UNIVERSIDADE FEDERAL DE MATO GROSSO
INSTITUTO DE COMPUTAÇÃO
COORDENAÇÃO DE ENSINO DE GRADUAÇÃO EM
CIÊNCIA DA COMPUTAÇÃO

RELATÓRIO DE ESTÁGIO SUPERVISIONADO

DESENVOLVIMENTO DO APLICATIVO GEOMAPPING

ERIK WILLIAM CHAGAS DA SILVA

CUIABÁ – MT
2015

UNIVERSIDADE FEDERAL DE MATO GROSSO
INSTITUTO DE COMPUTAÇÃO
COORDENAÇÃO DE ENSINO DE GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

RELÁTORIO DE ESTÁGIO SUPERVISIONADO
DESENVOLVIMENTO DO APLICATIVO GEOMAPPING

ERIK WILLIAM CHAGAS DA SILVA

Relatório apresentado ao Instituto de Computação da Universidade Federal de Mato Grosso, para obtenção do título de Bacharel em Ciência da Computação.

CUIABÁ – MT
2015

UNIVERSIDADE FEDERAL DE MATO GROSSO
INSTITUTO DE COMPUTAÇÃO
COORDENAÇÃO DE ENSINO DE GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

ERIK WILLIAM CHAGAS DA SILVA

Relatório de Estágio Supervisionado apresentado à Coordenação do Curso de Ciência da Computação como uma das exigências para obtenção do título de Bacharel em Ciência da Computação da Universidade Federal de Mato Grosso

Aprovado por:

Prof. MSc.Thiago Meirelles Ventura
Instituto de Computação
(ORIENTADOR)

Prof. MSc. Daniel Avila Vecchiato
Instituto de Computação

Esp. Noely de Oliveira Moraes
(SUPERVISORA)

DEDICATÓRIA

A toda a minha família pelo apoio irrestrito, especialmente a minha mãe Edilza, que me inspira com sua força.

AGRADECIMENTOS

Agradeço a todos os meus familiares que sempre me apoiaram durante todo este importante período da minha vida. Especialmente aos meus pais Edilza Chagas da Silva e Edvaldo Alves da Silva que fizeram de tudo para me manter nos estudos com o mínimo de sobressaltos, sabendo da dedicação quase que exclusiva que precisei assumir nesta trajetória, além de me oferecerem todo seu amor e carinho. Agradeço aos meus irmãos Wallison Chagas da Silva e Anna Luiza Chagas da Silva, pessoas maravilhosas que amo e meus amigos para a vida inteira. A Glaucia Montalvão da Silva e Rubia Elias da Silva por sua amizade e apoio. A todos os professores do Instituto de Computação por empenharem todo seu esforço na nobre missão de transmitir conhecimento. Agradeço especialmente ao professor Mauricio Fernando Lima Pereira que me deu a oportunidade de participar da iniciação científica ao me aceitar em seu grupo de pesquisa – o CAP – por dois anos e me dar a oportunidade de ser monitor de Algoritmos II. Ao professor Thiago Meirelles Ventura por aceitar ser meu orientador, mesmo sabendo do curto tempo que eu tinha para desenvolver este relatório. A todos os meus colegas de turma que passaram por essa experiência única da academia junto comigo, em especial a Michel Platini Gomes da Silva, Henrique Fortes Raia, Roberto da Silva Oliveira, que se dispuseram a me ajudar quando alguns percalços se apresentaram no período em que estive escrevendo este relatório. Aos colegas de estágio na DATALAB, sempre solícitos e prontos para sanar todas as dúvidas que surgiram durante a construção deste trabalho. Por fim, a todos que direta ou indiretamente me ajudaram a prosseguir nesta caminhada e chegar até aqui.

SUMÁRIO

LISTA DE FIGURAS **7**

LISTA DE SIGLAS E ABREVIATURAS **8**

RESUMO **9**

1. INTRODUÇÃO **10**

1.1 JUSTIFICATIVA 11
1.2 OBJETIVOS GERAIS 11
1.3 OBJETIVOS ESPECÍFICOS 11
1.4 ORGANIZAÇÃO DO TRABALHO 12

2. REVISÃO DE LITERATURA **13**

2.1 SISTEMAS OPERACIONAIS 13
2.1.1 Sistema de Armazenamento *14*
2.1.2 Entrada e Saída *14*
2.2 LINUX 15
2.3 PLATAFORMA ANDROID 15
2.4 CAMADAS DO ANDROID 16
2.5 GOOGLE MAPS 18
2.6 GEOCODING 19

3. MATERIAIS, TÉCNICAS E MÉTODOS **22**

3.1 DESENVOLVIMENTO MULTIPLATAFORMA 22
3.2 APACHE CORDOVA 23
3.3 HTML5 E CSS3 24
3.4 JAVASCRIPT E JQUERY 25
3.5 BOOTSTRAP 26
3.6 WEBSQL 27
3.7 XML 27
3.8 CLOUDAX 28
3.9 PROCESSO DE GEORREFERENCIAMENTO 29

4. RESULTADOS **31**

4.1 GEOMAPPING 31
4.2 AGENDA DE CONTATOS 32
4.3 ONDE ESTOU AGORA? 34
4.4 ANOTAÇÕES 35
4.5 DEMANDAS DE GEORREFERENCIAMENTO 36

5. DIFICULDADES ENCONTRADAS **40**

6. CONCLUSÕES **41**

7. REFERÊNCIAS BIBLIOGRÁFICAS **42**

APÊNDICE I: TRECHO DO CÓDIGO FONTE JAVASCRIPT PARA FUNCIONALIDADE "CONTATOS" **43**

APÊNDICE II: TRECHO DE CÓDIGO FONTE JAVASCRIPT COM USO DA API DO GOOGLE MAPS PARA A FUNCIONALIDADE "ONDE ESTOU AGORA?" **46**

APÊNDICE III: TRECHO DO CÓDIGO FONTE JAVASCRIPT PARA A FUNCIONALIDADE "DEMANDAS DE GEORREFERENCIAMENTO" **50**

LISTA DE FIGURAS

FIGURA 1: PILHA DO ANDROID (GARGENTA, 2011, P. 8) 18
FIGURA 2: REPRESENTAÇÃO DO SISTEMA DE COORDENADAS ADOTADO PELO GOOGLE MAPS, INDICANDO O CENTRO DO MUNDO NA COSTA OESTE DA ÁFRICA (SVENNERBERG, 2010, P. 5) 19
FIGURA 3: AMBIENTE DE DESENVOLVIMENTO INTEL XDK 23
FIGURA 4: GRID PADRÃO DO BOOTSTRAP COM 12 COLUNAS (SPURLOCK, 2013, P. 3) 27
FIGURA 5: DIAGRAMA DE FUNCIONAMENTO DOS PEDIDOS DE DEMANDA E GEORREFERENCIAMENTO NO GEOMAPPING 30
FIGURA 6: PAINEL ESQUERDO DE FUNCIONALIDADES E TELA INICIAL DO GEOMAPPING 32
FIGURA 7: TELA DE AGENDA DE CONTATOS DO GEOMAPPING 33
FIGURA 8: TELA COM MAPA DO PONTO ATUAL E PAINEL DIREITO COM DADOS DE COORDENADAS E ENDEREÇO 35
FIGURA 9: TELA DE INSERÇÃO DAS ANOTAÇÕES DO USUÁRIO 36
FIGURA 10: PAINEL DIREITO DA FUNCIONALIDADE "DEMANDAS DE GEORREFERENCIAMENTO" 38
FIGURA 11: MODELAGEM FÍSICA DOS DADOS DE GEORREFERENCIAMENTO DE PONTOS 39

LISTA DE SIGLAS E ABREVIATURAS

HTML	*Hyper Text Markup Language*
CSS	*Cascading Sheet Style*
PDA	*Personal Digital Assistant*
CEP	Código de Endereçamento Postal
API	*Application Programming Interface*
E/S	Entrada e Saída
CD-ROM	*Compact Disc-Read Only Memory*
CPU	*Central Processing Unit*
GPL	*General Public License*
ASL	*Apache Software License*
GPS	*Global Positioning System*
SGL	*Scalable Games Language*
SSL	*Secure Sockets Layer*
JSON	JavaScript *Object Notation*
CDT	*Chrome Developers Tools*
SDK	*Software Development Kit*
IMEI	*International Mobile Equipment Identity*
XML	*eXtensible Markup Language*
SQL	*Structured Query Language*

RESUMO

As atividades realizadas durante o período de estágio supervisionado consistiram na codificação de um aplicativo móvel multiplataforma denominado Geomapping para atender a requisitos de georreferenciamento dos pontos geográficos de interesse das aplicações móveis em atividade na empresa DATALAB. Para o desenvolvimento deste aplicativo, foi utilizado o ambiente denominado Intel XDK. São adotadas neste ambiente as tecnologias HTML (*Hyper Text Markup Language*) e CSS (*Cascading Sheet Style*) para estruturação e estilização do aplicativo, além do Javascript para as funcionalidades do aplicativo. O fato do ambiente de desenvolvimento adotado ser novo e ainda pouco experimentado gerou algumas dificuldades na utilização de seu *debugger* padrão, com travamentos recorrentes do ambiente nos testes realizados em alguns dos módulos do Geomapping. Obteve-se com resultados, todavia, a criação do *layout* do aplicativo, além da execução de funcionalidades como agenda de contatos, recurso de armazenamento para anotações e exibição de posição atual por meio de dados coletados por GPS (*Global Positioning Service*), com endereço aproximado relativo ao ponto geográfico captado e uma parte da funcionalidade de georreferenciamento dos pontos. Conclui-se que os ambientes de desenvolvimento multiplataforma para aplicativos móveis oferecem boas vantagens em relação ao desenvolvimento nativo, por dotar uma só linguagem de programação orientada a objetos – o Javascript, através de APIs (*Application Programming Interface*) – de controles dos recursos nativos aos dispositivos móveis indispensáveis à criação das aplicações móveis.

1. **INTRODUÇÃO**

A empresa em que foi realizado este estágio supervisionado é a DATALAB Mobility, que tem como atividade principal o desenvolvimento de *softwares* para ambiente coorporativo. A DATALAB tem grande experiência na área de negócios empresariais, sendo uma das pioneiras no Brasil em desenvolvimento e distribuição de softwares móveis através do uso de PDAs (*Personal Digital Assistant*) e *smartphones* (DATALAB, 2012).

No catálogo de aplicações da DATALAB para ambiente móvel, é grande o uso de dados de posição geográfica para diversos fins e cada uma destas aplicações realiza individualmente a captação dos pontos geográficos de interesse, além da persistência destes dados em conjunto com um perfil específico relativo a cada aplicação na estrutura de servidor de banco de dados em nuvem da empresa, a CLOUDAX.

Surgiu a necessidade de se construir uma aplicação móvel, denominada Geomapping, que fosse capaz de atender as demandas de georreferenciamento de pontos de maneira única para todos os aplicativos que utilizam serviços de geolocalização na empresa, cadastrando dados sobre os pontos que são de interesse comum, como latitude, longitude e campos de endereço como rua, CEP, etc, variando apenas as informações específicas de cada aplicação. No aplicativo móvel Bus.AO da DATALAB – que possui como uma de suas funcionalidades exibir em um mapa pontos de ônibus do transporte público e pontos de recarga de passagem –, além das informações sobre localização e endereço, é necessário armazenar dados referentes a pontos cadastrados para o sistema como um campo “tipo”, utilizado para indicar se o ponto cadastrado é de ônibus ou de recarga.

Atualmente, os aplicativos móveis desenvolvidos pela DATALAB são em sua totalidade construídos utilizando-se o ambiente de desenvolvimento oficial da Google para programação de dispositivos com sistema operacional Android, o SDK (*Software Development Kit*). Há, porém, uma demanda para que os aplicativos da empresa ofereçam suporte a outros sistemas operacionais móveis, principalmente iOS e Windows Phone, sendo necessária a construção de aplicativos que sejam mutiplataforma, possuindo também versões para estas plataformas móveis. Uma solução apropriada para este cenário é o desenvolvimento de *webapps*, que utiliza os

conceitos das tecnologias *web* para gerar aplicações que funcionam em múltiplas plataformas, a partir de um mesmo código fonte.

1.1 Justificativa

É possível verificar que baseado no mercado de negócios em que a empresa DATALAB está inserida, e possuindo ela diversas aplicações que se utilizam em grande medida de dados geográficos e georreferenciamentos, são significativos os benefícios de se ter um aplicativo único, dotado desta capacidade e que sirva as mais variadas aplicações móveis da empresa, centralizando uma atividade recorrente em apenas um lugar.

O desenvolvimento de aplicativos móveis multiplataforma representa uma mudança na forma de construção de aplicativos móveis na empresa, dado que se tem sido desenvolvido até hoje de maneira nativa e prioritariamente para o sistema operacional móvel Android. Pretende-se, portanto dinamizar o desenvolvimento de aplicações móveis, tornando-se desnecessário o conhecimento das linguagens para desenvolvimento nativas de cada um dos sistemas operacionais móveis, para disponibilizar versões a cada um deles.

1.2 Objetivos Gerais

Verificando a importância do processo de captação de pontos geográficos e seu posterior georreferenciamento dentro da empresa DATALAB, este trabalho tem como objetivo geral relatar os conhecimentos necessários para a construção de um aplicativo móvel e multiplataforma utilizando tecnologias *web* e com a capacidade de suprir os requisitos exigidos do aplicativo Geomapping, utilizando o ambiente de desenvolvimento de aplicativos multiplataformas Intel XDK.

1.3 Objetivos Específicos

Para realizar o desenvolvimento do aplicativo móvel Geomapping, este trabalho conta com os seguintes objetivos específicos:

- Estudar os principais conceitos das tecnologias HTML5, CSS3 e Javascript;

- Estudar os principais conceitos da API Apache Cordova, de manipulação de recursos nativos dos sistemas operacionais móveis, da biblioteca para Javascript e CSS3, respectivamente, denominadas jQuery e Bootstrap;
- Desenvolvimento do aplicativo móvel e multiplataforma Geomapping;
- Testes de validação nos módulos desenvolvidos do aplicativo Geomapping;

1.4 Organização do Trabalho

Os capítulos deste trabalho estão organizados da seguinte forma:

Capítulo 2: Faz uma revisão literária dos temas abordados neste trabalho.

