Desenvolvimento de uma aplicação móvel multiplataforma para gerenciar clientes

Maichel Junior de Oliveira Rosa[1], Ricardo Frohlich da Silva[1]

[1]Curso de Sistemas de Informação – Centro Universitário Franciscano
Santa Maria – RS – Brasil

maicheljr@gmail.com, ricardosma@gmail.com

Abstract: *This study aimed to hum the application development platform mobile to manage clients and visitors. The development was based environment using web Ionic Framework and AngularJS . Through plugins Apache Cordova was possible to have access to native device features , How GPS, and collect as geographic coordinates of customers Making local storage. With information on this basis , it was possible generate maps and routes using Google Maps API. Using the Intel XDK was possible generate a native application for Android.*

Resumo: *Este trabalho teve como objetivo o desenvolvimento de um aplicativo móvel multiplataforma para gerenciar clientes e visitas. O desenvolvimento foi baseado em ambiente web utilizando o Ionic Framework e AngularJS. Através dos plugins do Apache Cordova foi possível ter acesso aos recursos nativos do dispositivo, como GPS, e coletar as coordenadas geográficas dos clientes fazendo o armazenamento local. Com base nessas informações, foi possível gerar mapas e percursos utilizando API do Google Maps. Utilizando o Intel XDK, foi possível gerar uma aplicação nativa para Android.*

1. Introdução

Para Konish *and* Ribeiro (2012), Sistema de informação geográfico (SIG) é um sistema baseado em coordenadas geográficas, esse sistema permite, entre outros casos, manipular, modelar, consultar e analisar dados geograficamente referenciados.

Uma das tendências apontadas pelo Gartner para 2015 é a maior ênfase no atendimento das necessidades do usuário móvel, independentemente dos ambientes, em vez de focar apenas no dispositivo. Sendo assim, os aplicativos híbridos representam uma evolução, principalmente no ambiente corporativo [Moretti 2015].

Aplicativos multiplataforma são o caminho para distribuir o código entre diferentes plataformas sem a necessidade de criar diferentes implementações. Consiste em criar uma única aplicação utilizando as tecnologias de desenvolvimento Web *HyperText Markup Language* (HTML), *JavaScript* e *Cascading Style Sheets* (CSS), que através de ferramentas específicas poderá ser disponibilizada para vários dispositivos [Fowler 2012].

Na elaboração deste trabalho procura-se criar uma aplicação móvel que possa contribuir na execução de tarefas diárias dos profissionais de representação comercial, criando uma carta de clientes estruturada, possibilitando ter controle e efetividade nas suas visitas, bem como visualiza-los no mapa e gerar um percurso, até um cliente selecionado, a partir de sua localização utilizando uma *Application Programming Interface* (API) do Google Maps.

Visando explorar novas alternativas para o desenvolvimento mobile, como o uso do *Ionic Framework* para criar a interface gráfica, o AngularJS como linguagem de programação e o Apache Cordova para o acesso ao hardware do dispositivo, justifica-se a elaboração deste trabalho de modo a contribuir à cerca de tecnologias de desenvolvimento multiplataforma.

1.1. Objetivo Geral

O objetivo do trabalho é desenvolver uma aplicação híbrida e multiplataforma, utilizando a ferramenta Intel XDK e *Ionic Framework*, que possibilitasse a sua execução nos principais sistemas operacionais móveis existentes atualmente, como IOS, Android e WP.

Esta aplicação tem a finalidade de gerenciar a carteira de clientes de profissionais da área comercial, organizando melhor suas atividades, criando um repositório com as principais informações desses clientes, obtendo uma visão macro de onde eles estão agrupados em relação a sua posição atual e utilizando a ferramenta de mapas é possível criar uma rota até um cliente previamente selecionado, finalizando as funções da aplicação, conta também com uma ferramenta de agendamento de visitas, permitindo efetuá-las e finalizá-las.

1.2 Objetivos Específicos

Os objetivos específicos do trabalho são:

- Realizar uma pesquisa sobre desenvolvimento *mobile* multiplataforma, usando o Intel XDK e o *Ionic Framework*.
- Desenvolvimento de uma aplicação para aos profissionais da área comercial com o objetivo de organizar seus clientes e agendamentos;
- Testar a aplicação quanto a fluidez e usabilidade.

2. Referencial Teórico

Esta seção tem como objetivo apresentar os estudos bibliográficos, descrever a devida conceituação dos elementos componentes da proposta, metodologia e notações que atuam como a base de conhecimento deste projeto, entre eles: Sistema de Informação Geográfica, Dispositivos Móveis, *Ionic Framework,* Apache Cordova e Intel XDK.

2.1. Sistema de Informação Geográfica

Conforme Pena (2015) os Sistemas de Informações Geográficas são definidos como sendo equipamentos e meios tecnológicos para se estudar o espaço terrestre. Na maioria das vezes são utilizados por pesquisadores, empresas, Organizações Não Governamentais (ONG), governos, serviços de inteligência, entre outros.

Uma base de dados geográfica é um grupo de informações composta por fatos ou conceitos do mundo real constituída de atributos convencionais e atributos espaciais que descrevem sua forma e indicam sua localização na Terra [BIT 2015].

Compilando os parágrafos anteriores, é possível verificar que aplicações baseadas em coordenadas geográficas, podem produzir sistemas de apoio à decisão, ajudando na execução de tarefas diárias.

