension: http: https:">
```

- Webview

  Para fornecer suporte às mais modernas Web APIs utilizadas neste projeto, como o getUserMedia, foi necessário a utilização do Crosswalk em formato de *plugin* do Cordova. Sem ele essa mesma API não seria reconhecida pelo navegador (webview), impossibilitando o acesso aos recursos e configurações da câmera.

REFERÊNCIAS

ANDROID, Android Software Fundation. **Building Web Apps in WebView**. Disponível em: <http://developer.android.com/guide/webapps/webview.html>. Acesso em: 15 mai. 2015.

APACHE, Apache Software Foundation. **Apache Cordova**. Disponível em: <https://cordova.apache.org/>. Acesso em: 15 mai. 2015.

ASIAL, Asial Corporation. **Onsen UI**. Disponível em: <http://onsen.io/>. Acesso em: 16 jun. 2015.

Bond, T. C. et al., 2013: **Bounding the role of black carbon in the climate system: A scientific assessment**. J. Geophys. Res., doi: 10.1002/jgrd.50171.

CHROME, Chrome Apps. **Cordova Chrome App**. Disponível em: <https://developer.chrome.com/apps/chrome_apps_on_mobile>. Acesso em: 15 mai. 2015.

ECMA, Ecma International. **The JSON Data Interchange Format**. Disponível em: <http://www.ecma-international.org/publications/files/ECMA-ST/ECMA-404.pdf>. Acesso em: 15 mai. 2015.

EPA, United States Environmental Protection Agency. **Black Carbon**. Disponível em: <http://www.epa.gov/blackcarbon/basic.html>. Acesso em: 15 mai. 2015.

ESRI, Esri Proprietary Rights Acknowledgment. **Collector for ArcGIS**. Texto alterado. Disponível em: <http://doc.arcgis.com/pt-br/collector/>. Acesso em: 03 jun. 2015.

GOOGLE, Google, Inc. **Angular JS**. Disponível em: <https://angularjs.org/>. Acesso em: 16 jun. 2015.

IPCC, 2001. In Climate change 2001: The Sientific basis, ed. T. J. Houghton, Y. Ding, D. J. Griggs, M. Noguer, pp. 289-348. Cambridge Univ. Press, Cambridge, 2001: Third assessment report of the IPCC (http://www.grida.no/climate/ipcc_tar/wg1/160.htm).

Kingham, S., Longley, I., Salmond, J., Pattinson, W., Shrestha, K., 2013. **Variations in exposure to traffic pollution while travelling by different modes in a low density, less congested city**. Environmental Pollution 181, 211–218

Maurício Samy Silva, 2011. **HTML5 a linguagem de marcação que revolucionou a web**. Apresentação do HTML5 21-25

W3C, World Wide Web Consortium. **A vocabulary and associated APIs for HTML and XHTML**. Disponível em: <http://dev.w3.org/geo/api/spec-source.html>. Acesso em: 15 mai. 2015.

W3C, World Wide Web Consortium. **GetUserMedia API Specification**. Disponível em: <http://www.w3.org/TR/mediacapture-streams>. Acesso em: 15 mai. 2015.

W3C, World Wide Web Consortium. **Geolocation API Specification**. Disponível em: <http://dev.w3.org/geo/api/spec-source.html>. Acesso em: 15 mai. 2015.

W3C, World Wide Web Consortium. **File API**. Disponível em: <http://dev.w3.org/2009/dap/file-system/file-writer.html>. Acesso em: 15 mai. 2015.

W3C, World Wide Web Consortium. **Web SQL Database**. Disponível em: <http://www.w3.org/TR/webdatabase>. Acesso em: 15 mai. 2015.

W3C, World Wide Web Consortium. **Open Web Platform**. Disponível em: <http://www.w3.org/wiki/Open_Web_Platform>. Acesso em: 18 set. 2015.