Capítulo 3: Lista e descreve os materiais, métodos e técnicas utilizados.

Capítulo4: Descreve os resultados obtidos, aonde são apresentados o *layout* e funcionalidades desenvolvidas na construção do aplicativo móvel Geomapping.

Capítulo 5: São descritas as dificuldades encontradas durante a execução dos trabalhos.

Capítulo 6: O último capítulo descreve as conclusões obtidas com este trabalho, fazendo uma reflexão sobre a sua importância e citando os próximos passos a serem realizados.

2. REVISÃO DE LITERATURA

Os sistemas operacionais estão presentes na vida de todo usuário de sistemas computacionais, dado a importância da sua tarefa de oferecer abstrações lógicas – tanto para o usuário final, quanto para os desenvolvedores de aplicações – e para o hardware do sistema. Este capítulo apresenta os sistemas operacionais, bem como define seus principais aspectos. O Android, um sistema operacional para dispositivos móveis, também é introduzido neste capítulo, assim como o conceito de kernel do sistema operacional Linux, componente integrante e crucial no Android.

Esta revisão de literatura introduz ainda a API de geolocalização com grande penetração nos sistemas *web* baseados em georreferenciamento (SVENNERBEGER, 2010, p. 1), o Google Maps.

2.1 Sistemas Operacionais

Deitel (1992, p. 3), categoriza um sistema operacional como sendo o conjunto de programas, implementado como *software* ou *firmware*, que tornam o *hardware* utilizável. O *hardware* oferece capacidade computacional bruta. Os sistemas operacionais disponibilizam convenientemente tais capacidades aos usuários, gerenciando cuidadosamente o *hardware* para que se obtenha uma performance adequada.

Por outro lado, Tanenbaum (2009, p. 2) apresenta uma definição menos assertiva, ao apontar que é difícil definir o que é um sistema operacional além de dizer que é o software que executa em modo núcleo – e mesmo isso nem sempre é verdade – dado que os sistemas operacionais realizam basicamente duas funções não relacionadas: fornecer aos programadores de aplicativos um conjunto de recursos abstratos claros em vez de recursos confusos de *hardware* e gerenciar esses recursos de *hardware*. Dependendo do tipo de usuário, ele vai lidar mais com uma função ou outra.

De qualquer forma, podemos verificar a importância dos sistemas operacionais, posto que eles, em resumo, são os responsáveis por prover interface de acesso aos recursos de *hardware* para os aplicativos executados em modo de usuário, enquanto fazem a gestão destes recursos de maneira a oferecer eficiência e alto

desempenho ao sistema. Os próximos tópicos apresentam alguns aspectos importantes dos sistemas operacionais.

2.1.1 Sistema de Armazenamento

Segundo Tanenbaum (2009, p. 158), os arquivos são um mecanismo de abstração que oferecem meios de armazenamento e leitura de informações no disco, de modo que isole o usuário dos detalhes sobre como e onde a informação está. Para cada arquivo, existe um nome que deve obedecer a uma convenção, que varia de sistema para sistema. Não existe um padrão de nomeação dos arquivos entre os sistemas operacionais dado que, por exemplo, o UNIX, diferencia letras minúsculas de maiúsculas e não exige extensão ao fim do nome do arquivo, características não respeitadas pelo Windows.

Tanto Windows, como sistemas baseados em UNIX ademais, tratam os arquivos como uma sequência desestruturada de bytes, não atribuindo a priori nenhum significado a esta sequência, deixando para o programa em nível de usuário fazer este serviço.

Constituem com parte do sistema de arquivos também, os diretórios, que são hierárquicos ao compor um esquema diretório raiz, aonde a partir de um nó raiz, podem ser criados vários níveis de diretórios, cada qual contendo inúmeros arquivos.

2.1.2 Entrada e Saída

É responsabilidade dos sistemas operacionais o controle de todos os dispositivos de E/S (entrada e saída) de um computador, além da oferta de todas as abstrações como processo, endereçamento de memória e arquivos.

Segundo Tanenbaum (2009, p. 203), os dispositivos de E/S podem ser categorizados de forma genérica em dois tipos: os dispositivos de blocos e os dispositivos de caractere. A primeira categoria corresponde aos dispositivos que armazenam informações em blocos de tamanho fixo, cada um com seu próprio endereço, de maneira que todas as transferências se dão em unidades de um ou mais blocos inteiros. O outro tipo de dispositivo de E/S é o de caractere, no qual é enviado

ou recebido um fluxo de caracteres, sem qualquer estrutura de blocos, não endereçável e sem operações de posicionamento.

2.2 Linux

O Linux é um sistema operacional proveniente do UNIX, tendo sido concebido pelo então estudante de ciência da computação finlandês Linus Trovald, com sua primeira versão tendo sido lançada em 1991. A partir deste ano, em 1994 e em 1996, o Linux ganhou grandes atualizações e melhorias, chegando em 1996 ao número de cerca de 470 mil linhas de código em linguagem C e oito mil linhas de código em linguagem Assembly, adicionando diversas características à versão base, entre elas, suporte a arquiteturas de 64 bits, multiprogramação simétrica, novos protocolos de redes, dentre outras características (TANENBAUM, 2009, p. 447).

O kernel do Linux está diretamente relacionado ao hardware, permitindo interações com os dispositivos de E/S, com a unidade de gerenciamento de memória e controla o acesso da CPU (*Central Processing Unit*) a eles, segundo Tanenbaum (2009, p. 453). Há atualmente inúmeras distribuições que se utilizam do kernel do Linux, sendo o Android um exemplo de sistema operacional com baseado em Linux. A seguir, a plataforma Android será apresentada.

2.3 Plataforma Android

A plataforma Android é um ambiente de software escrito para dispositivos móveis. Inclui um sistema operacional baseado em kernel Linux, uma interface de usuário, aplicativos de usuário, bibliotecas de códigos, *frameworks* de aplicativo, suporte a multimídia, entre outros.

O Android possui duas licenças diferentes, a GPL (*General Public License*), do kernel Linux de código fonte aberto e toda a plataforma restantes, sobre a licença ASL (*Apache Software License*). Os dois modelos de licença são orientados a código fonte aberto, sendo a última mais amigável ao uso comercial (ABLESON *et. al.*, 2012, p. 9).

Segundo Lecheta (2009, p. 26), o sistema operacional do Android foi baseado na versão 2.6 do kernel Linux, sendo este componente o responsável por

gerenciar a memória, os processos, *threads* e a segurança dos arquivos e pastas, além de redes e drivers.

Cada aplicativo no Android dispara um novo processo no sistema operacional. Alguns deles podem exibir uma tela para o usuário, e outros podem ficar em execução em segundo plano por tempo indeterminado. Diversos processos e aplicativos podem ser executados simultaneamente, e o kernel do sistema operacional é o responsável por realizar todo o controle de memória.

2.4 Camadas do Android

Todos os recursos disponíveis no Android podem ser representados pela sua pilha de tecnologias. De acordo com Ableson et. al. (2012, p. 10-11), esta pilha segue a estrutura apresentada a seguir:

1. Um **kernel Linux**, fornecendo uma camada fundamental de abstração de *hardware*, além de serviços base como gerenciamento de processos, memória e sistema de arquivos. O kernel é onde os *drivers* específicos de *hardware* são implementados – recursos como Wi-Fi e Bluetooth são encontrados aqui. A camada Android é projetada para ser flexível, com muitos componentes opcionais que dependem muito da disponibilidade do hardware específico de um dado dispositivo. Esses componentes incluem recursos como teclas sensíveis ao toque, câmeras, receptores GPS (*Global Positioning System*) e acelerômetros.
2. **Bibliotecas de código proeminentes**, incluindo o seguinte:
    i. Tecnologia de navegador do WebKit, o mesmo mecanismo de código fonte aberto por trás dos navegadores Safari do Apple Machintosh e Safari Mobile do iPhone. O WebKit se tornou o padrão de fato para a maioria das plataformas móveis.
    ii. Suporte a banco de dados relacional via SQLite.
    iii. Suporte gráfico avançado, incluindo tecnologia em duas dimensões e três dimensões, animações com a *Scalable Games Language* (SGL).
    iv. Suporte a áudio e vídeo com o OpenCORE da companhia PacketVideo e o framework de mídia Stagefright do Google.

v. Capacidade *Secure Sockets Layer* (SSL) do projeto Apache, com estabelecimento de links encriptados entre servidor e cliente.

3. **Conjunto de gerenciadores** que fornece serviços para:
   i. Atividades e visualizações
   ii. Janelas
   iii. Serviços baseados em localização
   iv. Telefonia
   v. Recursos
4. O **runtime do Android**, que fornece:
   i. Pacotes básicos Java para um ambiente quase completo de programação Java
   ii. Dalvik *Virtual Machine,* que emprega serviços do kernel Linux para fornecer um ambiente de hospedagem de aplicativos Android.

Ao utilizar o kernel do Linux, o Android se beneficia de diversos níveis de abstrações, sem seja preciso ao desenvolvedor se preocupar com as características em nível de *hardware*. Além disso, o fato do Android ter sido construído com base em Linux lhe garante características como portabilidade, segurança e grande disponibilidade de bibliotecas. A portabilidade no Android é alcançada porque o Linux é fácil de compilar em diversos *hardwares*, devido a sua construção ter sido feita em código altamente portável na linguagem C, permitindo a empresas terceiras, portar o Android para uma variedade de dispositivos. Da mesma forma, a segurança do Android depende em grande medida do Linux, que pode ser considerado um sistema seguro, tendo sido testado em diversos ambientes por décadas (GARGENTA, 2011, p. 7-8).

A Figura 1 demonstra uma representação da pilha do Android, consistindo em quatro grupos principais: o kernel do Linux; as bibliotecas utilizadas no Android; a camada de *framework* da aplicação contendo vários serviços que provêm recursos com localização, sensores, Wi-Fi, camada de telefone; e por fim a camada aonde são executados os aplicativos desenvolvidos para o usuário final.

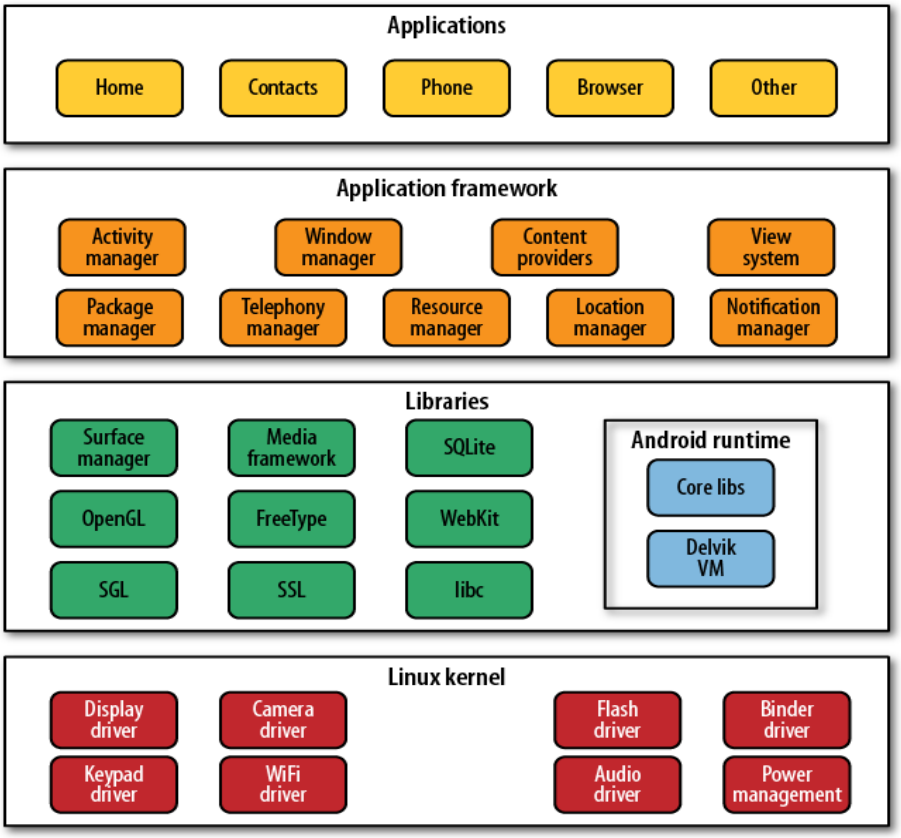

Figura 1: Pilha do Android (GARGENTA, 2011, p. 8)

2.5 Google Maps

O Google Maps é uma API que foi inaugurada em fevereiro de 2005 pela Google com um *post* em seu *blog* oficial. O serviço foi desenvolvido por dois irmãos dinamarqueses, Lars e Jens Rasmussen, em uma companhia dedicada a criar aplicações baseadas em mapas, a Where 2 Technologies, adquirida pelo Google em outubro de 2004.

Segundo Svennerberg (2010, p.3), a dinâmica de funcionamento do Google Maps em seus detalhes pode ser considerada, em geral, simples. Isto porque, as subdivisões de imagens apresentadas são carregadas por meio de chamadas Ajax e posteriormente inseridas nas páginas HTML que utilizam o serviço. Para garantir sua interatividade, a cada navegação pelo mapa, a API do Google Maps envia novas coordenadas e seus níveis de zoom a uma nova chamada Ajax que, por sua vez, retorna novas imagens. A API consiste completamente em arquivos da linguagem de

programação Javascript que possuem classes, métodos e propriedades que, ao serem utilizados, dizem como o mapa apresentado deve se comportar.

O padrão de coordenadas adotado na API, a despeito dos diferentes sistemas existentes, é o *Word Geodetic System* 84, por ser o mesmo que o sistema de GPS (*Global Positioning System*) utiliza. As coordenadas são expressas em números decimais, sobre eixos-x e eixos-y, formando um plano cartesiano no qual o eixo-y é a primeira entrada do par de coordenadas e o eixo-x a segunda entrada, representando respectivamente latitude e longitude.

A longitude permeia o espaço de norte a sul, enquanto que a longitude faz referência a pontos de oeste a leste, assim como ilustrado na Figura 2.

