Fundação Educacional do Município de Assis
Instituto Municipal de Ensino Superior de Assis
Campus "José Santilli Sobrinho"

JAURI DA CRUZ JUNIOR

SOLUÇÃO MULTIPLATAFORMA PARA SMARTPHONE UTILIZANDO OS FRAMEWORKS SENCHA TOUCH E PHONEGAP INTEGRADO À TECNOLOGIA WEB SERVICE JAVA

Assis
2014

JAURI DA CRUZ JUNIOR

SOLUÇÃO MULTIPLATAFORMA PARA SMARTPHONE UTILIZANDO OS FRAMEWORKS SENCHA TOUCH E PHONEGAP INTEGRADO À TECNOLOGIA WEB SERVICE JAVA

Trabalho de Conclusão de Curso apresentado ao Instituto Municipal de Ensino Superior de Assis, como requisito do Curso de Análise e Desenvolvimento de Sistemas.

Orientador: Dr. Almir Rogério Camolesi
Área de Concentração: Desenvolvimento de Sistemas

Assis
2014

FICHA CATALOGRÁFICA

JUNIOR, Jauri da Cruz

Solução multiplataforma para smartphone utilizando os frameworks Sencha Touch e PhoneGap integrado à tecnologia web service java /
Jauri da Cruz Junior. Fundação Educacional do Município de Assis, 2014.
72 p.

Orientador: Dr. Almir Rogério Camolesi

Trabalho de Conclusão de Curso

Instituto Municipal de Ensino Superior de Assis – IMESA.

1. Aplicativos Mobile, **2.** Programação, **3.** PhoneGap, **4.** Sencha Touch, **5.** Linguagens de Script HTML5, CSS3, JavaScript.

CDD: 001.61

SOLUÇÃO MULTIPLATAFORMA PARA SMARTPHONE UTILIZANDO OS FRAMEWORKS SENCHA TOUCH E PHONEGAP INTEGRADO À TECNOLOGIA WEB SERVICE JAVA

JAURI DA CRUZ JUNIOR

Trabalho de Conclusão de Curso apresentado ao Instituto Municipal de Ensino Superior de Assis, como requisito do Curso de Análise e Desenvolvimento de Sistemas, analisado pela seguinte comissão examinadora:

Orientador: Dr. Almir Rogério Camolesi
Avaliador: Me. Fábio Eder Cardoso

Assis
2014

DEDICATÓRIA

Dedico esse trabalho à minha família, e em especial aos meus pais, pois eles me deram toda a estrutura e sabedoria de vida para me tornar o homem que sou hoje; e também dedico à minha noiva e futura esposa que sempre esteve ao meu lado, me dando a força necessária para alcançar os meus objetivos; e por fim, a todos os amigos.

AGRADECIMENTOS

Agradeço a Deus por todas as bençõe que recebi. Bençõe que foram muito mais do que pedi ou esperava receber. Deus em sua infinita misericórdia me deu, e continua dando paz, alegria, saúde, amor e coragem. A Ele toda glória, honra e louvor!

Ao Drº Almir Rogério Camolesi, meu orientador, pela sua fundamental contribuição no desenvolvimento do trabalho desde o início; pela paciência; e pelos ensinamentos repassados, não somente acadêmicos, mas ensinamentos de vida. Sem dúvidas é um grande professor.

À minha família, minha mãe Santa Aparecida de Souza e meu pai Jauri da Cruz, pois eles sempre me ensinaram a lutar pelos meus objetivos, sem desistir.

À minha noiva Claudinéia, minha fortaleza, que me apoiou nessa fase da vida nos momentos mais difíceis, sempre me dando força para continuar em frente. Obrigado pela compreensão, companheirismo e amizade. Eu te amo muito!

RESUMO

Os *smartphones* vieram para mudar de vez a forma que as pessoas interagem entre si e com o mundo. Isso tudo não pelo aparelho em si, mas sim, pelo que se consegue fazer com eles, devido às inúmeras aplicações e tecnologias atualmente disponíveis.

O fato do mercado dos *smartphones* estar segmentado em termos de plataformas (*iOS, Android, Windows Phone 7, etc.*) faz com que os desenvolvedores encontrem um novo caminho para atender todas essas tecnologias; com maior agilidade e sem perder os padrões de um bom aplicativo.

O presente trabalho tem como proposta apresentar análise e estudo de caso sobre o desenvolvimento de um aplicativo móvel multiplataforma, utilizando os frameworks *PhoneGap e Sencha Touch* que se baseiam em tecnologias web como: HTML, CSS e *JavaScritp.* Para o estudo de caso será apresentado o aplicativo "*Virtual Bartender".* Este tem a função de autoatendimento, para auxílio em bares; o que permitirá o cliente realizar seu pedido direto do seu *smartphone.*

Palavras-chave: *PhoneGap*; *Sencha Touch*; HTML; CSS; *JavaScript*.