2.2. Dispositivos Móveis

O mercado brasileiro é um dos maiores consumidores de aplicativos móveis do mundo, ao movimentar cerca US$ 25 bilhões em 2013, de acordo com dados do Ministério da Ciência e Tecnologia. Ainda de acordo com o órgão, a expectativa é que esse setor atinja os US$ 70 bilhões em 2017. De carona nessas possibilidades de crescimento, as empresas travam agora uma corrida para oferecer soluções que facilitem a migração das telas do computador para dispositivos móveis, mas elas ainda têm um longo desafio para suprir as novas demandas e exigências dos usuários [Torres 2014].

A procura crescente por essa plataforma reflete na demanda por profissionais especializados no assunto. Estima-se que neste ano serão investidos cerca de US$ 38 bilhões em aplicativos, o que aumenta a demanda por profissionais capacitados dentro dessa área [G1 2015].

2.3. *Ionic Framework*

O *Ionic* é um *Framework* para desenvolvimento *mobile Front End* baseado no modelo de arquitetura *Model View Controller* (MVC), integrando HTML, CSS e JavaScript [Ionic Framework 2016].

A instalação se dá através do terminal, é necessário ter instalado o node.JS no computador. Com acesso ao terminal é possível criar um projeto em branco, baseado em *tabs* ou *sidemenus* [Ionic Framework 2016].

Ionic foi criada por Max Lynch, Ben Sperry, e Adam Bradley da empresa Drifty Co. em 2013. Os aplicativos podem ser construídos com tecnologias da Web e, em seguida, distribuídos em dispositivos móveis através da integração com Cordova [Tyagi et al. 2016].

Para interagir com os recursos nativos do dispositivo, como a câmera, o som ou GPS, existe uma biblioteca com mais de 70 *plugins* chamada ngCordova, escrita usando a estrutura do AngularJS, facilitando as implementações de código, pois todos os *plugins* possuem um código de exemplo [NgCordova 2016].

2.4. Apache Cordova

Apache Cordova é um conjunto de APIs que permitem que um desenvolvedor de aplicações móveis possa acessar os recursos nativos do dispositivo, entre eles, a câmera, o acelerômetro e GPS [Apache Cordova 2015].

Na prática, a aplicação nativa gerada pelo Cordova é apenas um contêiner que interpreta conteúdo HTML, JavaScript e CSS, de forma, é similar ao que um navegador web comum faz, exceto pelo fato de não possuir barra de endereço ou histórico de navegação [Consegi 2013].

Ao usar as APIs Cordova, um aplicativo pode ser construído sem qualquer código nativo (Java, Objective-C, Swift entre outros). Em vez disso, são utilizadas tecnologias web, e eles estão hospedados localmente no dispositivo, em geral, e não em um servidor *Hypertext Transfer Protocol* (HTTP) remoto [Cavalca 2015].

2.5. Intel XDK

O Intel XDK consiste em um conjunto de ferramentas para desenvolver, depurar, testar e construir aplicativos web móveis e híbridos, para múltiplas plataformas de destino [Intel XDK 2015].

O Intel XDK é uma plataforma de desenvolvimento híbrido que permite a utilização de linguagens e tecnologias amplamente difundidas na comunidade de desenvolvimento de *software*, como HTML5, CSS3 e Java Script. Além disso, a plataforma Intel XDK permite a criação de aplicativos nativos para as plataformas líderes de mercado mobile, isso com apenas um único projeto implementado [Silva 2015].

2.5.1 Considerações Finais Sobre Referencial Teórico

É possível visualizar na Figura 1, como acontece a integração das tecnologias referenciadas, escolhido o *Ionic* como *framework* de desenvolvimento *mobile*, será usado o AngularJS como linguagem de programação. Para acessar os recursos nativos dos dispositivos será usado o Apache Cordova, a compilação e *build* do projeto será feita usando o Intel XDK, tendo como produto final a aplicação dentro de um pacote nativo.

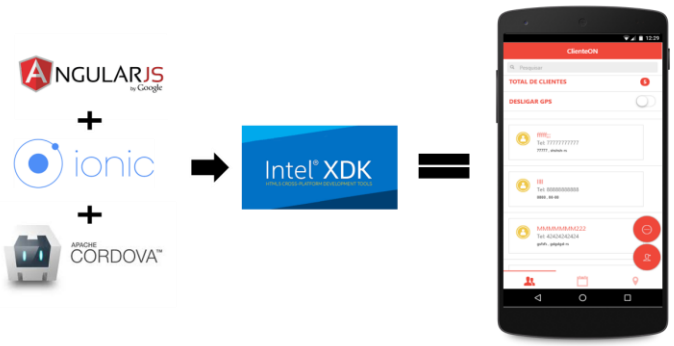

Figura 1. Integração das Tecnologias Referenciadas.

2.6 Trabalhos Relacionados

Nesta seção serão tratados sobre os trabalhos que possuem relação com a proposta do presente trabalho, envolvendo tanto soluções comerciais quanto trabalhos acadêmicos.

2.6.1 Chronos Mobi: uma aplicação móvel multiplataforma para o gerenciamento de projetos;

Nesse artigo, Kasperbauer *et al.* (2013), propuseram uma aplicação multiplataforma para o gerenciamento de projetos, utilizando o Phonegap como *Framework* para o desenvolvimento.

A motivação para a elaboração do trabalho surgiu da necessidade de sincronizar as atividades dos seus profissionais que geograficamente estavam em locais diferentes e não havia coordenação entre eles.

A aplicação possui uma topologia cliente/servidor, a troca de informações, entre o *webservice* desenvolvido em linguagem Java e os dispositivos móveis, são feitas por um arquivo *Json,* onde é processado localmente e montando as informações em HTML na tela do dispositivo.