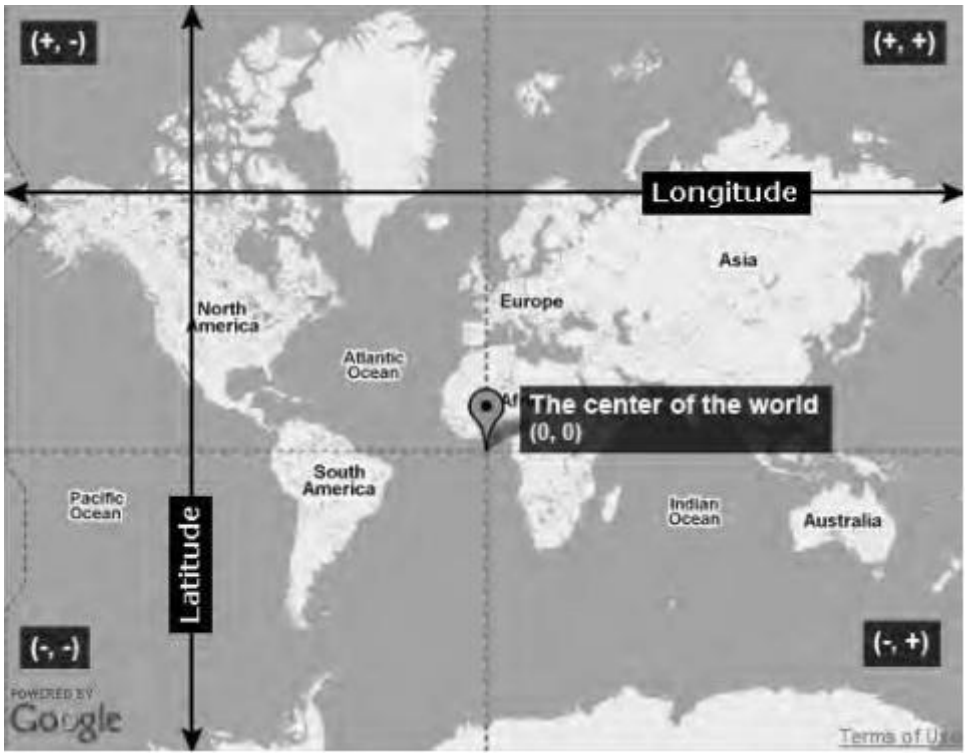

Figura 2: Representação do sistema de coordenadas adotado pelo Google Maps, indicando o centro do mundo na costa oeste da África (SVENNERBERG, 2010, p. 5)

2.6 Geocoding

Uma tarefa comumente requerida dos sistemas baseados em geolocalização é a de oferecer como entrada um endereço de uma localidade que não se conhece exatamente a posição e, receber como saída uma representação gráfica do ponto exato em um mapa do lugar aonde se deseja chegar.

Tarefas com este cunho são tratadas na API do Google Maps em uma parte integrada denominada Geocoding. Este serviço é disponibilizado de maneira gratuita, apesar de haver limitações para fazer requisições, devido a seu gasto considerável de processamento (Svennerberg, 2010, p. 211).

O acesso aos serviços do Geocoder é realizado a partir de uma chamada do método *geocode* em linguagem de programação Javascript, que passam dois valores juntamente com variáveis de resultados (*results*) e *status*. Segundo Svennerberg (2010, p. 216), há uma lista de possíveis códigos de status, assim como descritos a seguir:

- **OK**: Este código indica que tudo ocorreu bem e o resultado da requisição será retornado;
- **ZERO_RESULTS**: Neste caso, a requisição executada corretamente, porém não gerou nenhum resultado, possivelmente porque o endereço não exista;
- **OVER_QUERY_LIMIT**: Este *status* indica que o limite de requisições foi extrapolado, sendo ele de 2.500 por dia, na versão atualmente testada da API na versão livre;
- **REQUEST_DENIED**: Neste caso, a requisição foi recebida e negada por alguma razão, mais comumente devido a um parâmetro denominado *sensor* está faltando;
- **INVALID_REQUEST**: Este *status* indica que há algo de errado na requisição realizada, possivelmente o endereço, com o tipo *latlang* nativo da API e que representa um par de coordenadas, não foi definido corretamente.

Se não houveram problemas com o parâmetro *status*, os resultados obtidos pelo serviço de Geocoding devem ser posteriormente interpretados. O retorno da variável *results* possuirá um objeto JSON (*Java Script Object Notation*), que pode conter mais de um resultado, sendo eles portanto, atribuídos a um vetor de resultados. Este vetor contém um número de campos, assim com descrito por Svennerberg (2010, p. 217):

- **types**: este vetor contém qual é o tipo da localização o resultado retornado é, se por exemplo, um país ou estado;

- **formatted_address**: este campo é do tipo *string* contendo o endereço formatado, estendido, completo e legível. Uma pesquisa por “Cuiabá” retornaria “Cuiabá, MT, Brasil”;
- **address_components**: este vetor contém diferentes fatos a respeito da localização. Cada parte listada em *formatted_address* é um objeto neste vetor com variável denominada *long_name*, *short_name* e *types*;
- **geometry**: este campo é um objeto com várias propriedades. A principal é a *location*, contendo a posição do endereço em objeto do tipo LatLang.

Alternativamente, de posse dos dados geográficos coletados pelo GPS, é possível obter por intermédio da API do Google Maps um endereço aproximado relativo ao ponto oferecido como entrada. A este processo dá-se o nome de Geolocalização reversa e é justamente esta modalidade de aplicação oferecida pela API do Google Maps que é explorada na execução do Geomapping, assim com será apresentado na seção de resultados neste relatório.

3. MATERIAIS, TÉCNICAS E MÉTODOS

Foram definidos no capítulo anterior os conceitos básicos para o desenvolvimento das atividades deste trabalho de conclusão de curso. Este capítulo tem, por sua vez, o objetivo de expor os materiais, métodos e técnicas utilizadas no desenvolvimento deste trabalho.

3.1 Desenvolvimento multiplataforma

A maneira de se desenvolver aplicações para dispositivos móveis, utilizando-se de ferramentas para aplicativos nativos com os kits de desenvolvimento de *softwares* (SDK) para Android, Windows Phone e iOS, é ainda a mais comum de ser executada (IBM, 2012, p. 2). O problema deste método é que, toda vez que há a necessidade de desenvolver um software multiplataforma, há novo esforço de trabalho para se desenvolver em plataformas díspares, que utilizam linguagens diferentes e não permitindo, portanto, nenhum reaproveitamento de código.

Uma maneira de sanar este problema – evitando a necessidade de esforço de desenvolvimento utilizando os diversos ambientes para atingir as múltiplas plataformas existentes – é a utilização de ferramentas de construção de aplicativos baseados em *web*, também chamados de *webapps*, aonde são criadas páginas *web*completas e otimizadas para dispositivos móveis com as linguagens da ambiente *web*, como HTML, CSS e Javascript, além de um *design* construído para se assemelhar a um aplicativo nativo e total acesso aos recursos do sistema operacional móvel em questão, sendo posteriormente executadas a partir de um atalho na tela do dispositivo móvel, tornando-o indistinguível de um aplicativo construído de forma nativa.

A DATALAB possui a demanda de construção de aplicativos comerciais multiplataforma, e gostaria de não despender o tempo necessário para treinamento dos seus programadores para as várias ferramentas de desenvolvimento das diversas plataformas móveis existentes. Para tanto, foi utilizado para a construção do

aplicativo Geomapping neste trabalho, a ferramenta de desenvolvimento de *webapps*, denominado Intel XDK.

O Intel XDK permite que se desenvolva, teste e implante aplicativos baseados em HTML5 híbridos, fazendo uso de diversos *frameworks* e linguagens baseadas em *web*. É parte integrante do Intel XDK diversas tecnologias que, juntas resultam num ambiente de desenvolvimento de aplicativos multiplataformas. O Intel XDK é formado por componentes como ADB *debugger*, o Apache Cordova, o editor de textos Brackets, além de possuir suporte a banco de dados do lado do cliente, o WebSQL.

A Figura 3 é a tela inicial do Intel XDK em sua versão mais recente, a 1621, utilizada neste projeto. As próximas seções são destinadas ao detalhamento destas ferramentas de desenvolvimento que abrangem o Intel XDK.

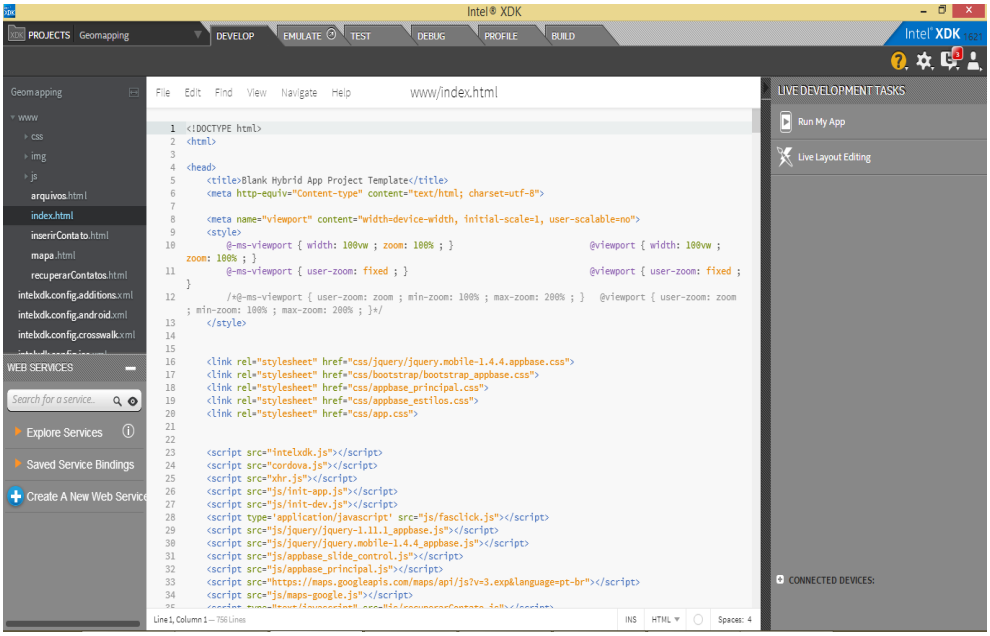

Figura 3: Ambiente de desenvolvimento Intel XDK

3.2 Apache Cordova

O Apache Cordova é o *framework* responsável por disponibilizar acesso aos recursos do dispositivo móvel por meio de APIs, para componentes com acelerômetro, câmera, contatos, geolocalização, armazenamento, etc. A partir das APIs, é possível manipular os diversos componentes dos dispositivos por meio de funções a serem invocadas e executadas em linguagem Javascript.

O Apache Cordova permite, por exemplo, que sejam adicionados novos contatos na agenda telefônica local dos dispositivos móveis suportados, com a passagem de um objeto como parâmetro, utilizando a notação JSON (JavaScript *Object Notation*) em um método *contacts.create()*. Da mesma maneira, o *framework* permite a coleta de dados de longitude e latitude atuais dos dispositivos móveis, bem como é capaz de informar a posição atual, enquanto ela mudar (MYER, 2012, p. 11).

Estas duas características disponíveis no Apache Cordova são exploradas no projeto do Geomapping, na medida em que são requeridos nos módulos do aplicativo, que se exiba a posição geográfica atual do usuário em um mapa e uma lista de contatos para telefonar, funcionalidades oferecidas por intermédio dos recursos nativos do dispositivo móvel como o GPS e os dados dos contatos armazenados localmente.

3.3 HTML5 e CSS3

O HTML5 é a versão mais recente da linguagem de marcação HTML. O HTML é o responsável por oferecer a estrutura básica de *layout* dos aplicativos desenvolvidos no Intel XDK. Por meio desta linguagem, foram definidos neste projeto, componentes de *layout* para os módulos do Geomapping como cabeçalhos, rodapés, botões, campos de entrada e saída de texto, além de estabelecer posicionamento dos ambientes para as interações com os mapas, sincronização de dados geográficos coletados com a infraestrutura de nuvem da DATALAB.

As definições de estruturação no HTML são realizadas por meio de um conjunto de *tags*, uma de abertura e outra de finalização, que estabelecem padrões de exibição do conteúdo entre essas *tags*. As *tags* obrigatórias em todos os documentos HTML são:

- <html>, responsável pela delimitação de início e fim do documento;
- <head>, responsável por indicar parâmetros de configuração do documento;
- <body>, responsável por conter o corpo do documento, com todo o seu conteúdo principal.

Além destas obrigatórias, várias são utilizadas para exibição de títulos, tabelas, listas, *links,* etc. A partir da definição da estrutura básica de *layout* construída

em HTML5, utilizou-se como ferramenta de estilização do aplicativo CSS3, definindo características como fontes, cores, posicionamentos e espaçamentos.

Os estilos são adicionados ao documento HTML seguindo uma regra, considerada a unidade básica de uma folha de estilo CSS. A sintaxe de uma regra CSS possui um seletor – elemento alvo da regra CSS declarada –, a declaração da regra entre chaves, sendo que a declaração consiste em uma propriedade – que define a característica do seletor a ser estilizada – seguido de seu valor – que é a quantificação ou qualificação da propriedade (SILVA, 2012, p. 26).

A DATALAB possui um modelo para seus aplicativos, definidos como protótipos, e a confecção do aplicativo Geomapping foi construído seguindo as regras definidas no projeto.

3.4 Javascript e jQuery

O Javascript é uma linguagem de *script*, orientada a objetos e atualmente tem ampla utilização em projetos de aplicações *web*. Assim com foi dito na seção anterior, o HTML e o CSS têm funções bem definidas, sendo o primeiro responsável por oferecer uma estrutura da página *web* – no caso deste projeto, estrutura do aplicativo –, enquanto que o segundo adiciona estilos a estas estruturas.

O uso dos dois formatos não é suficiente para construção de um sistema *web*, ou mesmo um aplicativo no Intel XDK. Isso porque o Javascript é o encarregado de cumprir as tarefas de manipulação e processamento de dados (SILVA, 2010, p.23). A partir de um código escrito em Javascript, são chamados os diversos métodos de manipulação de dados dispositivos móveis – como status de bateria, contatos, sistema de arquivos, geolocalização, informação sobre conexão de internet, etc – fornecidos pelo Apache Cordova.

Já o jQuery é uma biblioteca que adiciona novas características ao Javascript, de maneira a tornar mais fáceis de serem executadas, como a manipulação de eventos.