ABSTRACT

The Smartphone have definitely come to change the form that the people interact between themselves and with the world. However, it does not happen because of the equipment, but what the people get to do with it, due to uncountable applications and technologies, which exist nowadays.

The fact of the Smartphone market be segmented in platforms (iOS, Android, Windows Phone 7, etc.) it does with the programmers find out one new way to attend all this technologies with a bigger agility and without lose the standard of a good application.

The propose of this work is present an analysis and a study of case about the development of one multiplatform mobile application using the frameworks PhoneGap and Sencha Touch, that are based on web technologies as HTML, CSS and JavaScript. For this study, the application “Virtual Bartender” is going to be presented. It has the function of assist the self-service in bars; it will permit to the client realize directly his solicitation through his own Smartphone.

Keywords: PhoneGap; Sencha Touch; HTML; CSS; JavaScript.

LISTA DE ILUSTRAÇÕES

Figura 1 - Comunicação dos Componentes. 26
Figura 2 - Arquitetura do PhoneGap (In: GATOL; PATEL,2012). 26
Figura 3 - Pacote PhoneGap. 28
Figura 4 - Arquitetura Orientada a Dados. 29
Figura 5 - Sistema Baseado em Camadas. 31
Figura 6 - Modelo de Camadas Mobile. 32
Figura 7 - Comunicação entre Dispositivos. 33
Figura 8 – PhoneGap Build. 34
Figura 9 - Mapa Mental 46
Figura 10 - Caso de Uso Administrador. 48
Figura 11 - Caso de Uso Funcionário. 49
Figura 12 - Caso de Uso Cliente 50
Figura 13 - Diagrama de Atividade Pedido via Smartphone. 51
Figura 14 – Diagrama de Atividade Menu Principal do Aplicativo. 52
Figura 15 - Diagrama de Sequencia Efetuar Pedido 53
Figura 16 - Diagrama de Classe Aplicação Web. 54
Figura 17 - Diagrama de Classe - Serviço Web (Web Service). 55
Figura 18 - Diagrama de Classe Aplicativo Smartphone. 56
Figura 19 - Modelo Entidade-Relacionamento. 57
Figura 20 - Work Breakdown Structure (WBS). 58
Figura 21 - Sequenciamento de Atividades. 59
Figura 22 - Arquitetura da Solução 60
Figura 23- IDE Eclipse 65
Figura 24 - Estrutura do Aplicativo Virtual Bartender 70

LISTA DE CÓDIGOS

Código 1 - Classe de Entidade. 62
Código 2 - Arquivo JSON. 63
Código 3 - Classe VO. 66
Código 4 - Classe DAO. 67
Código 5 - Classe BO. 68
Código 6 - Classe Rest. 68

LISTA DE ABREVIATURAS E SIGLAS

API – Application Programming Interface

APK – Android Package

CSS – Cascading Style Sheets

HTML – Hyper Text Markup Language

IPA – iOS Application Archive

JEE – Java Enterprise Edition

JS – JavaScript

JSON – JavaScript Object Notation

MVC – Model-View-Controller

SDK - Software Development Kit

SQL - Structure Query Language

YQL – Yahoo Query Language

XAP - Application Package

SUMÁRIO

1 – INTRODUÇÃO 14
1.1 – OBJETIVOS 16
1.2 – JUSTIFICATIVA 17
1.3 – PÚBLICO ALVO 18
1.4 - ESTRUTURA DO TRABALHO 18

2 - TECNOLOGIAS E FERRAMENTAS DE DESENVOLVIMENTO 20
2.1 - SENCHA TOUCH 20
2.1.1 – Licenciamento 21
2.1.2 - Principais características do Sencha Touch: 22
2.1.3 – Documentação do Sencha Touch 23
2.2 – PHONEGAP 25
2.2.1 - Pacote PhoneGap: 27
2.2.2 - Arquitetura Orientada a Dados 28
2.2.3 – Integração PhoneGap com Web Services 29
2.2.4 – PhoneGap Build 33
2.3 - HTML5 35
2.4 - CSS3 38
2.5 - JAVASCRIPT 38
2.6 - MAPA MENTAL 40
2.8 – UML 41
2.8.1 - Casos de uso 42
2.8.2 - Diagrama de atividades 42
2.8.3 - Diagrama de Sequência 43
2.8.4 - Diagrama de Classes 43
2.8.5 - Modelagem de Entidade e Relacionamento 43
2.9 - PLANEJAMENTO DE PROJETO 44
2.9.1 - Work Breakdown Structure – WBS 44
2.9.2 - Sequenciamento de Atividades 44
2.9.3 - Cronograma (estimativa de tempo) – Diagrama de Gantt 45
2.9.4 - Especificação de Recursos 45
2.9.5 - Especificação de Custos (orçamento) 45