No servidor está o dotProject que é uma aplicação web para gerenciamento de projetos, o Chronos Mobi consome as informações do banco de dados desta aplicação através do web service renderizando-as posteriormente nos dispositivos.

Na conclusão do trabalho, a aplicação, trouxe maior agilidade e otimização no gerenciamento dos projetos entre os *Stackholders.*

2.6.2 Protótipo de aplicativo móvel multiplataforma para consulta de estimativas de chegada das linhas de ônibus de Florianópolis;

Casagrande *and* Conceição (2014), trouxeram a proposta de criação de um protótipo de uma aplicação móvel multiplataforma, sua função seria informar aos usuários uma estimativa de tempo em que um ônibus, de uma determinada linha, irá demorar para passar na parada de ônibus, usando como referência as coordenadas geográficas capturadas através do GPS dos smartphones dos usuários.

Para Traeg (2013), há uma dificuldade em gerar e manter aplicativos para uma variedade de modelos e diferentes tipos de versões dos sistemas operacionais, com base nessa análise a proposta de desenvolvimento híbrido e multiplataforma seria ser uma alternativa viável.

2.6.3. Desenvolvendo aplicativos multiplataforma com tecnologias web.

Kasperbauer (2014), produziu um material de apoio sobre desenvolvimento multiplataforma utilizando o Phonegap como *Framework* de desenvolvimento, mostrando as possibilidades que os *plugins* dão ao desenvolvedor para acessar e interagir com o hardware do dispositivo.

Uma curiosidade deste artigo é que a ferramenta de desenvolvimento utilizada foi o Eclipse e a integrado com o Phonegap se deu através da instalação do *plugin* na IDE, onde se desenvolve usando linguagem nativa, porém, o método "onCreate" do Java é carregado e também um outro método chamado "loadUrl" que carrega a aplicação numa *webview* aplicando conceito de programação híbrida.

2.6.4. Considerações finais sobre trabalhos correlatados.

Nos três trabalhos citados é possível perceber que há o uso de tecnologias híbridas de desenvolvimento, como o Phonegap, por exemplo, evidenciando uma tendência a esse tipo de programação que permite desenvolver para múltiplas plataformas.

O último trabalho relacionado, serviu como guia para os primeiros passos no desenvolvimento deste projeto, mostrando na prática como seria uma aplicação híbrida e como interagir com os plugins do Cordova.

3. Metodologia

As metodologias ágeis usadas para o desenvolvimento de *software* são diferenciais que agregam valor ao produto final, visando aumentar a satisfação do cliente e auxiliar as organizações no decorrer do projeto [Filho 2012].

Essas metodologias são adaptativas, trabalhando com constante retorno dos clientes, permitindo adaptar-se rapidamente a eventuais mudanças nos requisitos. Também se destacam das metodologias tradicionais (tais como, Modelo Cascata, Modelo Espiral e *Rational Unified Process* (RUP) por darem maior prioridade a implementação das funcionalidades através do código, ao invés da produção da documentação escrita, sendo que assim, proporcionam um retorno mais rápido [Libardi 2010].

Para o desenvolvimento deste trabalho, foi escolhida a metodologia *Feature Driven Development* (FDD) pelo fato de atender os requisitos e possuir características importantes como alteração no escopo caso seja necessário, fazer entregas frequentes, mensuráveis e repetitivas.

3.1 *Feature Driven Development*

A metodologia FDD foi criada entre 1997 e 1999, em Cingapura, por Jeff De Luca e Peter Coad para o desenvolvimento de *software*. Inclui alguns benefícios de processos rigorosos, como modelagem, planejamento prévio e controle do projeto, contendo também características de processos ágeis, como foco na programação, interação constante com o cliente e entrega frequente da versão do produto [Filho 2012].

Para o desenvolvimento deste trabalho foi utilizada a metodologia ágil *Feature Driven Development*, (Desenvolvimento Guiado por Funcionalidades), que é uma metodologia para gerenciamento e desenvolvimento de *software*. Ela combina as melhores práticas do gerenciamento ágil de projetos com uma abordagem completa para Engenharia de *Software* orientada por objetos, sempre mantendo resultados frequentes, tangíveis e funcionais [Heptagon 2014].

Com base na Figura 2 é possível descrever as duas fases da metodologia na seguinte forma:

- **Concepção e Planejamento:** tem como objetivo promover o entendimento inicial e mais abrangente sobre o escopo do sistema a ser desenvolvido. Geralmente tem duração de uma a duas semanas e é executada apenas uma vez.

- **Construção:** tem como objetivo o foco no desenvolvimento e entrega das funcionalidades de maneira incremental e iterativa durante um período de até duas semanas.

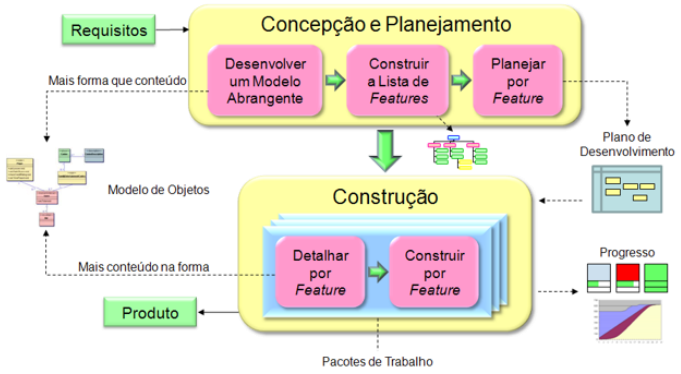

Figura 2. Fases do FDD [Heptagon 2014].