Os eventos em Javascript são parte essencial tanto para aplicações *web*, quanto para *webapps* desenvolvidos na plataforma Intel XDK. Essas funções são específicas para captar as interações que o usuário faz no aplicativo e as ações a serem tomadas a partir dos comandos do usuário. O uso do jQuery simplifica a

manipulação de eventos consideravelmente do que se fossem feitas apenas em Javascript (MACRAE, 2013, p. 1).

Além da simplicidade no tratamento de eventos, o jQuery torna possível a aplicação de animações variadas a telas e painéis desenvolvidos em HTML para as aplicações *web* e para os *webapps*. Foi utilizado no aplicativo Geomapping, principalmente, o efeito de slide de painéis de esquerda e direita, atendendo aos parâmetros de design definidos em projeto.

3.5 Bootstrap

Os aplicativos para a plataforma móvel devem ser capazes de oferecer grande capacidade de adaptação no que tange a seu *layout*. O projeto do aplicativo Geomapping prevê que a aplicação deve ser capaz de adaptar a disposição de suas funcionalidades na tela a depender-se do tamanho do dispositivo utilizado.

Estes módulos devem, além de serem executados em *smartphones*, serem executados também em *tablets*. Aplicativos com esta característica possuem *Design* Responsivo. O *Design* Responsivo pode ser definido como um método para fazer com que todo o conteúdo existente em uma página seja otimizado, a depender-se do das dimensões de cada dispositivo exibindo esta página (SPURLOCK, 2013, p. 7). Foi utilizada neste projeto, uma ferramenta para auxiliar na construção dos *layouts* responsivos, denominada Bootstrap.

O Bootstrap é um produto de código aberto, criado por Mark Otto e Jacob Thornton, funcionários da rede social de *microblogging* Twitter. O grande motivo de sua construção surgiu da necessidade de padronizar o conjunto de ferramentas para desenvolvimento *frontend* na companhia. Na época de seu lançamento, em agosto de 2011, era um projeto inteiramente baseado em folhas de estilos CSS e com o passar dos anos, adicionou *plugins* JavaScript e ícones que acompanham formulários e botões (SPURLOCK, 2013, p. 1).

Foi possível, por meio de diversos estilos pré-definidos no Bootstrap, atribuí-los a vários elementos que compõem o aplicativo Geomapping. Além disso, para garantir a responsividade dos componentes de *layout*, foi adotado o sistema de *grids* do Bootstrap. Nele, uma tela pode ser subdividida em 12 colunas, cada uma

com 60 *pixels* de largura e 20 *pixels* de espaçamento entre uma coluna e a seguinte (SPURLOCK, 2013, p. 4).

A Figura 4 ilustra algumas configurações de *grids* possíveis. Todo componente definido dentro destas *grids*, se adapta em largura e altura, dependendo da área total do dispositivo usado.

Figura 4: Grid padrão do Bootstrap com 12 colunas (SPURLOCK, 2013, p. 3)

3.6 WebSQL

O WebSQL é uma API para armazenamento em banco de dados em páginas *web*, na qual as pesquisas pode ser feitas utilizando-se uma variante do SQL (*Structured Query Language*). Para que sejam feitas as consultas ao banco, é necessário utilizar o método *executeSql*. Este método recebe uma lista de parâmetros, sendo o primeiro uma *string* com a declaração em SQL da busca desejada, seguido de uma lista variável de argumentos adicionais que podem incluir funções de retorno para sucesso ou erro na consulta realizada. Quando a consulta retorna algum resultado, como em uma seleção aonde se deseja obter uma saída, deve haver um método para lidar com o evento de sucesso da pesquisa realizada, na qual é possível utilizar o conjunto de respostas retornado pela busca SQL. O método de sucesso é, no entanto opcional, não havendo necessidade de criá-lo quando se deseja apenas inserir dados, por exemplo.

3.7 XML

O acrônimo XML significa *eXtensible Markup Language* (linguagem de marcação extensível) e é um formato de texto que representa informação de maneira

estruturada. Sua maior aplicabilidade é o compartilhamento de informações estruturadas entre sistemas computacionais, geralmente por meio de acesso à rede. Tem sintaxe detalhada e cada etiqueta aberta deve possuir uma correspondente para fechá-la como no exemplo <descricao></descricao>.

O XML e o HTML têm semelhanças, sendo que o XML é mais restrito em sua sintaxe. Segue abaixo, uma lista de algumas diferenças do XML para o HTML(W3C):

- Todos os elementos podem ser fechados ou marcados como vazio.
- Elementos vazios podem ser fechados normalmente, assim como no exemplo <casa></casa>
- Em HTML, já existe previamente uma lista de nomes de elementos e seus atributos. No XML não há nomes de elementos estabelecidos a priori.

Este projeto se utiliza do formato XML para salvar dados coletados e armazenados em um banco de dados local, de maneira que seja possível acessá-lo e persisti-los no servidor de banco de dados na nuvem, a CLOUDAX, que será apresentada na seção a seguir.

3.8 CLOUDAX

A CLOUDAX é uma aplicação de computação em nuvem que permite acesso remoto a serviços e dados por meio da internet. A DATALAB contrata esta estrutura de nuvem para utilizar máquinas virtuais que provêm servidores de banco de dados e servidores *web*. Esse ambiente de computação em nuvem é capaz de receber dados de todas as aplicações da DATALAB, fazer buscas e enviar novos registros no banco de dados central, armazenando os dados dos locais georreferenciados. Os aplicativos da DATALAB se conectam a um *web service* presente na CLOUDAX visando trafegar os dados georreferenciados localmente por meio de arquivos XML, para que sejam serem salvos no banco de dados central da CLOUDAX.

3.9 Processo de Georreferenciamento

Para alcançar o objetivo de georreferenciar os pontos de interesse das várias aplicações da DATALAB há, além do Geomapping, outros dois entes importantes neste sistema: uma aplicação *web* de retaguarda e a estrutura de servidor em nuvem da DATALAB, a CLOUDAX.

A aplicação *web* de retaguarda, neste momento ainda em fase de desenvolvimento, será responsável por gerar as demandas dos pontos de localização que necessitam ser georreferenciados por meio de um cadastro de demandas. Uma demanda possui informações sobre a quantidade de pontos que precisam ser registrados, um endereço contendo a rua aonde será feita a captação de pontos, um indicador de qual aplicação da DATALAB está sendo georreferenciado, além de possuir informações sobre o IMEI (*International Mobile Equipment Identity*) do dispositivo móvel que será responsável pela geolocalização.

As demandas com estas características são enviadas por meio de um *Web Service* à CLOUDAX que, por sua vez, identifica o IMEI especificado na demanda e envia ao dispositivo correspondente um arquivo XML, com as exigências da demanda. Neste ponto, o Geomapping, de posse destes dados estruturados em formato XML, salva os campos em um banco de dados do WebSQL local do dispositivo, juntamente com os dados de latitude, longitude e endereço do ponto geográfico atual captado, além dos dados específicos a cada aplicação da DATALAB.

A Figura 5, desenvolvida pelo acadêmico Anderson Martins da Silva de ciência da computação e estagiário da DATALAB na área de análise de sistemas, apresenta um diagrama detalhando o passo a passo de como estes entes interagem entre si para garantir o georreferenciamento dos pontos geográficos de interesse dos vários aplicativos da DATALAB.

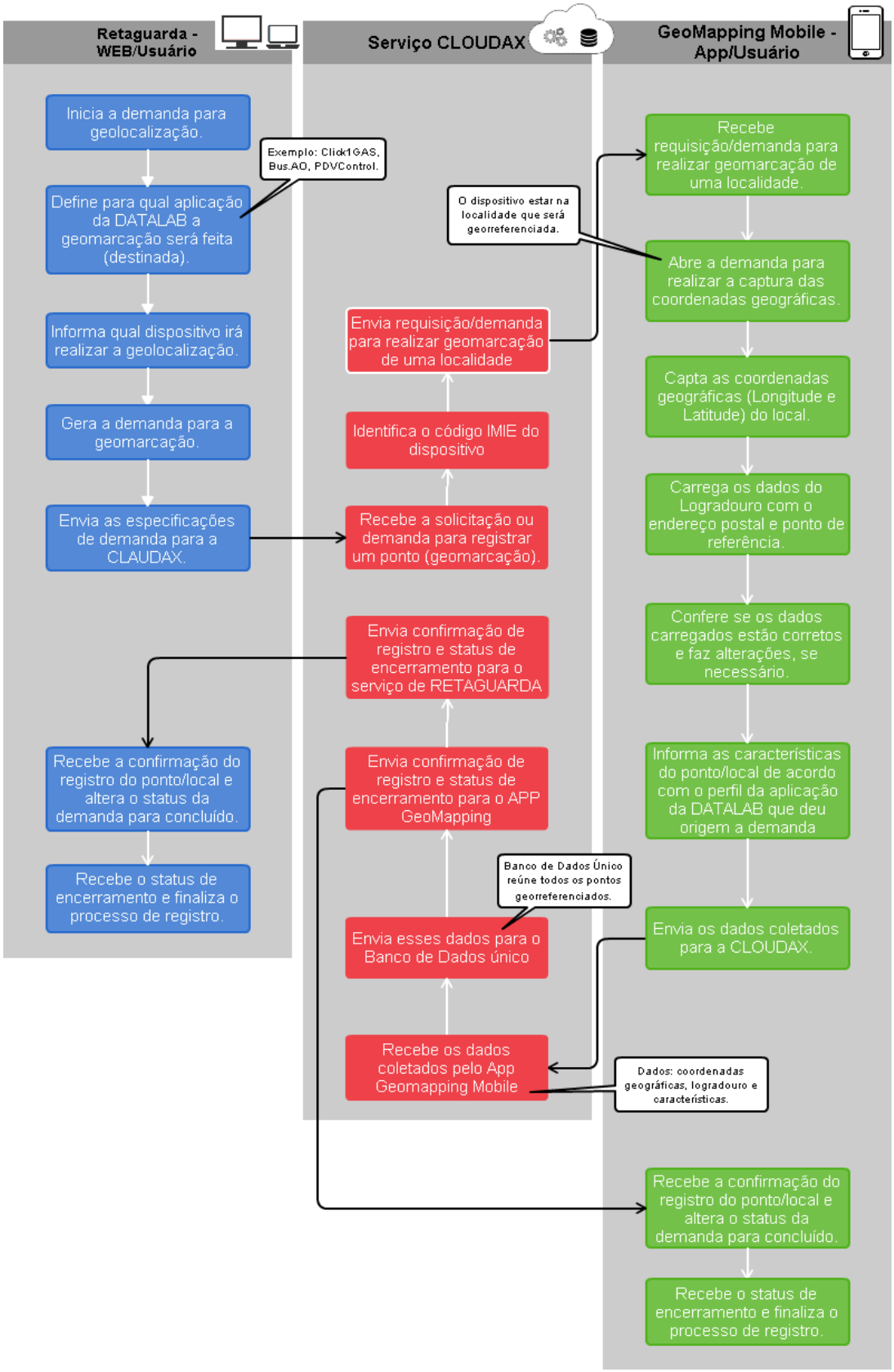

Figura 5: Diagrama de funcionamento dos pedidos de demanda e georreferenciamento no Geomapping

4. RESULTADOS

Neste capítulo serão apresentados os resultados do trabalho, desenvolvidos durante o estágio supervisionado na empresa DATALAB. Para realizá-los foram utilizados os conceitos de sistemas operacionais móveis, introduzidos durante a seção de revisão de literatura e, além disso, os conhecimentos e ferramentas apresentadas na seção de materiais e métodos sobre linguagens de programação, APIs e *frameworks* para ambiente *web*.

4.1 Geomapping

Muitas das aplicações móveis e *web* da DATALAB utilizam, em grande medida, dados de posição geográfica para seu funcionamento e, neste contexto, deseja-se que o Geomapping seja o aplicativo de uso interno único para efetuar o georreferenciamento dos pontos de interesse para todas as aplicações da empresa, com dados comuns a todos os pontos como latitude, longitude, rua, estado, país, dentre outros, além de dados específicos que concernem exclusivamente a cada aplicação que a DATALAB possui e utilizará o Geomapping.

Juntamente com a funcionalidade de georreferenciamento de pontos geográficos do aplicativo, há ainda uma área destinada as funcionalidades de caráter pessoal. Nesta área é possível escolher por visualizar a agenda telefônica dos contatos contidos no dispositivo, escrever e salvar anotações e visualizar em um mapa sua posição atual. Estas características do Geomapping serão apresentadas nas seções a seguir. A Figura 6 ilustra o painel esquerdo de funcionalidades e a tela inicial do Geomapping.

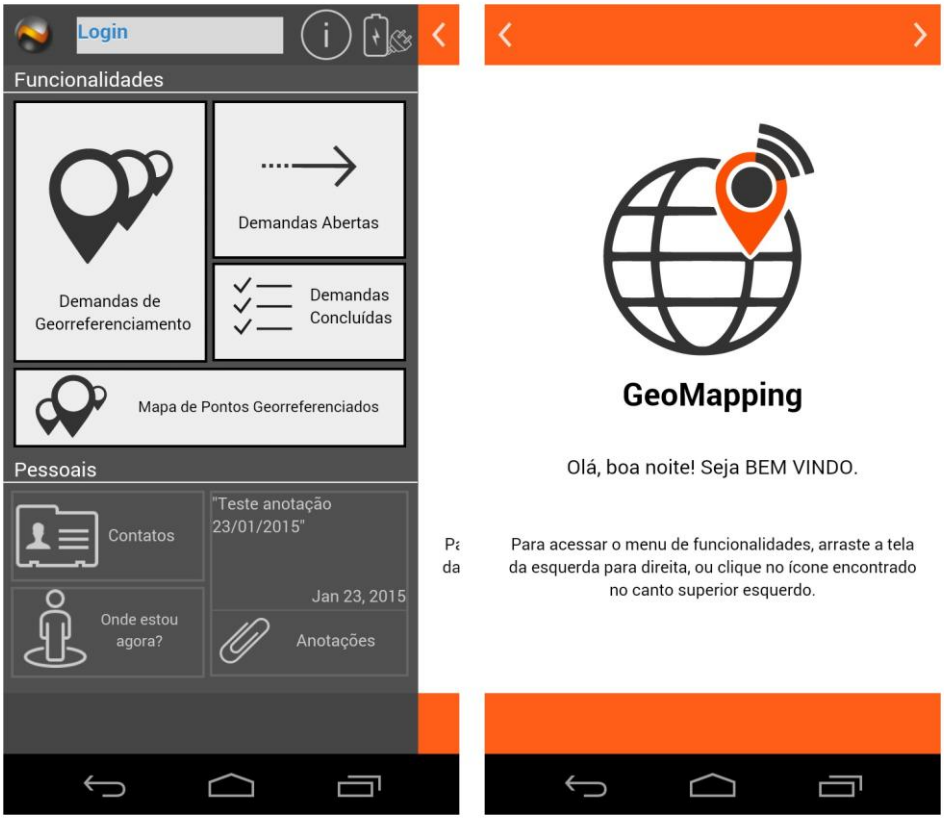

Figura 6: Painel esquerdo de funcionalidades e tela inicial do Geomapping

4.2 Agenda de contatos

A primeira opção disponível na seção de funcionalidades pessoais é a de contatos. Este módulo é o responsável por recuperar os contatos armazenados pelo sistema operacional e exibi-los de maneira tabular, com os campos de nome, sobrenome, telefone celular e residencial. Além da exibição dos contatos, o usuário tem a possibilidade de selecionar uma linha que representa um contato, a partir da tabela e escolher se deseja ligar para o telefone residencial ou celular deste contato.