3 - ANÁLISE E ESPECIFICAÇÃO DO SISTEMA 46
3.1 - MAPA MENTAL 46
3.2 – LISTA DE EVENTOS 47

3.3 - CASOS DE USO 48
3.4 - DIAGRAMA DE ATIVIDADES 51
3.5 - DIAGRAMA DE SEQUÊNCIAS 53
3.6 - DIAGRAMA DE CLASSES 54
3.7 - MODELAGEM DE ENTIDADE E RELACIONAMENTO 57

4 - ESTRUTURA DO PROJETO 58
4.1 - WORK BREAKDOWN STRUCTURE – WBS 58
4.2 - SEQUENCIAMENTO DE ATIVIDADES 59
4.3 – CRONOGRAMA 59

5 – IMPLEMENTAÇÃO DA SOLUÇÃO 60
5.1 – BASES DE DADOS 61
5.2 – PERSISTÊNCIA DOS DADOS 61
5.3 – MÓDULO DE INTEGRAÇÃO 63
5.4 – TIPOS DE CONEXÃO 64
5.5 – APLICAÇÃO WEB JEE 65
5.6 - APLICATIVO MÓVEL 69

6 – CONCLUSÃO 71

REFERÊNCIAS BIBLIOGRÁFICAS 72

1 – INTRODUÇÃO

O futuro da tecnologia está muito além da certeza, mas o que se pode afirmar é que a população vive uma era, onde os principais atrativos pessoais tecnológicos do mundo são os dispositivos móveis[1]. Os dispositivos mais modernos, como os famosos *tablets*[2] e *smartphones*[3], apresentam *software e hardware* capazes de fazer coisas fantásticas.

Esses dispositivos apresentam integração com muitas outras tecnologias como: Acelerômetro, Câmera, conectividade WI-FI, 3G *e Bluetooth*, GPS e serviços de Geolocalização, USB, armazenamento interno e externo com micro *SD Card*, tela sensível ao toque (*Touch Screen)*, reprodução de música, TV digital, entre outras. Com todas essas tecnologias embarcadas em um único aparelho, os *smartphones* tornam-se imprescindíveis na vida moderna (BLANC, OEHLMAN, 2012).

Em 2007 e 2008 dois grandes acontecimentos marcaram o início da era dos *smartphones*: o lançamento do *iPhone* pela *Apple* e o primeiro *smartphone* com o sistema operacional *Android* pela *Google*. Desde então, o desenvolvimento de aplicações para *smartphone* tem ganhado cada vez mais adeptos no mundo; formando grandes alianças, comunidades de partilha, de ideias e de entre ajuda (CLARK; JOHNSON, 2012).

Mesmo sendo uma área tecnológica ainda em seu estágio inicial, pois é muito recente, com apenas sete anos, o desenvolvimento de aplicativos para *smartphone* já é considerado muito amadurecido e bem desenvolvido para sua época.

Desenvolver para essa plataforma requer uma nova linha de estratégia e habilidade dos desenvolvedores, pois com esses aparelhos surgiram muitos novos desafios. No que diz respeito ao requisito de programação, suas características são muito diferentes das de um computador pessoal (PCs), tanto na arquitetura como nos recursos de *hardware*. Alguns dos grandes desafios encontrados dão-se no contexto de interface, devido às suas dimensões reduzidas.

---

[1] http://www.dimap.ufrn.br/~gold/intro.htm

[2] http://web.ist.utl.pt/joao.rodrigo/CM/?page_id=53

[3] http://www.tecmundo.com.br/celular/915-como-escolher-um-smartphone-.htm

Hoje em dia existem vários modelos de *smartphones,* cada um com suas características particulares, como:

- Sistemas operacionais *(Apple[4] - iOS, Canonical[5] - Ubuntu Phone, Google[6] – Android, Microsoft[7] - Windows Phone, Mozila[8] - Firefox OS e o Rim[9] - Blackberry OS);*
- Lojas de aplicativos (*Ovis tore*, *Google Play*, *Black Barry World*, *App Store*) entre outras;
- Plataformas e linguagens de desenvolvimento;
- *Design* e Modelo.

O fato de o desenvolvimento estar segmentado em plataformas (*iOS*, *Android*, *Windows Phone*, etc.) acaba por se tornar muito atraente, mas também requer uma certa experiência dos desenvolvedores em desenvolver aplicações para todas as plataformas.

No contexto de pesquisa encontramos dois tipos de linguagem de programação: as nativas e as híbridas. Em linguagens nativas, as aplicações são desenvolvidas para uma plataforma específica, como, para *Android* (utilizando Java), *iOS* (com *ObjectiveC*) e *Windows Phone* (com C Sharp), entre outras.

As grandes vantagens dessas aplicações são:

- Total acesso aos recursos dos dispositivos e *hardwares*;
- Armazenamento local de dados;
- Maior desempenho;
- Componentes projetados para a plataforma.