Essas duas fases se subdividem em cinco processos: Desenvolver Modelo Inicial, Criar Lista de Funcionalidades, Planejar por Funcionalidade, Arquitetar por Funcionalidade e Construir por Funcionalidade. Cada processo contém requisitos a serem atendidos e tarefas a serem executadas e dentre estas tarefas, algumas são obrigatórias e outras não [FDD 2002].

3.1.1. Desenvolver Modelo Inicial

O processo denominado Desenvolver Modelo Inicial, é o primeiro processo do ciclo de vida de um projeto desenvolvido com FDD. O resultado é uma arquitetura inicial, chamada de *object model* (Modelo de Dados Abrangente). Realiza-se um estudo dirigido sobre o escopo do sistema e seu contexto. Então, são realizados estudos mais detalhados sobre o domínio do negócio para cada área a ser modelada. Esse estudo pode tomar forma como um diagrama de domínio, ilustrando a estrutura física [FDD 2002].

Para o desenvolvimento deste projeto montou-se o diagrama de classes a seguir na Figura 3, trazendo visão simplificada apenas, pois não possui todas as classes implementadas.

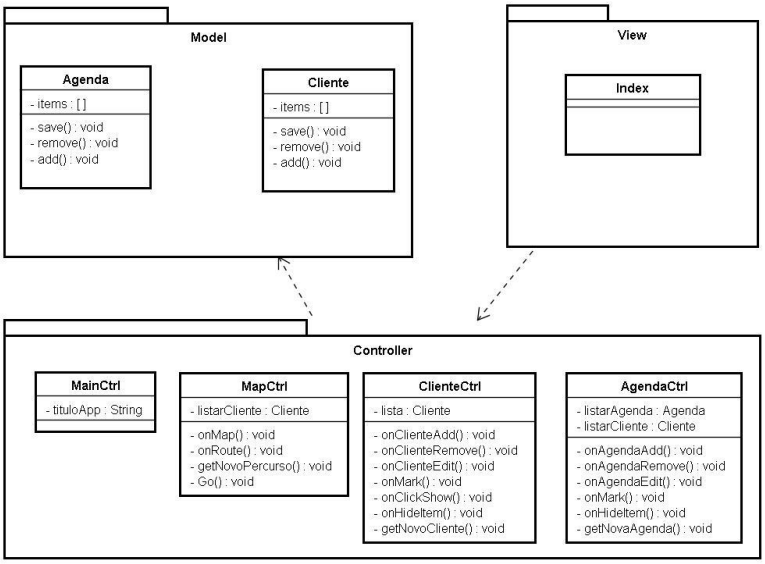

Figura 3 – Diagrama de Classes

3.1.2 Criar Lista de Funcionalidades

É uma atividade que abrange todo o projeto, para identificar todas as funcionalidades que satisfaçam os requisitos. Esse processo tem como critérios de saída, uma Lista de Funcionalidades, que pode ser representada por um conjunto de Requisitos Funcionais (RF) e Requisitos Não Funcionais (RNF) que devem descrever as necessidades reais do negócio do ponto de vista do cliente [FDD 2002].

Para a construção do referido sistema, foram propostos os seguintes Requisitos Funcionais (RF) e Requisitos Não Funcionais (RNF) a seguir:

- RF01: Gerenciar Clientes
  - RF01.1 – Cadastrar Clientes
  - RF01.2 – Editar Clientes
  - RF01.3 – Excluir Clientes
  - RF01.4 – Pesquisar Clientes

- RF02: Gerenciar Visitas
  - RF02.1 – Agendar Visita
  - RF02.2 – Editar Agendamento
  - RF02.3 – Excluir Agendamento
  - RF02.4 – Finalizar Visita
  - RF02.5 – Pesquisar Agendamento

- RF03: Gerenciar Mapas
  - RF03.1 – Localizar Usuário (*Find me*)
  - RF03.2 – Visualizar Clientes
  - RF03.3 – Gerar Percurso

- RNF01: Armazenamento dos dados localmente (*LocalStorage*)

- RNF02: Usar a API do Google Maps para gerar os percursos
- RNF03: Usar o *Ionic Framework* para criação de *layout*
- RNF04: Linguagem de programação AngularJS;

3.1.3 Planejar por Funcionalidades

É uma atividade do projeto que tem como objetivo produzir um plano de desenvolvimento. Deve-se planejar a ordem na qual as funcionalidades serão implementadas, baseada nas dependências entre elas e também em sua complexidade. Como resultado o plano de desenvolvimento deve ser constituído por uma tabela descritiva das funcionalidades e seus tempos de desenvolvimento estimado [FDD 2002].

3.1.4 Arquitetar por Funcionalidades

Para Heptagon (2014), é a atividade que abrange todo o projeto, identificando todas as funcionalidades e satisfazendo todos os requisitos. Geralmente é composta apenas por programadores líderes, decompondo funcionalmente o domínio por áreas e seguindo os passos dentro de cada regra de negócio gerando assim a lista de funcionalidades.

Analisando o diagrama de caso de uso, na Figura 4, nota-se que os requisitos funcionais estão todos dispostos no contexto de cada funcionalidade a fim de proporcionar uma visão das funcionalidades da aplicação.