O desenvolvimento deste módulo envolveu a utilização da API do Apache Cordova, que provê uma classe denominada *Contacts*, que dá acesso à base de dados nativa de contatos, fornecida pelos diversos sistemas operacionais móveis suportados pela API. Uma instância da classe *Contact* possui vários atributos e métodos referentes à base de contatos nativa, e neste projeto do Geomapping, acessou-se os campos *name* e *phoneNumbers*, ambos vetores que contém, sobre nomes, sobrenomes e os telefones relacionados a cada contato armazenado em

dispositivo. Foi necessário gerar dinamicamente uma *string* estruturada no formato HTML, para posteriormente adicioná-la à página principal do Geomapping, assim que o evento de clique é captado no botão “Contatos”, ilustrado na Figura 7.

Os dados obtidos pelo banco nativo do dispositivo retornam os contatos na ordem em que foram inseridos e, além disso, a tabela gerada dinamicamente tinha sua exibição, inicialmente, de maneira integral na tela e necessitando que o usuário utilizasse a barra de rolagem para procurar o contato desejado. Portanto, foi necessário utilizar um *plugin* escrito em jQuery, denominado DataTables, que recebe uma tabela e a apresenta com paginação e ordena os elementos contidos nela.

Adicionalmente, criou-se uma função destinada a captar os cliques em cada uma das linhas da tabela, para que o usuário pudesse escolher se deseja realizar uma chamada para o telefone residencial ou celular do contato selecionado. No Apêndice I, há o trecho de código fonte em linguagem Javascript criado para ler os contatos armazenados localmente no dispositivo e a adição destes dados em uma tabela HTML dinâmica.

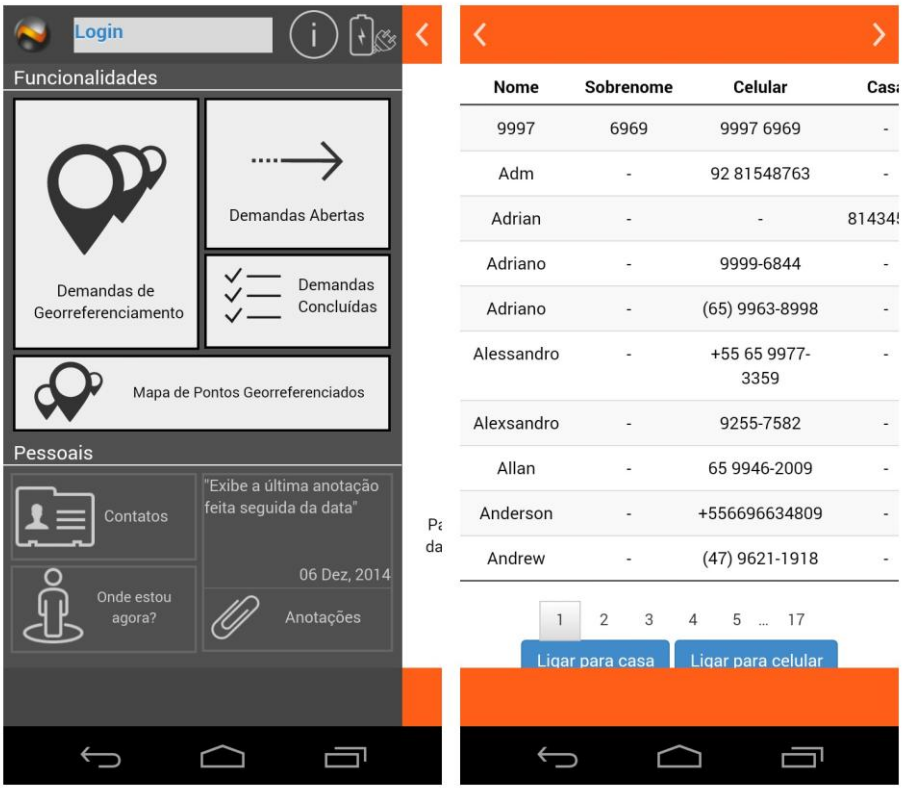

Figura 7: Tela de agenda de contatos do Geomapping

4.3 Onde estou agora?

A segunda funcionalidade da seção de funcionalidades pessoais é a “Onde Estou Agora?”. Neste módulo do aplicativo, é oferecido ao usuário a possibilidade de verificar a sua posição geográfica atual em um mapa, com informações obtidas do GPS do dispositivo móvel.

O mapa carregado no aplicativo é proveniente da API do Google Maps, serviço de geolocalização da empresa, adicionando-se uma referência à API na seção *head* da página HTML. Para abrir o mapa na página, criou-se o elemento HTML denominado *div*, que representa o container aonde o mapa é inserido. Essa inserção é realizada por meio de uma chamada a uma nova instância de objeto *google.maps.Map* contendo dois parâmetros, um fazendo referência a *div* container definida no HTML e um objeto denominado *mapOptions*, que possui atributos configuração a respeito do mapa como zoom, desabilitação de indicadores carregados de maneira padrão juntamente com o mapa, etc. Depois, é necessário acrescer um marcador sobre a posição do usuário, captada pelo GPS do dispositivo, além de definir o centro do mapa com a posição obtida.

O ato de clicar no marcador sobre o mapa faz abrir em um painel à direita do aplicativo contendo informações de endereço, relativas à latitude e longitude captadas, em um processo definido na documentação da API do Google Maps como geolocalização reversa. Para realizar esta geolocalização, assim como citado na seção 2.5 deste relatório, é preciso instanciar um objeto do tipo *Geocoder*, proveniente da API, e acessar um método denomiando *geocode*, passando como parâmetro a posição atual, além de uma função anônima com os parâmetros *results* e *status*. Depois de verificado se a posição utilizada gerou resultados, são lidos os campos de endereço obtidos por meio desta posição: rua, número, CEP, complemento, bairro, estado e país. Há que se frisar, porém, que os resultados de endereço obtidos neste processo de geolocalização reversa não são exatos, de forma que as informações sobre o ponto geográfico são aproximações do seu real endereço. A Figura 8 ilustra o painel da direita, aonde são abertos os dados de endereço captados e o Apêndice II apresenta o trecho do código fonte construído para obter o carregamento do mapa com a posição atual, adicionando-se um marcador neste local. Este apêndice demonstra ainda, como é feita a chamada do método *geocode*, aonde são carregadas as informações sobre a

posição atual e escritas nos seus respectivos campos da página HTML do painel à direita.

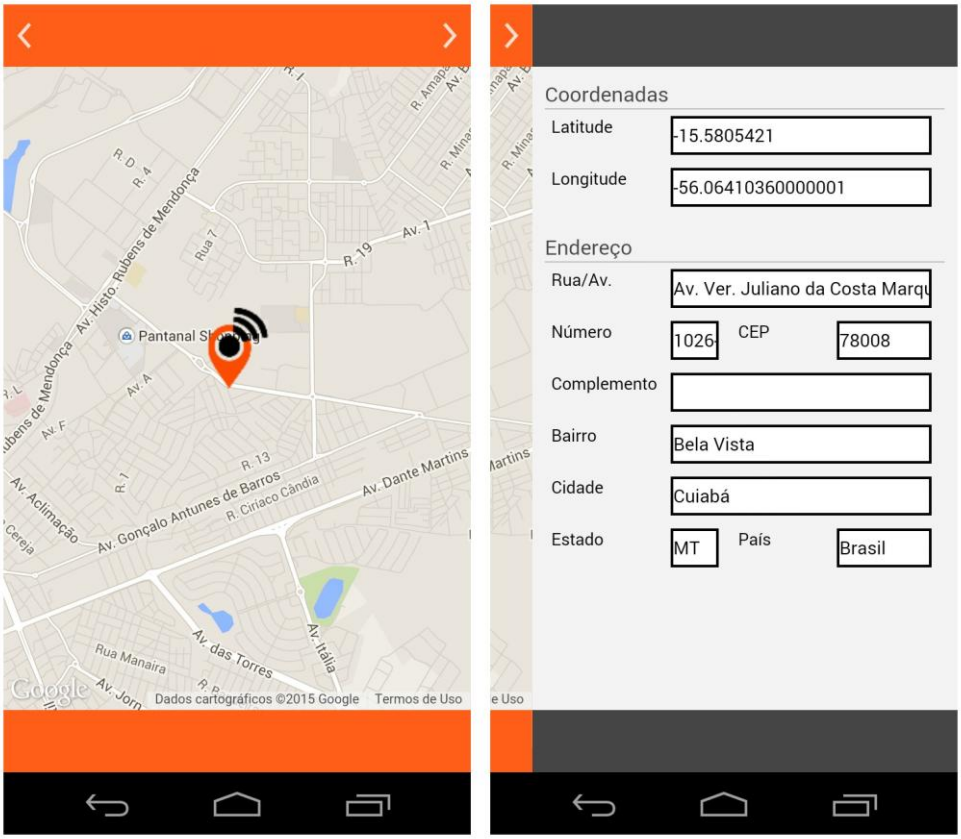

Figura 8: Tela com mapa do ponto atual e painel direito com dados de coordenadas e endereço

4.4 Anotações

A terceira funcionalidade da seção pessoal do aplicativo Geomapping é a responsável por oferecer aos usuários a capacidade de escrever anotações e salvá-las localmente no aplicativo. Para tanto, foram criados um campo para inserção das anotações ao lado de um botão que as salva no banco de dados.

Utilizou-se como banco de dados para armazenar as anotações inseridas pelo usuário o WebSQL, componente padrão do Apache Cordova. O banco de dados para persistência destes dados consiste apenas de uma tabela “Anotações” contendo seu identificador não nulo e auto incrementado, um literal para armazenar o conteúdo das anotações, além de um campo pra salvar a data e a hora em que a anotação foi inserida no banco.

O usuário pode ver a lista de anotações feitas no aplicativo, a partir do botão “Listar” e abaixo do botão “Guardar”, assim como ilustrado na Figura 9.

Assim como definido em projeto, a última anotação salva pelo usuário do aplicativo é exibida acima do botão de inserção de anotações, juntamente com o dia em que esta anotação foi salva.

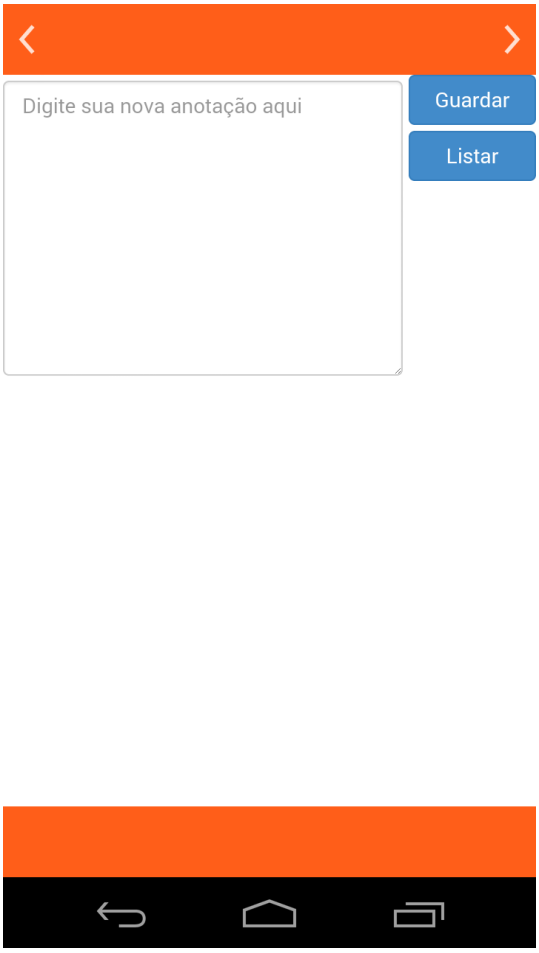

Figura 9: Tela de inserção das anotações do usuário

4.5 Demandas de Georreferenciamento

A seção “Funcionalidades” no painel esquerdo do Geomapping, possui todas as opções referentes às demandas de geolocalização. Esta área concentra o ciclo a ser cumprido de recebimento das demandas de georreferenciamento de pontos e a execução do georreferenciamento destas demandas.