[4] http://www.apple.com/ios/

[5] http://www.ubuntu.com/phone

[6] https://www.google.com.br/mobile/android/

[7] http://www.windowsphone.com/pt-BR

[8] http://www.mozilla.org/pt-BR/firefox/os/

[9] http://br.blackberry.com/software/smartphones/blackberry-6-os.html

Já nas linguagens híbridas é possível escrever o código apenas uma vez, de forma que ele poderá ser interpretado em várias plataformas, usando as linguagens dos navegadores de internet (*browsers*), através dos *WebKit*[10]. Além disso, podemos transformar as aplicações desenvolvidas como forma nativa, sem ter a necessidade de reescrever todo o código para a plataforma de destino, pois o código é desenvolvido em linguagem de *script*, utilizando normalmente HTML, *JavaScript*, CSS e *frameworks*. No desenvolvimento híbrido são encontrados alguns *frameworks*[11], que portam as aplicações para multiplataformas. São exemplos: o *PhoneGap* (com HTML, *JavaScript* e CSS), *Xamarin* (utilizando *C Sharp*), *Sencha Touch* (HTML5) ou *Appacelerator* (*JavaScript*).

A portabilidade das apps (*termo utilizado para referir-se às aplicações móveis*) pode ser para: *Android, iOS, BlackBerry*, etc.

Em um contexto de crescimento explosivo no mercado dos *smartphones* surgem novos caminhos para todos os desenvolvedores. Cabe a cada um escolher sua melhor abordagem.

1.1 – OBJETIVOS

O Presente trabalho tem como objetivo principal, realizar análise e estudo sobre as tecnologias de desenvolvimento de *apps* móvel multiplataforma, utilizando a junção dos *frameworks, PhoneGap e Sencha Touch*.

Desta forma, espera-se aplicar os conhecimentos e práticas adquiridas nos estudos e análise, tendo como motivação, o desenvolvimento de um aplicativo móvel, o "*Virtual Bartender"* que fornecerá uma experiência de utilização, semelhante às aplicações desenvolvidas como sendo nativas. Além dos frameworks, serão abordadas as linguagens de *script Web*, tais como: HTML5, CSS3 e *JavaScript*, utilizadas como bases tecnológicas para o desenvolvimento do aplicativo

A aplicação será destinada ao auxílio de atendimento a bares. O aplicativo fará a integração com outro sistema, por meio de serviços web (*Web Service),* neste caso, pelo seu próprio

[10] http://www.webkit.org/

[11] http://www.dsc.ufcg.edu.br/~jacques/cursos/map/html/frame/oque.htm

aparelho *smartphone*, o cliente poderá fazer seu pedido, sem ter que esperar o atendimento do garçom.

1.2 – JUSTIFICATIVA

A justificativa para a abordagem de estudo de caso do aplicativo móvel multiplataforma, que será desenvolvido neste trabalho, se baseia no fato de os *frameworks*, *PhonGap e Sencha Touch*, ainda se encontrar em seu estágio inicial de evolução tecnológica. Pois pouco se sabe sobre esses *frameworks*, contudo, existem muitas vantagens ao se usar essas ferramentas de desenvolvimento. Em primeiro lugar, é bastante provável que numa empresa ou equipe de desenvolvimento, já haja um conhecimento aprofundado das tecnologias *Web* (HTML e *JavaScript*), ao contrário das tecnologias e linguagens nativas dos vários sistemas.

A segunda vantagem está de certa forma ligada à primeira, pois há a possibilidade de se integrar *toolkits* e outros *frameworks JavaScript*, já existentes, tal como *jQuery Mobile*, *Dojox Mobile*, e etc.

Outra vantagem é a arquitetura do *PhoneGap*, que possibilita a extensão de funcionalidades, para que uma aplicação tenha acesso a mais funcionalidades nativas, do que aquelas que estão presentes nas *API's*[12] atuais da *framework.*

Além disso, para o desenvolvimento do aplicativo terá um estudo aprofundado sobre a integração de várias tecnologias, pois estas terão que se comunicarem todas entre si para o seu funcionamento. O aplicativo desenvolvido poderá vir a ser utilizado nos bares e terá por finalidade, promover uma maior interação e facilidade na relação entre bar e clientes. Isso possibilitará que o bar receba rapidamente o pedido dos clientes, evitando enganos e reclamações sobre a qualidade do serviço, pois tanto o cliente quanto o funcionário do estabelecimento estarão acompanhando a movimentação e o consumo dos produtos.

O aplicativo ajudará a automatizar o estabelecimento e evitar problemas de se fazer tudo manualmente, pois em dias de grande movimentação em bares, chamar o garçom e fazer seu pedido pode ser uma tarefa difícil.