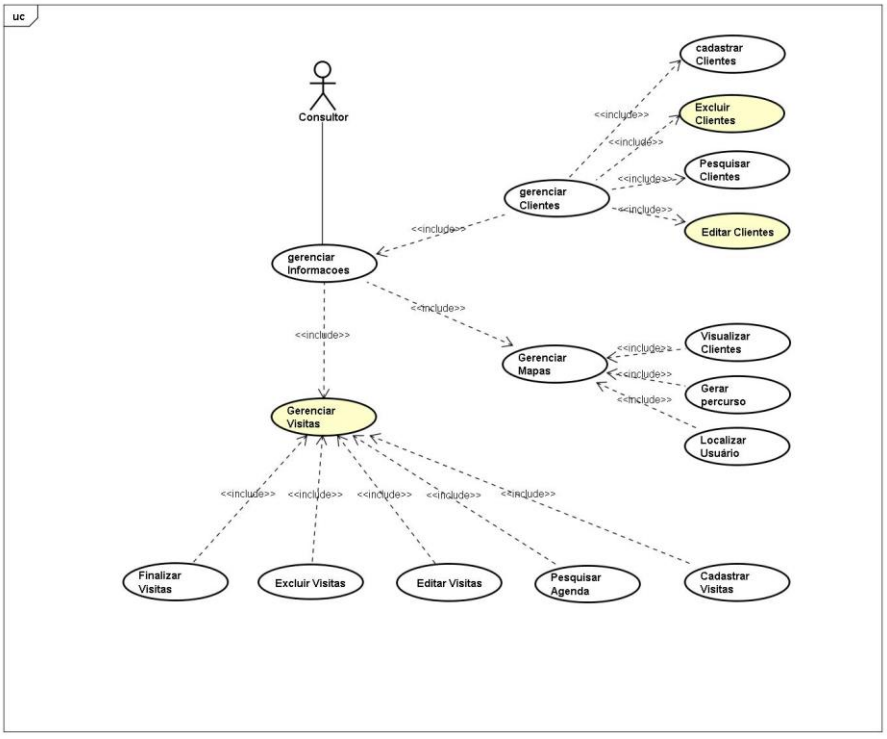

Figura 4 – Diagrama de caso de Uso

É possível verificar o detalhamento do modo de operação do sistema, pelo diagrama de processos, descrito na Figura 5, que indica o fluxo das três atividades principais.

O usuário pode optar por não coletar as coordenadas GPS na tela inicial de cadastro de clientes, é possível assim fazer o processo em massa e durante as visitas coletar as coordenadas editando os cadastros já existentes.

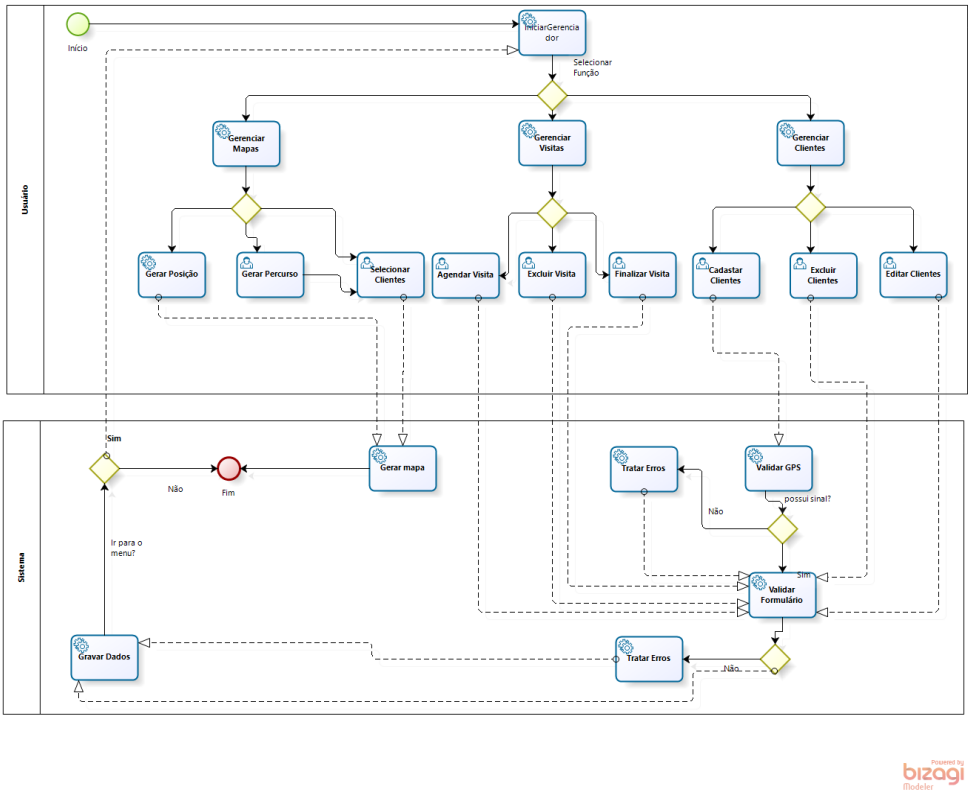

Figura 5 – Diagrama de Processos

O *Ionic Framework* tem por característica estruturar o projeto usando arquitetura *Model View Controller* (MVC) [*Ionic Framework* 2015].

Projetos no formato MVC interagem com o usuário através várias *Views*, no caso deste projeto o usuário interage com apenas uma *View*, todas as interações estão inseridas na página Index.html, desta forma, dependendo do menu que está sendo usado, a *View* consulta o *Controller* correspondente e executa as rotinas definidas trocando informações com as *Models*.

3.1.5 Construir por Funcionalidade

Tem como objetivo produzir (implementar um dos itens planejados e arquitetados para construir cada funcionalidade) uma função com valor para o cliente (funcionalidade), ou seja, ao final deste processo, tem uma funcionalidade implementada em código testado e inspecionado, pronto para ser entregue. Esse processo se inicia com Pacote de Arquitetura construído no processo anterior pronto [FDD 2002].