Inicialmente, o aplicativo Geomapping cria uma pasta denominada “Downloads” na memória interna do dispositivo, destinada receber por meio da CLOUDAX um arquivo em formato XML contendo instruções sobre como devem

ser executados o lote de pontos a serem georreferenciados, indica sobre qual aplicativo estes pontos se referem, entre outras informações. O arquivo XML que deve ser recebido no dispositivo Geomapping é similar ao representado a seguir:

```
<DEMANDA>
    <TITULO>
        Cadastrar pontos de ônibus da Av. Fernando Correa
    </TITULO>
    <LOCAL>
        Av. Fernando Correa
    </LOCAL>
    <QUANTIDADE>
        5
    </QUANTIDADE>
    <APLICACAO>
        2
    </APLICACAO>
    <ORIENTACOES>
        Deve-se cadastrar os pontos de embarque e desembarque
do transporte coletivo na Av. Fernando Correa de ambas as pistas. A
coleta das coordenadas deve ser realizada quando o dispositivo
estivaer no local exato do ponto de ônibus.
    </ORIENTACOES>
    <IMEI>
        359323060296568
    </IMEI>
    <DATA_INICIO>
        19-01-2015
    </DATA_INICIO>
    <DATA_FINAL>
        30-01-2015
    </DATA_FINAL>
</DEMANDA>
```

Os dados obtidos do XML são decodificados e a partir disto, essas informações são utilizadas em diversos momentos no aplicativo. No botão “Demandas de Georreferenciamento”, haverá um indicador numérico equivalente ao atributo <quantidade> do arquivo XML recebido, que será visualizado após o usuário

do Geomapping tiver feito *login* no sistema – funcionalidades ainda não implementadas – demonstrando quantas demandas de georreferenciamento de pontos serão necessárias serem efetivadas.

Ao clicar no botão, um mapa com a posição atual do usuário é aberto na tela principal do Geomapping, da mesma maneira que ocorre na funcionalidade "Onde Estou Agora?" e ao clicar no marcador, o painel direito apresenta os mesmos dados obtidos pelo Geocoder na função "Onde Estou Agora?", a diferença, porém é que ao clicar no marcador do mapa, adiciona-se logo abaixo dos dados de coordenada e endereço, um perfil do aplicativo exigido na demanda é aberto, perfil esse condicionado ao valor recebido no atributo <aplicacao>, aonde no exemplo, o número 2 indica a aplicação Bus.AO da DATALAB, citada na introdução deste trabalho. A Figura 10 ilustra o painel direito com o perfil do aplicativo Bus.AO.

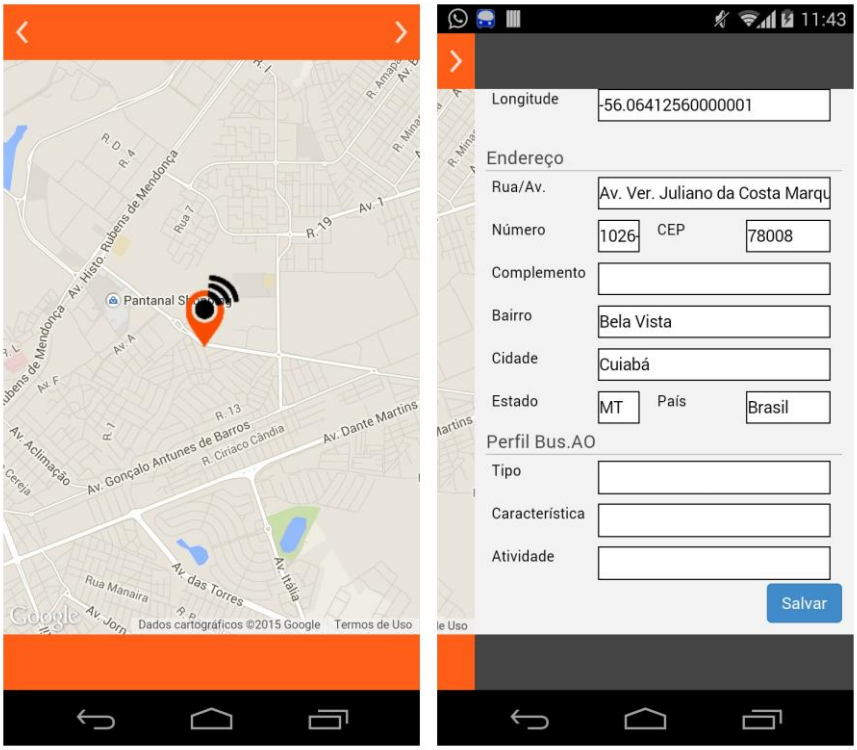

Figura 10: Painel direito da funcionalidade "Demandas de Georreferenciamento"

Estando o usuário do Geomapping no ponto aonde deve georreferenciar, e tenha ele aberto o painel direito da funcionalidade "Demandas de Georreferenciamento", ele tem a possibilidade de alterar os campos de endereço, caso haja algum erro nos dados obtidos automaticamente pelo Geocoder, como também poderá adicionar os dados específicos ao perfil do ponto, dependendo da

aplicação escolhida na demanda. Somente os campos de latitude e longitude é que se mantém não editáveis.

Ao clicar no botão salvar, depois de completo todo cadastro do ponto geográfico, os dados devem ser salvos no banco de dados WebSQL local, seguindo a modelagem física da Figura 11, igualmente desenvolvida por Anderson Martins da Silva e definida em projeto. Há nela a representação de dois possíveis tipos de perfis de aplicações da DATALAB, o Click1Gás e o Bus.AO. O Apêndice III possui o trecho do código fonte responsável por criar as tabelas representadas pelo modelo da Figura 11, carregar um mapa indicando a posição atual, escrever os dados coletados pelo Geocoder no painel à direita e por fim, contém as funções responsáveis por inserir novos endereços, coordenadas e perfis adicionados pelo usuário e salvos no banco de dados local.

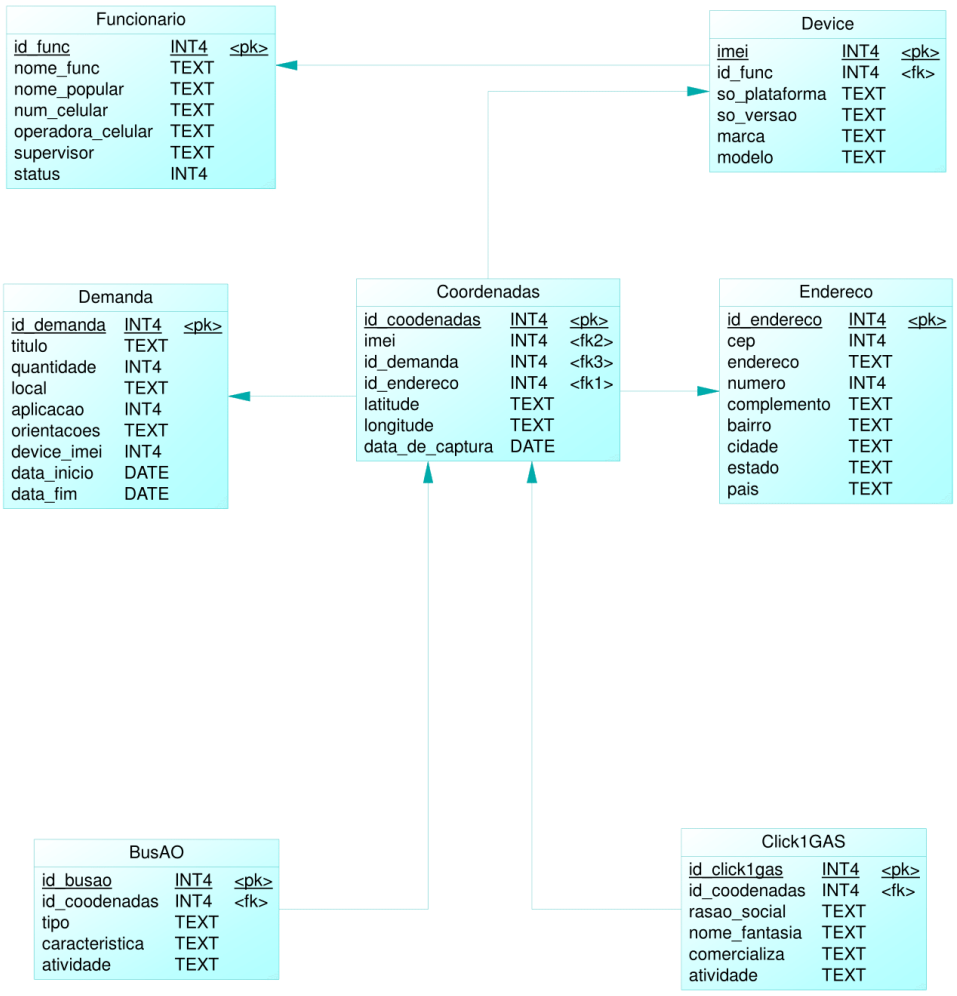

Figura 11: Modelagem física dos dados de georreferenciamento de pontos

O próximo passo é a geração de um novo XML, com estrutura ainda indefinida no projeto, que contenha todos os pontos georreferenciados obtidos do banco de dados local e salvo na pasta “Downloads”. Este XML será coletado por um aplicativo que já existe na DATALAB, o DtlMobSync, que é o responsável por ler o XML dos pontos georreferenciados que foram gerados, e sincronizar esses dados com o servidor de banco de dados do serviço CLOUDAX.

5. DIFICULDADES ENCONTRADAS

O ambiente de desenvolvimento Intel XDK é ainda uma ferramenta nova e pouco estabelecida. Este fato gerou diversas dificuldades na execução deste projeto.

Parte fundamental para o desenvolvimento de qualquer aplicativo, os testes de comportamento de *layout* e funcionalidades podem ser feitos no Intel XDK por diversas vias. São disponibilizadas no ambiente, três opções que possibilitam a visualização prévia dos aplicativos desenvolvidos, apresentando gradações de fidelidade em comparação com a aplicação em sua versão final, em formato de executável. Foram encontradas inconsistências de comportamento quando testadas versões idênticas no *debugger* contido no Intel XDK e seu executável correspondente. Foi constatado que estas inconsistências ainda ocorrem na plataforma e que próximas versões devem corrigi-las por intermédio de consultas ao fórum oficial do ambiente.

O Intel XDK apresentou ainda alguns problemas pontuais quando executado como fechamentos involuntárias do ambiente e também do seu *debugger* interno nos testes que *layouts* de maior complexidade e com funcionalidades mais exigentes com recursos do sistema. Estes percalços foram superados em parte com atualizações na versão do XDK, além da substituição do *debugger* nativo do ambiente utilizado para o *Google Chrome Developer Tools* (CDT).

Os protótipos do módulo Geomapping foram redefinidos por algumas vezes durante o período de estágio, tornando-se necessário fazer readequações de *layout* também de funcionalidades deste módulo, ocasionando em última instância em atraso na execução do projeto. Adicionalmente, a pouca experiência do estagiário em programação também contribuiu com o atraso. Para superar essa dificuldade foram necessárias diversas pesquisas de conteúdos relacionados às linguagens, *frameworks*

e bibliotecas utilizadas no projeto, além do auxílio dos colegas de trabalho e superiores da empresa.

6. CONCLUSÕES

A empresa DATALAB possui diversos aplicativos que utilizam do conceito de georreferenciamento, variando os perfis dos pontos geográficos coletados e cadastrados, dependendo do aplicativo a ser utilizado. O processo de georreferenciamento é atualmente feito localmente em cada um dos aplicativos que usam os serviços de geolocalização, juntamente com a infraestrutura de nuvem da DATALAB.

Porém, houve a necessidade de construir um aplicativo que funcione como georreferenciador padrão dos pontos geográficos de interesse da empresa para todos os aplicativos e a ele denominou-se Geomapping. Este aplicativo deve servir futuramente ao projeto do Geomapping no cadastramento das revendedoras e distribuidoras de água e gás que desejam comprar o sistema.

O objetivo deste trabalho foi utilizar os conceitos de programação com linguagens de programação para ambiente *web* como Javascript, HTML e a linguagem de folha de estilo CSS para construir aplicativos multiplataforma a partir do novo ambiente de desenvolvimento lançado pela Intel, o XDK.

Com é possível verificar nos resultados, este trabalho se limitou a construção do aplicativo Geomapping, que dá suporte a outros aplicativos da empresa, ainda de forma inconclusa. Portanto, os próximos passos deste projeto são a conclusão do aplicativo Geomapping, e a continuação da construção de aplicativos utilizando as tecnologias para desenvolvimento multiplataforma com linguagens para ambiente *web* abordadas neste relatório de estágio.

7. REFERÊNCIAS BIBLIOGRÁFICAS

ABLESON, Frank W.;ROBI,Sen; KING, Chris; 2012. *Android in Action*. 3ª Edição. Shelter Islands : Manning Publications.

GARGENTA, Marko.; 2011. *Learning Android*. 1ª Edição. Sebastopol: O'Reilly Media, Inc.

IBM. Native, web or hybrid mobile-app development. Disponível por ftp em ftp://public.dhe.ibm.com/software/pdf/mobile-enterprise/WSW14182USEN.pdf (acessado em 25 de janeiro de 2015).

LECHETA, Ricardo R.; 2009. *GoogleAndroid*. 3ª Edição. São Paulo : EditoraNovatec Ltda.

MACRAE, Callum; 2013. *Learning from jQuery*. 1ª Edição. Sebastopol: O'Reilly Media, Inc.

MYER, Thomas.; 2012. *BeginingPhonegap*. 1ª Edição. Indianapolis: John Wiley & Sons, Inc.

SILVA, Maurício Samy.; 2012. *CSS3:Desenvolva Aplicações Web Profissionais Com Uso dos Poderosos Recursos de Estilização das CSS3*. 1ª Edição. São Paulo: Editora Novatec Ltda.

SILVA, Maurício Samy.; 2010. *JavaScript: Guia do Programador*. 1ª Edição. São Paulo: Editora Novatec Ltda.

SPURLOCK, Jake; 2013. *Bootstrap*. 1ª Edição. Sebastopol: O'Reilly Media, Inc.

SVENNERBERG, Gabriel; 2010. *Beginning Google Maps API 3*. 1ª Edição. Nova York: SpringerScience+Business Media, LLC.

TANENBAUM, Andrew S.; 2009. *Sistemas Operacionais Modernos*. 3ª Edição. São Paulo : Editora Pearson Prentice Hall.