[12] http://www.lbd.dcc.ufmg.br/colecoes/semish/2009/007.pdf

1.3 – PÚBLICO ALVO

A Aplicação desenvolvida será destinada aos bares e estabelecimentos, que possam se interessar pelas vantagens trazidas pelo presente aplicativo.

Os estudos apresentados poderão contribuir com futuras pesquisas, para aqueles que venham a se atrair por essa abordagem de desenvolvimento multiplataforma para *smartphone.*

1.4 - ESTRUTURA DO TRABALHO

Este trabalho está estruturado em capítulos. O primeiro capítulo apresenta uma introdução sobre os dispositivos móveis *(smartphones e tablets)*; algumas funcionalidades; características; e tecnologias presentes nesses dispositivos. Também apresenta os objetivos, as justificativas e as principais tecnologias para o trabalho proposto.

No segundo capítulo são apresentadas as ferramentas e tecnologias que serão abordadas para o estudo, análise e desenvolvimento do trabalho.

O terceiro capítulo trás as análises e especificações do sistema, com os requisitos levantados para o desenvolvimento da proposta de trabalho. Esse capítulo também apresenta os principais diagramas UML (Casos de Uso, classe, sequência e atividade).

O quarto capítulo apresenta a metodologia de desenvolvimento utilizada para o trabalho, especificada em fases e etapas. Este capítulo define a *Work Breakdown Structure* (WBS), conhecida como estrutura analítica de trabalho. Define também, o diagrama de sequenciamento de atividades e o cronograma para acompanhamento das atividades realizadas.

O quinto capítulo apresenta a implementação da solução, nele é especificado todas as tecnologias que foram utilizadas para o desenvolvimento do estudo de caso dos frameworks *Sencha Touch* e *PhoneGap*, e também é apresentado o aplicativo desenvolvido com os frameworks, o *Virtual Bartender*.

O Sexto capítulo apresenta a conclusão do trabalho e o planejamento para os trabalhos futuros.

Por fim, o último capítulo trás as Referências Bibliográficas levantadas para o desenvolvimento do trabalho.

2 - TECNOLOGIAS E FERRAMENTAS DE DESENVOLVIMENTO

Neste capítulo serão apresentadas as ferramentas e tecnologias que serão abordadas para o estudo, análise e desenvolvimento do aplicativo multiplataforma "*Virtual Bartender"*.

O desenvolvimento do aplicativo terá como foco principal os *frameworks PhoneGap e Sencha Touch*. A programação será em códigos *JavaScript*, uma das principais linguagens para o desenvolvimento de aplicativos *Web mobile*, o *layout* com base em HTML5 e CSS3. Para a integração do aplicativo com o sistema de controle principal será criado uma aplicação JEE[13] *Web Service* (Serviços de Web). O servidor de aplicação JEE será o Apache Tomcat[14].

2.1 - SENCHA TOUCH

Sencha Touch[15] é um *framework* MVC[16] *JavaScript* especialmente projetado para criar aplicativos *web mobile*, para funcionar em dispositivos com telas sensíveis ao toque (*touchscreen)*. *Sencha Touch* permite aos desenvolvedores criar aplicativos para plataformas móveis que possuem navegadores de aplicação das normas mais recentes, como *WebKit browser engine*[17] (KOSMACZEWSKI**,** 2013).

Em 2009, a ascensão dos *smartphones touchscreen*, e, mais tarde, o *iPad*, levou o *ExtJS team*[18] a criar um *framework* voltado exclusivamente para estes novos dispositivos. O resultado de seus esforços foi o *Sencha Touch*, lançado na versão 1.0, no final de 2010 (KOSMACZEWSKI**,** 2013).

---

[13] http://www.oracle.com/technetwork/java/javaee/overview/index.html

[14] http://tomcat.apache.org/

[15] http://www.sencha.com/products/touch

[16] http://www.macoratti.net/vbn_mvc.htm

[17] http://www.webkit.org/

[18] http://www.sencha.com/company/team/

A primeira versão do *Sencha Touch* não era totalmente compatível com a versão mais recente do *Ext JS*, e também foi criticada por seus testes (*benchmarks)* de desempenho relativamente baixo, especialmente em dispositivos antigos, como o *iPhone 3G*. Para tratar dessas questões, *Sencha Touch 2* foi lançado em março de 2012, proporcionando um mecanismo totalmente novo, baseado 100% em CSS (*Cascading Style Sheets*), e um novo sistema de classes compatível com *Ext JS4* (KOSMACZEWSKI**,** 2013).

2.1.1 – Licenciamento

O *Sencha Touch* está disponível sob um regime de licenciamento bastante complexo, os desenvolvedores podem usar o *framework* sob licenças open source e comerciais grátis, ou uma licença comercial paga.

Em projetos de código aberto:

- Se você pretende distribuir a aplicação, divulgando totalmente o código fonte, existe uma versão do *Sencha Touch* distribuído através da licença GPLv3[19];
- Se você não quiser usar a licença GPLv3, há uma Licença *free* e *Open Source*[20].