A Figura 6 a seguir representa a interface de cadastro de clientes, é possível desabilitar a coleta das coordenadas geográficas do local durante o cadastramento, eventualmente queira fazer o cadastramento em local diferente, posteriormente ele ir até o local correto, clicar em editar e coletar as coordenadas.

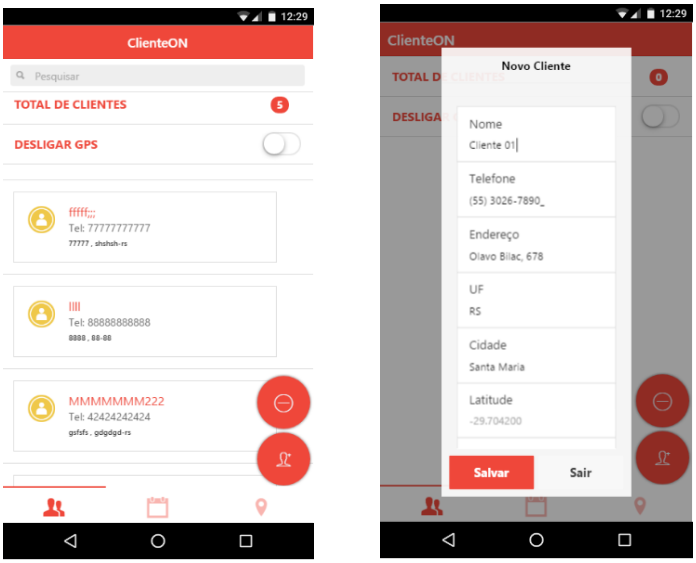

Figura 6 – Interface de cadastro de clientes.

Na Figura 7 é possível visualizar o trecho de código para cadastro de clientes, ao clicar no botão adicionar clientes na tela de cadastro a diretiva ng-click do AngularJS chama o método "onClienteAdd()" que está dentro do respectivo Controller, esse método possui uma função que cria um vetor de dados vazio contendo os dados do cliente e o envia para a função "getNovoCliente()" junto a um outro parâmetro booleano que identifica se é um cadastro novo ou não.

```
52  <button class="btn-clie50" ng-click="onClienteAdd()"><i class="ion-ios-personadd-outline"></i></button>

168       //metodo que adiciona novos clientes
169       $scope.onClienteAdd = function(){
170       var item = {nome:"",telefone:"",endereco:"",uf:"",cidade:"",lat:"",lng:""};
171       getNovoCliente(item,true);
172
173       };

59       function getNovoCliente(item,novo ){
60       $scope.lista =  cliente.items;
```

Figura 7 – Trecho de código adicionar Cliente.

Na Figura 8 é possível visualizar um outro trecho de código, dessa vez da função "getNovoCliente()", se o segundo parâmetro recebido for *true* então é gravado um novo item na lista chamando o método "cliente.add(item)", da classe Cliente, passando o item como parâmetro, senão, apenas salva os novos dados usando o método "cliente.save()", processo idêntico ao que ocorre com o método "onClienteEdit()".

```
117                 if(novo){
118                   cliente.add(item);
119
120                   $cordovaToast.showShortCenter('Cliente inserido com sucesso!')
121                   .then(function(success) {
122                   }, function(error) {  });
123
124                 }
125                   cliente.save();
126
127                   $cordovaToast.showShortCenter('Cliente editado com sucesso!')
128                   .then(function(success) {
129                   }, function(error) {  });
130
```

Figura 8 – Trecho de código função getNovoCliente.

A Figura 9 traz um trecho de código da classe Cliente sendo possível verificar como o método de *save()* inseri um item na lista de clientes.

```
11    // metodo para salvar os dados
12    this.save = function (){
13
14      //converter a uma string para formato json
15      var lista = angular.toJson(this.items);
16      localStorage.setItem("listaCliente",lista);
17    }
18
19    //adicionar item na lista de cliente
20    this.add = function(item){
21      this.items.push(item);
22    };
```

Figura 9 – Trecho de código função getNovoCliente.

Na Figura 10 é possível visualizar a interface do menu Mapas. Acionando o primeiro botão, de cima para baixo, é ativado o gerador de percurso, a partir da posição atual do usuário até um cliente selecionado. O segundo botão imprimi todos os clientes cadastrados no mapa. Ao acionar o primeiro botão, o sistema irá localizar o coletar a coordenada atual do dispositivo e localizá-lo no mapa. Essa ferramenta está usando uma API do Google Maps e precisa de acesso a uma rede dados para gerar os percursos.

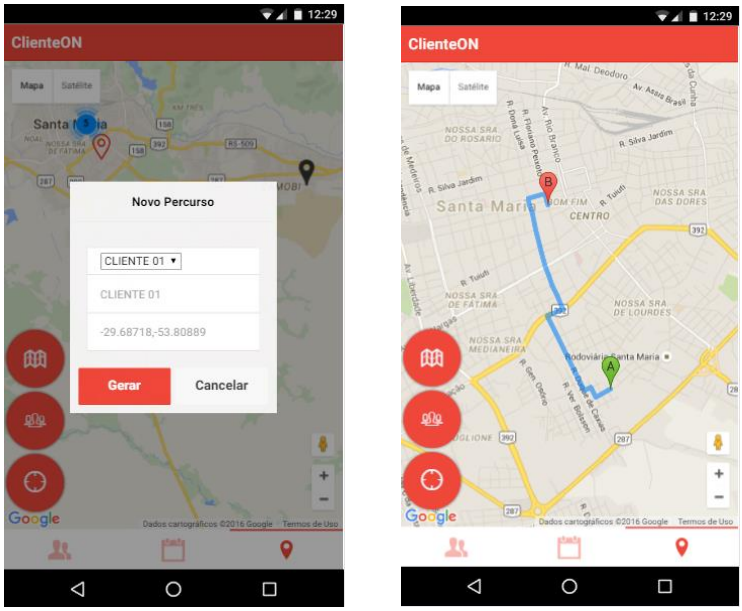

Figura 10 – Interface para Gerar Novo Percurso.