APÊNDICE I: TRECHO DO CÓDIGO FONTE JAVASCRIPT PARA FUNCIONALIDADE “CONTATOS”

```
function recuperar(){
    var fields = ["phoneNumbers", "name"];
    var options = new ContactFindOptions();
    options.filter="";
    options.multiple=true;
    navigator.contacts.find(fields, onSuccess, onError,
options);
}

function onSuccess(contacts) {

    var temCelular;
    var temResidencia;
    var temEmail;

    var html = "<table id=\"tabela\" data-role=\"table\"
class=\"table table-condensed table-striped\" style=\"text-
align:center; font-size:1.1em;\"><thead><tr><th style=\"text-
align:center;\">Nome</th><th style=\"text-
align:center;\">Sobrenome</th><th style=\"text-
align:center;\">Celular</th><th style=\"text-
align:center;\">Casa</th><tbody>";

    var cont = 1;
    for (var i = 0; i < contacts.length; i++)
    {
          var nome = contacts[i].name.givenName;
          var sobrenome = contacts[i].name.familyName;

          if ( contacts[i].phoneNumbers !== null ){

              if (nome !== undefined || sobrenome !==
undefined)
              {
                  html += "<tr>";
              }

              if (nome === undefined && (sobrenome !==
undefined || sobrenome === ""))
              {
                  html += "<td> - </td><td>" + sobrenome +
"</td>";
              }
              else if (sobrenome === undefined && (nome !==
undefined || nome === ""))
              {
                   html += "<td>" + nome + "</td><td> -
</td>";
              }
```

```
                else if (nome === undefined && sobrenome ===
undefined)
                {
                    continue;
                }
                else
                {
                     html += "<td>" + nome + "</td><td>" +
sobrenome + "</td>";
                }

                /*
                Vasculha o array 'phoneNumbers' e procura pelo
primeiro cadastrado
                com o campo type == 'mobile'. Se encontrar,
adiciona à tabela, senão
                adiciona o valor ' - ' à celula da tabela.
                */
                var j = 0;
                temCelular = false;
                while  ( j < contacts[i].phoneNumbers.length &&
!temCelular )
                {

                    if (String(contacts[i].phoneNumbers[j].type)
== "mobile")
                    {
                          html += "<td>" +
contacts[i].phoneNumbers[j].value + "</td>";
                          temCelular = true;
                    }

                    j++;

                }

                if (!temCelular)
                {
                    html += "<td> - </td>";
                }

                /*
                Vasculha o array 'emails' e procura pelo
primeiro cadastrado
                com o campo type == 'home'. Se encontrar,
adiciona à tabela, senão
                adiciona o valor ' - ' à celula da tabela.
                */

                var k = 0;
                temResidencia = false;
                while( k < contacts[i].phoneNumbers.length &&
!temResidencia )
```

```
                {

                    if (
String(contacts[i].phoneNumbers[k].type) == "home" ){
                        html += "<td>" +
contacts[i].phoneNumbers[k].value + "</td>";
                        temResidencia = true;
                    }

                    k++;

                }

                if ( !temResidencia )
                {
                    html += "<td> - </td>";
                }

                html += "</tr>";
                cont = cont + 1;
            }

        }
        html += "</tbody></table>";
```

APÊNDICE II: TRECHO DE CÓDIGO FONTE JAVASCRIPT COM USO DA API DO GOOGLE MAPS PARA A FUNCIONALIDADE “ONDE ESTOU AGORA?”

```
    var marker;
    var map;
    var pos;
    var geocoder = new google.maps.Geocoder();
    function initializeOndeEstou() {

        var numero = "",
            rua = "",
            cep = "",
            bairro = "",
            cidade = "",
            estado = "",
            pais = "", i;

        var mapOptions = {
            zoom: 14,
            disableDefaultUI: true
        };

        //Inicializar mapa na div no content da tela principal
        map = new
google.maps.Map(document.getElementById('divPageBody'),mapOpti
ons);

        if(navigator.geolocation) {

navigator.geolocation.getCurrentPosition(function(position) {
                //Criar tipo LatLang com a latitude e
longitude atuais
                pos = new
google.maps.LatLng(position.coords.latitude,position.coords.lo
ngitude);

                map.setCenter(pos);
                var image = "img/marcador64px.png";
                //Criar marcador na posição atual
                marker = new google.maps.Marker({
                    position: pos,
                    map: map,
                    icon: image
                });


                google.maps.event.addListener(marker, 'click',
function() {

                    $("#divDireitaBody").html("");
```

```
$("#divDireitaBody").load("cadastroOndeEstou.html");
                    iniciaGeocoder(this);
                    $("#pnlDireita").panel("open");

                });

                },function(){
                    handleNoGeolocation("Não encontrou
coordenadas");
            });
        }else{
            handleNoGeolocation("html5 - geolocation não
suportado");
        }

        google.maps.event.addDomListener(window,'load',
initializeOndeEstou);

     }

    function iniciaGeocoder(marker){

        var pos = marker.getPosition();

        geocoder.geocode({'latLng': pos}, function(results,
status) {
            if (status == google.maps.GeocoderStatus.OK)
            {
              if (results[0])
              {

$("#txtDisplayLatitude").val(marker.getPosition().lat());

$("#txtDisplayLongitude").val(marker.getPosition().lng());

                for (i = 0; i <
results[0].address_components.length; i++){

                    if
(results[0].address_components[i].types[0] ==
"street_number"){
                        numero =
results[0].address_components[i].short_name;
                        $("#txtDisplayNumero").val(numero);
                    }
                    else if
(results[0].address_components[i].types[0] == "route"){
                        rua =
results[0].address_components[i].short_name;
                        $("#txtDisplayRua").val(rua);
                    }
```

```
                    else if
(results[0].address_components[i].types[0] == "postal_code"
                                ||
results[0].address_components[i].types[0] ==
"postal_code_prefix"){
                            cep =
results[0].address_components[i].long_name;
                            $("#txtDisplayCep").val(cep);
                    }
                    else if
(results[0].address_components[i].types[0] == "neighborhood"){
                            bairro =
results[0].address_components[i].long_name;
                            $("#txtDisplayBairro").val(bairro);
                    }
                    else if
(results[0].address_components[i].types[0] ==
"administrative_area_level_2"){
                            cidade =
results[0].address_components[i].long_name;
                            $("#txtDisplayCidade").val(cidade);
                    }
                    else if
(results[0].address_components[i].types[0] ==
"administrative_area_level_1"){
                            estado =
results[0].address_components[i].short_name;
                            $("#txtDisplayEstado").val(estado);
                    }
                    else if
(results[0].address_components[i].types[0] == "country"){
                            pais =
results[0].address_components[i].long_name;
                            $("#txtDisplayPais").val(pais);
                    }
                }

              }
              else
              {
                alert('Nenhum resultado encontrado!');
              }
            }
            else
            {
              alert('Falha no Geocoder devido a: ' + status);
            }
            configuraAlturaOndeEstou();
        });
    }

    function handleNoGeolocation(erro){
        console.log("Erro ao carregar o mapa: "+erro);
```

```
}

function onfail (){
 alert("erro geolocation ");
}
```