Em projetos comerciais:

- Você pode usar o *Sencha Touch free*[21], sem qualquer custo, seja por aplicação, por usuário ou por desenvolvedor;
- Para aplicações embarcadas, você pode usar *Sencha Touch* gratuitamente até 5.000 instalações;
- Finalmente, uma licença OEM comercial está disponível, bem como, para as empresas dispostas a distribuir *Sencha Touch*, como parte de suas próprias aplicações ou serviços comerciais.

---

[19] http://gplv3.fsf.org/

[20] http://opensource.org/

[21] http://www.sencha.com/products/touch/download/

Sencha Touch também é distribuído e licenciado como parte do pacote de *"Sencha Complete*[22]*"*, que inclui o seguinte:

- Licenças de desenvolvedor *Sencha Touch e Ext JS;*
- Gráficos *Sencha*;
- Plug-in do Eclipse *Sencha;*
- *Sencha* Cmd;
- Tickets de Suporte;
- A licença de uso do *Sencha Architect*, uma ferramenta de design visual do aplicativo.

2.1.2 - Principais características do Sencha Touch:

- Grande biblioteca de pequenos aplicativos com funcionalidades específicas (*Widget UI)*, que tem a sua maior parte inspirada pelo *iOS*, tanto em *design*, quanto em funcionalidade;
- Arquitetura bem definida, reforçando a arquitetura MVC desde o início;
- Conectores internos em seu código para serviços de dados de rede, como Transferência de Estado Representativo (*Representational State Transfer*) ou somente (REST) de serviços web e suporte para aplicações *web mobile off-line*;
- Mecanismo de carregamento de classe avançada, reforçando as diretrizes de arquitetura MVC;
- *Sencha Touch* oferece suporte completo AJAX[23], incluindo CORS[24] e JSON-P[25];

[22] https://www.sencha.com/store/sencha-complete

[23] http://www.pontodatecnologia.com.br/2006/08/desvendando-o-ajax.html

[24] http://www.israelaece.com/post/Introducao-ao-CORS.aspx

[25] http://www.devmedia.com.br/introducao-ao-formato-json/25275

- Um sistema de linha de comando para construção, gestão de fusão e minimização do código das aplicações, bem como a construção de aplicativos nativos para *Android e iOS*;
- Vasta documentação, disponível como um conjunto de páginas HTML dinâmico, incluindo busca e filtragem de recursos sem a necessidade de qualquer infraestrutura de servidor.

A maioria dos *frameworks*, como o *Sencha Touch*, são construídos em torno de programação orientada a objeto (OOP) [26].

O *framework* também possui elementos da interface do usuário, que fornecem tamanhos de telas maiores, para os dispositivos móveis (tais como os dispositivos *tablets*).

Ele também possui um excelente suporte para eventos de toques, portanto, é uma ótima opção para construir aplicativos orientados a toques e *canvas*[27] (OEHLMAN; BLANC, 2012).

2.1.3 – Documentação do Sencha Touch

A *Sencha Touch* Interface de Programação de Aplicativo (*Application Programming Interface)* ou API fornece informações detalhadas sobre cada uma das classes de objeto e está disponível no site da *Sencha Touch*. Cada classe na API inclui documentação detalhada para cada opção de configuração, propriedade, método e evento. A API também inclui breves exemplos e outras informações úteis.[28]

O site *Sencha* também inclui uma série de exemplos de aplicações. O mais útil deles é a aplicação *Kitchen Sink*.

---

[26] http://www.hardware.com.br/artigos/programacao-orientada-objetos/

[27] http://www.w3schools.com/html/html5_canvas.asp

[28] http://docs.sencha.com/touch/2.3.1/

A aplicação *Kitchen Sink* fornece exemplos para:

- Item da interface dos usuários como: botões, formulários, barras de ferramentas, listas e outros;
- Animações para mudança de páginas;
- Eventos de toque;
- Manipulação de dados para JSON, YQL[29] e AJAX;
- Suportes para áudio e vídeo;
- Temas para mudar o visual do seu aplicativo.

Cada exemplo tem um botão *Source* no canto superior direito, que irá exibir o código para o exemplo atual.

Aplicações *Web* funcionam bem em todos os lugares. Mas ainda existem alguns recursos disponíveis exclusivamente para aplicativos nativos, que são essenciais para desenvolvedores de aplicativos. *Sencha Touch* atualmente suporta *Apache Cordova APIs*[30] para Acelerômetro, Câmera, Captura, Conexão, Contatos, Dispositivo, Eventos, Arquivo, Geolocalização, Mídias, Comunicação e Armazenamento. *Sencha Touch* também oferece suporte integrado com o *Adobe PhoneGap* para construir dentro do *Sencha Touch* comandos que podem portar seus aplicativos para vários dispositivos com um único comando.