4. Resultados

O desenvolvimento do sistema proposto neste trabalho, permitiu que o produto final fosse um aplicativo funcional, do ponto de vista da usabilidade, para o gerenciamento e agendamento de visitas. O aplicativo foi testado em ambiente de desenvolvimento e obteve o funcionamento conforme o esperado.

Os testes de compilação foram executados utilizando o Intel XDK, embora o *Ionic* tivesse o seu próprio emulador, console e build, requer que esteja instalado um *Software Development Kit* (SDK) de desenvolvimento, e para compilações de pacotes IOS, necessariamente, é preciso ter o XCODE instalado, logo, um computador com o sistema operacional OSX.

O uso do *Ionic Framework* permitiu criar um aplicativo bem estruturado através da arquitetura MVC. Com a biblioteca ngCordova escrita em AngularJS agilizou as implementações de códigos, trazendo uma lista de opções de plugins que interagem com os recursos nativos do sistema operacional como *Alert*, *Toast*, *Geolocation*, *LocalStorage*, etc, permitindo que o aplicativo, em alguns casos, tivesse um aspecto e comportamento nativo.

Durante o desenvolvimento desse trabalho, houveram várias atualizações da ferramenta Intel XDK, trazendo instabilidade no momento de implementar o código, dessa forma, foi usado o ATOM como plataforma de desenvolvimento, passando para o Intel XDK a tarefa de fazer o *build* da aplicação no pacote nativo, gerenciar os plugins e suas respectivas versões.

Um diferencial do sistema foi trabalhar com API do Google Maps, para este trabalho, foram usados os serviços de *Directions Service*, *Geolocation* e *MarkerClusterer*, e foram executadas com êxito, visto que são implementações e serviços Web já difundidos e na linguagem *Javascript*.

Não foi possível realizar testes de funcionamento para os sistemas operacionais IOSX e Windows Phone, pois em ambos os sistemas, é necessário ter uma conta de desenvolvedor nas respectivas lojas de aplicativos sendo necessário digitar o usuário e senha durante o *build* usando o Intel XDK.

É possível concluir que o desenvolvimento deste trabalho ocorreu em conformidade com a proposta inicial de criar um aplicativo multiplataforma, obedecendo aos requisitos propostos e conduzidos pelos processos de produção da metodologia FDD.

5. Conclusões e Trabalhos Futuros

Desenvolver aplicativos para sistemas operacionais móveis diferentes requer, necessariamente, implementar o mesmo projeto em linguagem nativa para cada sistema operacional. Em alguns casos há ainda, uma fragmentação entre as versões de cada sistema, habilitando novas funções e descontinuando outras já obsoletas, por exemplo no sistema operacional (SO) Android, há versões 2.x, 4.x, 5.x, 6.x e na eminência de lançamento da versão Android N.

Ao trazer uma dinâmica diferente para criar aplicativos híbridos, o desenvolvimento multiplataforma permite implementar o mesmo código e através do Intel XDK, gerar um *build* para cada SO. Dessa forma, há um ganho de tempo e produtividade, com o reaproveitamento de código total.

A integração das tecnologias referenciadas, permitiram aumentar a dinâmica de produção do projeto, somadas, deram origem a uma aplicação visualmente elegante, funcional, dentro da proposta inicial e com potencial de uso corporativo para trabalhos futuros.

A metodologia usada no desenvolvimento do presente trabalho, foi a FDD, por ser uma metodologia ágil que busca o desenvolvimento por funcionalidades, com planejamento prévio e controle do projeto. Ao utilizar FDD abordou-se diretamente, a produção de sistemas utilizando uma metodologia de desenvolvimento simples de implementar e acompanhar. O presente trabalho foi conduzido de modo a contemplar o desenvolvimento a partir dos artefatos solicitados pela metodologia.

Para trabalhos futuros, principal objetivo é executar testes de funcionamento em outros sistemas operacionais. Poderão ser desenvolvidas novas funcionalidades como integração da agenda do aplicativo com o calendário nativo dos dispositivos, vinculado a uma conta de email. Poderá ser criada uma nova funcionalidade ao menu Mapas sugerindo uma rota a ser percorrida baseada na agenda diária. Criar uma arquitetura cliente servidor aliviando o processamento de informações no dispositivo, fazendo do aplicativo um módulo de coleta, parte de um sistema macro, como uma rede de lojas, por exemplo.

Implementar um SIG poderá tornar o aplicativo mais efetivo em relação aos seus clientes, baseado nas coordenadas geográficas, criar um servidor gerencial que mantenha um painel com um mapa informando quem foi visitado no dia, quais produtos foram vendidos, incluindo percentuais de metas e desempenho de vendas.

Referências Bibliográficas

Apache cordova, (2015). Disponível em: https://cordova.apache.org. Acessado em: 16 de setembro de 2015.

Barleze, A.; Fusão de dados em esquemas híbridos envolvendo AGPS para localização de posicionamento. Universidade Católica do Paraná. Curitiba, (2003). Disponível em: https://www.ppgia.pucpr.br/pt/arquivos/mestrado/dissertacoes/2003/alessandro_barleze-2003l.pdf. Acessado em 18 de agosto de 2015.