APÊNDICE III: TRECHO DO CÓDIGO FONTE JAVASCRIPT PARA A FUNCIONALIDADE "DEMANDAS DE GEORREFERENCIAMENTO"

```
//-------------------------------------------------------------
---------------------------------
//                  ABRIR CONEXAO COM O BANCO E CRIAR A TABELA
USUÁRIO
//-------------------------------------------------------------
---------------------------------

document.addEventListener("deviceready", openDatabaseGeo,
false);

function openDatabaseGeo(){
    var db = window.openDatabase("georreferencia", "1.0", "my
db", 2*1024*1024);
    db.transaction(criaTabelasGeo, successCB, erroDb);
}

function criaTabelasGeo(t) {

    //CRIA TABELAS COORDENADA, DEMANDA, DEVICE, ENDERECO,
PERFILBUSAO, PERFILCLICK1GAS, COLABORADOR
    //-------------------------------------------------------
---------------------------------------

    t.executeSql('CREATE TABLE IF NOT EXISTS
COORDENADA(idCoordenada INTEGER NOT NULL PRIMARY KEY
AUTOINCREMENT, imei TEXT NOT NULL, idDemanda INTEGER NOT NULL,
idEndereco INTEGER NOT NULL, latitude TEXT NOT NULL, longitude
TEXT NOT NULL, dataCaptura TEXT NOT NULL, FOREIGN KEY(imei)
REFERENCES DEVICE(imei), FOREIGN KEY(idDemanda) REFERENCES
DEMANDA(idDemanda), FOREIGN KEY(idEndereco) REFERENCES
ENDERECO(idEndereco));');

    t.executeSql('CREATE TABLE IF NOT EXISTS DEMANDA(idDemanda
INTEGER NOT NULL PRIMARY KEY, titulo TEXT NOT NULL, quantidade
INTEGER NOT NULL, local TEXT NOT NULL, aplicacao INTEGER NOT
NULL, orientacoes TEXT NOT NULL, deviceImei TEXT NOT NULL,
dataInicio TEXT NOT NULL, dataFim TEXT NOT NULL);');

    t.executeSql('CREATE TABLE IF NOT EXISTS DEVICE(imei TEXT
NOT NULL PRIMARY KEY, idColaborador INTEGER NOT NULL,
SOPlataforma TEXT NOT NULL, SOVersao TEXT NOT NULL, marca TEXT
NOT NULL, modelo TEXT NOT NULL, FOREIGN KEY(idColaborador)
REFERENCES COLABORADOR(idColaborador));');
```

```
    t.executeSql('CREATE TABLE IF NOT EXISTS
ENDERECO(idEndereco INTEGER NOT NULL PRIMARY KEY
AUTOINCREMENT, cep TEXT NOT NULL, rua TEXT NOT NULL, numero
TEXT NOT NULL, complemento TEXT NOT NULL, bairro TEXT NOT
NULL, cidade TEXT NOT NULL, estado TEXT NOT NULL, pais TEXT
NOT NULL);');

    t.executeSql('CREATE TABLE IF NOT EXISTS
PERFILBUSAO(idPerfilBusao INTEGER NOT NULL PRIMARY KEY
AUTOINCREMENT, idCoordenada INTEGER NOT NULL, tipo TEXT NOT
NULL, caracteristica TEXT NOT NULL, atividade TEXT NOT NULL,
FOREIGN KEY (idCoordenada) REFERENCES
COORDENADA(idCoordenada));');

    t.executeSql('CREATE TABLE IF NOT EXISTS
PERFILCLICK1GAS(idPerfilClick1GAS INTEGER NOT NULL PRIMARY KEY
AUTOINCREMENT, idCoordenada INTEGER NOT NULL, razaoSocial TEXT
NOT NULL, nomeFantasia TEXT NOT NULL, comercializa TEXT NOT
NULL, atividade TEXT NOT NULL, FOREIGN KEY (idCoordenada)
REFERENCES COORDENADA(idCoordenada));');

    t.executeSql('CREATE TABLE IF NOT EXISTS
COLABORADOR(idColaborador INTEGER NOT NULL PRIMARY KEY
AUTOINCREMENT, nomeColaborador TEXT NOT NULL, nomePopular TEXT
NOT NULL, nomeCelular TEXT NOT NULL, operadoraCelular TEXT NOT
NULL, supervisor TEXT NOT NULL, status INTEGER NOT NULL);');

    //PREENCHER 2 DAS TABELAS COM APENAS UM REGISTRO,
COLABORADOR E DEVICE. ELAS SÃO INICIALIZADAS NA ABERTURA DO
APLICATIVO
    //-------------------------------------------------------
-------------------------------------------------------------
----

    //INSERIR COLABRADOR, CONTEÚDO VINDO DO SERVIDOR
    //t.executeSql('INSERT INTO COLABORADOR(nomeColaborador,
nomePopular, nomeCelular, operadoraCelular, supervisor,
status) VALUES (\'Erik W. Chagas Silva\', \'erik\',
\'nomeCelular\', \'Claro\', \'Noely Moraes\', \'Ativo\');');

    //LER DADOS DO DISPOSITIVO
    var uuid = window.device.uuid;
    var plataforma = window.device.platform;
    var versao = window.device.version;
    var modelo = window.device.model;

    //t.executeSql("INSERT INTO DEVICE (imei,idColaborador,
SOPlataforma, SOVersao, marca, modelo) VALUES ('" + uuid + "',
1, '" + plataforma + "', '" + versao +  "', " + "'marca', '" +
modelo + "');");

}
```

```
//-------------------------------------------------------------
-----------------------------------------------
//      FUNÇÕES PARA CRIAR DIRETORIOS PADRÕES SE NECESSÁRIO E
LER ARQUIVO demanda.xml
//-------------------------------------------------------------
-----------------------------------------------

document.addEventListener("deviceready", criarNovosDiretorios,
false);

function criarNovosDiretorios(){

window.requestFileSystem(LocalFileSystem.PERSISTENT,0,gotFS,fa
il);
}

function gotFS(fileSystem){

fileSystem.root.getDirectory("DATALAB/Geomapping/downloads",{c
reate: true, exclusive: false}, function(){}, fail);

fileSystem.root.getFile("DATALAB/Geomapping/downloads/demanda.
xml",{create:false,exclusive:false},gotFileEntry,fail1);
}

function gotFileEntry(fileEntry){
    fileEntry.file(gotFile,fail);
}

function gotFile(file){
    readAsText(file);
}

function readAsText(file) {
    var reader = new FileReader();
    reader.onloadend = function(evt) {

        //Ler string contendo o XML e coletar seus campos
        /* http://api.jquery.com/jQuery.parseXML/ */
        var xml = evt.target.result;
        xmlDoc = $.parseXML(xml),
        $xml = $( xmlDoc ),
        $idDemanda = $xml.find("ID_DEMANDA");
        $titulo = $xml.find("TITULO");
        $quantidade = $xml.find("QUANTIDADE");
        $local = $xml.find("LOCAL");
        $aplicacao = $xml.find("APLICACAO");
        $orientacoes = $xml.find("ORIENTACOES");
        $deviceImei = $xml.find("DEVICE_IMEI");
        $dataInicio = $xml.find("DATA_INICIO");
        $dataFinal = $xml.find("DATA_FINAL");
```

```
        //Criar objeto com todos os campos coletados do XML
        var xmlObj = {
            idDemanda : Number($idDemanda.text()),
            titulo : $titulo.text(),
            quantidade : Number($quantidade.text()),
            local : $local.text(),
            aplicacao : Number($aplicacao.text()),
            orientacoes : $orientacoes.text(),
            deviceImei : $deviceImei.text(),
            dataInicio : $dataInicio.text(),
            dataFinal : $dataFinal.text()
        }

        //ASSIM QUE O XML TENHA SIDO LIDO, CHAMAR FUNÇÃO PARA
INSERIR
        openDbInserirDemanda(xmlObj);

    };
    reader.readAsText(file);
}


//-------------------------------------------------------------
-----------------------------------------------
//                              FUNÇÕES PARA INSERIR A TABELA
DEMANDA
//-------------------------------------------------------------
-----------------------------------------------

function openDbInserirDemanda(xmlObj){
    var db = window.openDatabase("georreferencia", "1.0", "my
db", 2*1024*1024);

    db.transaction(function(t){ inserirDemanda(t, xmlObj) },
successCB, erroDbGeo);
}

function inserirDemanda(t, xmlObj){

    t.executeSql("INSERT INTO DEMANDA VALUES (" +
xmlObj["idDemanda"] + ", '" + xmlObj["titulo"] + "', " +
xmlObj["quantidade"] + ", '" + xmlObj["local"] + "' , " +
xmlObj["aplicacao"] + ", '" + xmlObj["orientacoes"] + "' , '"
+ xmlObj["deviceImei"] + "' , '" + xmlObj["dataInicio"] + "' ,
'" + xmlObj["dataFinal"] + "' );");


}


//-------------------------------------------------------------
-----------------------------------------------
```

```
//                                FUNÇÕES PARA CONSULTAR A
TABELA DEMANDA
//-------------------------------------------------------------
-----------------------------------------------

function openDatabaseTipoDemanda(){
    var db = window.openDatabase("georreferencia", "1.0", "my
db", 2*1024*1024);
    db.transaction(selecaoTipoDemanda, successCB, erroDb);

}

function selecaoTipoDemanda(t) {
    t.executeSql('SELECT * FROM DEMANDA', [],
selecaoSucessoTipoDemanda, erroDb);
}

function selecaoSucessoTipoDemanda(t, data) {

    var codigoPerfil = data.rows.item(0).aplicacao;
    return codigoPerfil;
}


//-------------------------------------------------------------
-----------------------------------------------
//            FUNÇÕES PARA SALVAR UM PONTO, NAS TABELAS
ENDERECO, COORDENADA E CLICK1GAS
//-------------------------------------------------------------
-----------------------------------------------

function openInserirEndereco(inserirCoordenadas){
    var db = window.openDatabase("georreferencia", "1.0", "my
db", 2*1024*1024);
    db.transaction(inserirEndereco, successCB, erroDb);
    inserirCoordenadas();
}

function inserirEndereco(tx) {

    // INSERIR TABELA "ENDEREÇO"
    var rua = $("#txtDisplayRua").val();
    var numero = $("#txtDisplayNumero").val();
    var cep = $("#txtDisplayCep").val();
    var complemento = $("#txtDisplayComplemento").val();
    //var complemento = "vazio";
    var bairro = $("#txtDisplayBairro").val();
    var cidade = $("#txtDisplayCidade").val();
    var estado = $("#txtDisplayEstado").val();
    var pais = $("#txtDisplayPais").val();

    tx.executeSql("INSERT INTO ENDERECO (cep, rua, numero,
complemento, bairro, cidade, estado, pais) VALUES ('" + cep +
```

```
"', '" + rua + "', '" + numero + "', '" + complemento +  "',
'" + bairro + "', '" + cidade + "', '" + estado +"', '" + pais
+ "');");


}

function inserirCoordenadas(inserirPerfis){

    // INSERIR TABELA "DEMANDA"
    var latitude = $("#txtDisplayLatitude").val();
    var longitude = $("#txtDisplayLongitude").val();
    var dataCaptura = new Date();

    var idDemanda = getDemandaFK();

    getDemandaFK().done(function(demandaFK){
        var demandaFK;

        getDeviceFK().done(function(deviceFK){
            var deviceFK;

            getEnderecoFK().done(function(enderecoFK){

                var enderecoFK;

                var db = window.openDatabase("georreferencia",
"1.0", "my db", 2*1024*1024);

                db.transaction (function (t){
                    t.executeSql("INSERT INTO COORDENADA
(imei, idDemanda, idEndereco, latitude, longitude,
dataCaptura) VALUES ('" + deviceFK + "', '" + demandaFK + "',
'" + enderecoFK + "', '" + latitude + "', '" + longitude + "',
'" + dataCaptura + "');");
                }, successCBGeo, erroDbGeo);

            });
        });
    });

    inserirPerfis();

}


function inserirPerfis(){

    getCoordenadaFK().done(function(idCoordenadaFK){
        var idCoordenadaFK;

        getValorAplicacao().done(function(codigoPerfil){
            var codigoPerfil;
```

```
            // Se código da aplicação == 1 --> Mostrar
cadastro do Bus.AO
            if (codigoPerfil == 1){

                var tipo = $("#txtTipo").val();
                var caracteristica =
$("#txtCaracteristica").val();
                var atividadeBusao =
$("#txtAtividadeBusao").val();

                var db = window.openDatabase("georreferencia",
"1.0", "my db", 2*1024*1024);

                db.transaction (function (t){
                    t.executeSql("INSERT INTO PERFILBUSAO
(idCoordenada, tipo, caracteristica, atividade) VALUES ('" +
idCoordenadaFK + "', '" + tipo + "', '" + caracteristica + "',
'" + atividadeBusao + "');");
                }, successCBGeo, erroDbGeo);

                alert("Perfil do Bus.AO adicionado com
sucesso");

                $("#txtTipo").val("");
                $("#txtCaracteristica").val("");
                $("#txtAtividadeBusao").val("");
            }

            // Se código da aplicação == 2 --> Mostrar
cadastro do Click1Gas
            else if (codigoPerfil == 2){

                var razaoSocial = $("#txtRazaoSocial").val();
                var nomeFantasia =
$("#txtNomeFantasia").val();
                var comercializa =
$("#txtComercializa").val();
                var atividadeClick1Gas =
$("#txtAtividadeClick1Gas").val();

                var db = window.openDatabase("georreferencia",
"1.0", "my db", 2*1024*1024);

                db.transaction (function (t){
                    t.executeSql("INSERT INTO PERFILCLICK1GAS
(idCoordenada, razaoSocial, nomeFantasia, comercializa,
atividade) VALUES ('" + idCoordenadaFK + "', '" + razaoSocial
+ "', '" + nomeFantasia + "', '" + comercializa + "', '" +
atividadeClick1Gas + "');");
                }, successCBGeo, erroDbGeo);
```

```
                alert("Perfil do Click1Gás adicionado com
sucesso");

                $("#txtRazaoSocial").val("");
                $("#txtNomeFantasia").val("");
                $("#txtComercializa").val("");
                $("#txtAtividadeClick1Gas").val("");
            }

            else{
                alert("Erro ao captar codigoPerfil");
            }


            $("#txtDisplayRua").val("");
            $("#txtDisplayNumero").val("");
            $("#txtDisplayCep").val("");
            $("#txtDisplayComplemento").val("");
            $("#txtDisplayBairro").val("");
            $("#txtDisplayCidade").val("");
            $("#txtDisplayEstado").val("");
            $("#txtDisplayPais").val("");

        });
    });

}


//-----------------------------------------------------------------
---------------------------------
//           CARREGAR O MAPA, MARCADOR E ABRIR O PAINEL
DIREITO PARA CADASTRO DE PONTOS
//-----------------------------------------------------------------
---------------------------------

var marker;
var map;
var pos;
var geocoder = new google.maps.Geocoder();
function initializeGeorreferencia() {

    //inicializar mapa
    var numero = "",
        rua = "",
        cep = "",
        bairro = "",
        cidade = "",
        estado = "",
        pais = "", i;

    var mapOptions = {
        zoom: 14,
```

```
        disableDefaultUI: true
    };

    map = new
google.maps.Map(document.getElementById('divPageBody'),mapOpti
ons);

    if(navigator.geolocation) {

navigator.geolocation.getCurrentPosition(function(position) {

            pos = new
google.maps.LatLng(position.coords.latitude,position.coords.lo
ngitude);

            map.setCenter(pos);
            var image = "img/marcador64px.png";
            marker = new google.maps.Marker({
                position: pos,
                map: map,
                icon: image
            });

            //CRIAÇÃO DO MARCADOR E OS EVENTOS SOBRE ELE
            google.maps.event.addListener(marker, 'click',
function() {


getValorAplicacao().done(function(codigoPerfil){
                    var codigoPerfil;

                    if  (codigoPerfil == 1){
                        $("#perfilClick1Gas").css("display",
"none");
                    }
                    else if (codigoPerfil == 2){
                        $("#perfilBusao").css("display",
"none");
                    }
                    else{
                        alert("Erro ao captar codigoPerfil");
                    }
                });

                $("#divDireitaBody").html("");

$("#divDireitaBody").removeClass("appbaseBodyDireita");
                $("#divDireitaBody").css("background-
color","#F3F3F3");



$("#divDireitaBody").load("cadastroOndeEstou.html");
```

```
                iniciaGeocoder(this);
                $("#pnlDireita").panel("open");

            });

            },function(){
                handleNoGeolocation("Não encontrou
coordenadas");
        });
    }else{
        handleNoGeolocation("html5 - geolocation não
suportado");
    }

    google.maps.event.addDomListener(window,'load',
initializeOndeEstou);

}

function iniciaGeocoder(marker){

    var pos = marker.getPosition();

    geocoder.geocode({'latLng': pos}, function(results,
status) {
        if (status == google.maps.GeocoderStatus.OK)
        {
          if (results[0])
          {

$("#txtDisplayLatitude").val(marker.getPosition().lat());

$("#txtDisplayLongitude").val(marker.getPosition().lng());

            for (i = 0; i <
results[0].address_components.length; i++){

                if (results[0].address_components[i].types[0]
== "street_number"){
                    numero =
results[0].address_components[i].short_name;
                    $("#txtDisplayNumero").val(numero);
                }
                else if
(results[0].address_components[i].types[0] == "route"){
                    rua =
results[0].address_components[i].short_name;
                    $("#txtDisplayRua").val(rua);
                }
                else if
(results[0].address_components[i].types[0] == "postal_code"
```

```
                            ||
results[0].address_components[i].types[0] ==
"postal_code_prefix"){
                        cep =
results[0].address_components[i].long_name;
                        $("#txtDisplayCep").val(cep);
                    }
                    else if
(results[0].address_components[i].types[0] == "neighborhood"){
                        bairro =
results[0].address_components[i].long_name;
                        $("#txtDisplayBairro").val(bairro);
                    }
                    else if
(results[0].address_components[i].types[0] ==
"administrative_area_level_2"){
                        cidade =
results[0].address_components[i].long_name;
                        $("#txtDisplayCidade").val(cidade);
                    }
                    else if
(results[0].address_components[i].types[0] ==
"administrative_area_level_1"){
                        estado =
results[0].address_components[i].short_name;
                        $("#txtDisplayEstado").val(estado);
                    }
                    else if
(results[0].address_components[i].types[0] == "country"){
                        pais =
results[0].address_components[i].long_name;
                        $("#txtDisplayPais").val(pais);
                    }
                }

            }
            else
            {
              alert('Nenhum resultado encontrado!');
            }
          }
          else
          {
            alert('Falha no Geocoder devido a: ' + status);
          }
          configuraAlturaOndeEstou();
      });
}


//-----------------------------------------------------------------
--------------------------------
```

```
//              FUNÇÕES DE SELEÇÃO NO BANCO PARA POSSIBILITAR
RETORNOS
//-----------------------------------------------------------------
--------------------------------

function getDemandaFK() {
    var results = "";
    var def = $.Deferred();

    var db = window.openDatabase("georreferencia", "1.0", "my
db", 2*1024*1024);
    db.transaction(function(tx) {

        tx.executeSql("SELECT * FROM DEMANDA", [],
function(tx, results) {
            var demandaFK = results.rows.item(0).idDemanda;
            def.resolve(demandaFK);
        }, erroDbGeo);
    }, erroDbGeo);
    return def.promise();
}

function getDeviceFK() {
    var results = "";
    var def = $.Deferred();

    var db = window.openDatabase("georreferencia", "1.0", "my
db", 2*1024*1024);
    db.transaction(function(tx) {

        tx.executeSql("SELECT * FROM DEVICE", [], function(tx,
results) {
            var imeiFK = results.rows.item(0).imei;
            def.resolve(imeiFK);
        }, erroDbGeo);
    }, erroDbGeo);
    return def.promise();
}

function getEnderecoFK() {
    var results = "";
    var def = $.Deferred();

    var db = window.openDatabase("georreferencia", "1.0", "my
db", 2*1024*1024);
    db.transaction(function(tx) {

        tx.executeSql("SELECT * FROM ENDERECO order by
idEndereco DESC", [], function(tx, results) {
            var enderecoFK = results.rows.item(0).idEndereco;
            def.resolve(enderecoFK);
        }, erroDbGeo);
    }, erroDbGeo);
```

```
    return def.promise();
}

function getValorAplicacao() {
    var results = "";
    var def = $.Deferred();

    var db = window.openDatabase("georreferencia", "1.0", "my
db", 2*1024*1024);
    db.transaction(function(tx) {

        tx.executeSql("SELECT * FROM DEMANDA", [],
function(tx, results) {
            var codigoPerfil = results.rows.item(0).aplicacao;
            def.resolve(codigoPerfil);
        }, erroDbGeo);
    }, erroDbGeo);
    return def.promise();
}

function getCoordenadaFK() {
    var results = "";
    var def = $.Deferred();

    var db = window.openDatabase("georreferencia", "1.0", "my
db", 2*1024*1024);
    db.transaction(function(tx) {

        tx.executeSql("SELECT * FROM COORDENADA order by
idCoordenada DESC", [], function(tx, results) {
            var idCoordenadaFK =
results.rows.item(0).idCoordenada;
            def.resolve(idCoordenadaFK);
        }, erroDbGeo);
    }, erroDbGeo);
    return def.promise();
}

//-----------------------------------------------------------------
--------------------------------
//                                ESTADOS DE ERRO E SUCESSO
//-----------------------------------------------------------------
--------------------------------


function handleNoGeolocation(erro){
    console.log("Erro ao carregar o mapa: "+erro);
}

function onfail (){
    alert("erro geolocation ");
}
```

```
function erroDbGeo(err) {
    alert("[Geo]Processo de Erro SQL: "+err.message);
}

function successCBGeo() {
    alert("[Geo]Sucesso!");
}
```