Sencha Touch 2.3 oferece uma grande variedade de temas e *layout* para *iOS*, *Android*, *BlackBerry e Windows Phone*. Você pode portar as aplicações para qualquer plataforma e aplicar temas específicos da plataforma usando o recurso de comutação, tema da *Sencha Touch*.

[29] http://yuilibrary.com/yui/docs/yql/

[30] http://cordova.apache.org/docs/en/2.4.0/

2.2 – PHONEGAP

PhoneGap também conhecido como *Cordova*[31], é um *framework* de desenvolvimento móvel *open-source.* Com ele é possível construir aplicações móveis nativas para várias plataformas de dispositivos móveis, utilizando como padrão, tecnologias *Web* como HTML5, CSS3 e JS.[32]

O PhoneGap foi originalmente desenvolvido pela Nitobi, empresa adquirida pela Adobe[33] em 2011. Depois que foi adquirida, a Nitobi doou o código fonte do *PhoneGap* à *Apache Software Foundation*[34] (ASF), com o nome de projeto *Cordova*, que é o nome de uma rua em *Vancouver*, onde os escritórios da Nitobi foram localizados e a empresa criou a primeira versão do *framework* (NATILI, 2013).

Os aplicativos desenvolvidos usando o *PhoneGap* são aplicações híbridas. Partes do aplicativo, principalmente, a interface do usuário, a lógica da aplicação e comunicação com um servidor, é com base em HTML, JS e CSS. A outra parte, que se comunica com sistema operacional do dispositivo (*smartphone ou tablet*) é baseada no idioma nativo de cada plataforma. O *PhoneGap* fornece uma ponte entre o mundo *JavaScript* e o mundo das plataformas nativas.

A Figura 1 ilustra a comunicação de um aplicativo desenvolvido utilizando linguagens de script (HTML, CSS e JS) com os recursos do dispositivo. O *PhoneGap* API permite que seja realizado o acesso às funcionalidades do sistema operacional. Com o *JavaScript* e a API do *PhoneGap* constrói-se a lógica e a comunicação, com os componentes nativos como: câmera, GPS, Geolocalização, agenda, entre outros.

---

[31] https://cordova.apache.org/

[32] http://phonegap.com/

[33] http://www.adobe.com/

[34] http://www.apache.org/

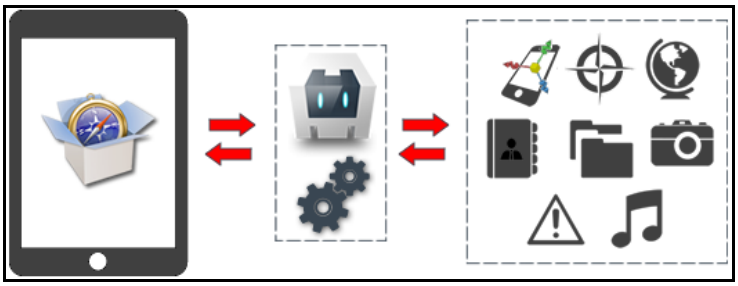

Figura 1 - Comunicação dos Componentes.[35]

A figura 2 ilustra a arquitetura do PhoneGap baseado em camadas.

A primeira é basicamente a camada de Interface do usuário, ou seja a parte visual de uma aplicação que é apresentada ao usuário. Essa camada é desenvolvida em linguagens de script.

A segunda camada é composta pelas bibliotecas JavaScript, responsável pelo acesso aos recursos do dispositivo. O PhoneGap Também tem um componente nativo, que trabalha por trás das aplicações acessando os recursos dos dispositivos.

A terceira camada são os recursos nativos do dispositivo, que são acessados pelo PhoneGap API. Por exemplo, quando você tirar uma foto e desejar enviá-la para o Facebook, você chamaria o PhoneGap câmera API para obter acesso a câmera do telefone, tirar uma foto, e enviar o arquivo.

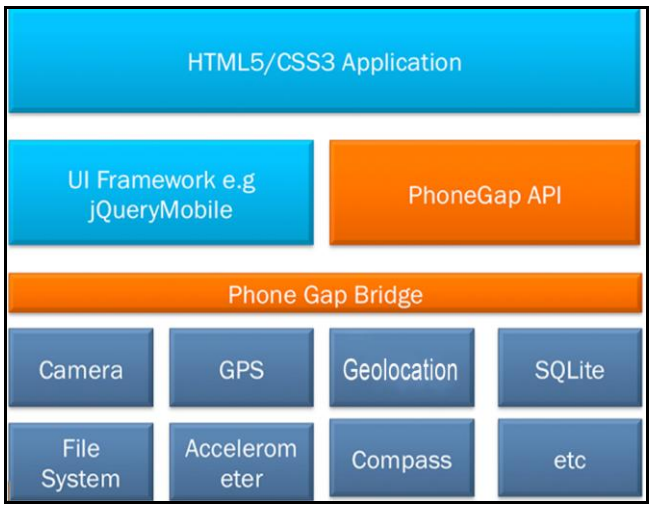

Figura 2 - Arquitetura do PhoneGap (In: GATOL; PATEL,2012).