BIT**,** Sistemas de Informações Geográficas, (2015), Disponível em: http://www2.transportes.gov.br/bit/01-inicial/sig.html. Acessado em: 29 de outubro de 2015.

Casagrande, P. A, Conceição V. S.; Protótipo de aplicativo móvel multiplataforma para consulta de estimativas de chegada das linhas de ônibus de Florianópolis, (2014), Disponível em: https://projetos.inf.ufsc.br/arquivos_projetos/projeto_1580/relatorio_final.pdf. Acessado em 27 de outubro de 2015.

Cavalca, D.; Seminário de Desenvolvimento Mobile, (2015), Disponível em: http://pt.slideshare.net/DiegoCavalca/seminrio-de-desenvolvimento-mobile-etec-cafelndia. Acessado em: 13 de novembro de 2015.

Consegi, VI Congresso Internacional de *Software* Livre e Governo Eletrônico, (2013), Disponível em: http://funag.gov.br/loja/download/1041-Consegi_2013_-_VI_Congresso_Internacional_de_*Software*_Livre_e_Governo_Eletronico.pdf. Acessado em 13 de novembro de 2015.

Decicino**,** R.; GPS: Sistema de Posicionamento Global tem diferentes utilidades,(2014), Disponível em: http://educacao.uol.com.br/disciplinas/geografia/gps-sistema-de-posicionamento-global-tem-diferentes-utilidades.htm. Acessado em: 08 de novembro de 2015.

FDD. (2002) "*Feature Driven Development Processes*", Disponível em:http://www.featuredrivendevelopment.com/files/fddprocessesA4.pdf, Acessado em abril de 2015.

Filho, Nivaldo T. (2012) "Estudo do Impacto do Uso das Metodologias Ágeis na Melhoria do Planejamento e Acompanhamento do Processo de Ensino e Aprendizagem em Sala de Aula", Fortaleza: Universidade Estadual do Ceará.

Fowler, Martin. Developing *Software* for Multiple Mobile Devices. (2012) Disponível em: http://martinfowler.com/articles/multiMobile/. Acesso em: 06 abril 2014.

G1, Veja os 9 profissionais que o mercado mais procura no início deste ano, (2015); Disponível em: http://glo.bo/16XgnVk. Acessado em: 13 de novembro de 2015.

Heptagon, Feature-Driven Development, Descrição dos processos, (2014), Disponível em: http://www.heptagon.com.br/files/FDD-Processos.pdf. Acessado em 13 de novembro de 2015.

Intel XDK, (2015); Disponível em: https://*software*.intel.com/en-us/intel-*software*-technical-documentation.Acessado em: 04 de agosto de 2015.

Ionic Framework, *Ionic* Documentation Overview, (2016); Disponível em: http://*IonicFramework*.com/docs/overview/. Acessado em: 26/05/2016.

Kasperbauer, M. et al; Chronos Mobi: uma aplicação móvel multiplataforma para o gerenciamento de projetos, (2013), disponível em: http://www.upf.br/seer/index.php/rbca/article/view/2774/2191. Acessado em: 01 de setembro de 2015.

Kasperbauer, M.; Desenvolvendo aplicativos multiplataforma com tecnologias web, (2014), disponível em: http://www.univale.com.br/unisite/mundo-j/artigos/56_multiplataforma.pdf. Acessado em: 10 de setembro de 2015.

Konish,R.K, Ribeiro, S. M.; Banco de dados Geográficos: Uma solução de baixo custo para empresas de pequeno e médio Porte, FATEC – São José dos Campos, (2012), Disponível em: http://fatecsjc.edu.br/trabalhos-de-graduacao/wp-content/uploads/2012/03/BDR1_renato_stephen2009.pdf. Acessado em: 10 de agosto de 2015.

Libardi, Paula L. O.; Barbosa, Vladimir. (2010) "Métodos Ágeis", Limeira: Universidade Estadual de Campinas.

Moretti, J.; Apps corporativos híbridos: tendência mobile nas empresas, (2015); Disponível em: http://cio.com.br/tecnologia/2015/01/29/apps-corporativos-hibridos-tendencia-mobile-nas-empresas/. Acessado em 13 de novembro de 2015.

NgCordova, Cordova With The Power Of Angularjs; (2016), Disponível em:http://ngcordova.com/. Acessado em 01 de janeiro de 2015

Pena, R. F.; SIG; *Brasil Escola*, (2015), Disponível em http://www.brasilescola.com/geografia/sig.htm. Acesso em 08 de outubro de 2015.

Silva, G. C. D. P. R. C. Da (2015). Aplicação Mobile De Apoio A Abordagem De Reuso De Requisitos. *Uinfebe*, N. 47, P. 98.

Torres, G.; Desafios Do Setor De Aplicativos Móveis, (2014), Disponível Em: Http://Www.Mobiletime.Com.Br/26/03/2014/Desafios-Do-Setor-De-Aplicativos-Moveis/372505/News.Aspx. Acessado Em: 20 De Outubro De 2015.

Traeg, P. Best Of Both Worlds: Mixing Html5 And Native Code, (2013), Disponível Em: Http://Mobile.Smashingmagazine.Com/2013/10/17/Best-Of-Bothworlds-Mixing-Html5-Native-Code/. Acesso Em: 06 Novembro 2015.

Tyagi, A., Gulhane, K., Mamarde, L. And Mokashi, N. (2016). Daily Needs Management Application. P. 838–842.