[35] http://www.mobiltec.com.br/blog/index.php/projetando-um-sistema-multiplataforma-baseado-em-phonegap/

O *PhoneGap* API tem acesso a vários recursos dos sistemas operacionais dos *smartphones*. Ele pode variar de acordo com a plataforma, sendo ela *iOS, Android ou Windows Phone*. Mas na grande maioria, ele tem acesso aos seguintes recursos:

- Acelerômetro: Obtém informações do sensor de movimento do dispositivo;
- Câmera: Tira uma foto usando a câmera do dispositivo;
- Captura: Captura de áudio ou vídeo;
- Bússola: Obtém a direção que o dispositivo está a apontar;
- Conexão: Verifica o status de rede *WI-FI ou 3G;*
- Contatos: Trabalha com o banco de dados de contato a bordo;
- Dispositivo: Reúne informações específicas do dispositivo;
- Eventos: Escuta para eventos nativos do dispositivo;
- Arquivo: Lê e escreve para o sistema de arquivos nativos;
- Geolocalização: Reúne informações sobre a localização, mais detalhada;
- Mídia: Reproduz arquivos de áudio;
- Notificação: Cria notificações do dispositivo;
- Armazenamento: Armazena os dados diretamente no dispositivo.

2.2.1 - Pacote PhoneGap:

O *PhoneGap* utiliza as tecnologias *web* para o seu desenvolvimento. No entanto, o produto final nem sempre é *web*, pois a aplicação final pode ser um arquivo binário distribuído através dos ecossistemas das plataformas móveis.

Nas aplicações *iOS*, a saída é um arquivo IPA (*iOS Application Archive*), para as aplicações *Android* a saída será um arquivo APK (*Android Package*) e para o *Windows Phone* será uma

saída XAP (*Application Package*). Os formatos gerados pelo *framework* são os mesmos formatos de pacotes usados por aplicativos "nativos".

A figura 3 ilustra o processo de criação de um aplicativo desenvolvido com o *PhoneGap*. Inicialmente o aplicativo é desenvolvido com as tecnologias HTML5, CSS3 e JS; podendo ser executada através dos navegadores dos *smartphones*. O *PhoneGap* é o responsável por criar o arquivo binário, que é executado nas plataformas como sendo um aplicativo nativo tendo total acesso aos recursos de cada plataforma.

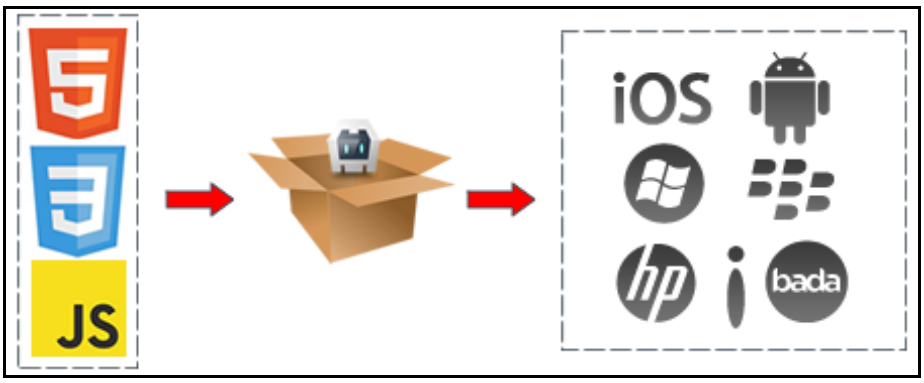

Figura 3 - Pacote PhoneGap.[36]

2.2.2 - Arquitetura Orientada a Dados

Um aplicativo criado pelo *PhoneGap* atua como um cliente. Para o usuário interagir com ele, o "cliente" *PhoneGap* se comunica com um servidor de aplicativo, para receber ou enviar dados.

A figura 4 mostra o processo de comunicação entre um aplicativo *PhoneGap* e um Banco de dados.

O usuário solicita algum dado ou informação na interface de usuário. O servidor deve executar a lógica de negócios, cálculos e recuperar dados de um repositório de dados

[36] http://www.mobiltec.com.br/blog/index.php/projetando-um-sistema-multiplataforma-baseado-em-phonegap/2-7/

separado. Geralmente o servidor é *web* como: *Apache Tom Cat*[37], IIS[38] entre outros, e tem do seu lado linguagens como *ColdFusion*[39], *Java*[40], *NET*[41], PHP[42], entre outras. Após processado os dados são retornados para a interface do usuário, de acordo com a solicitação.

As arquiteturas de aplicações específicas costumam variar de caso a caso, no entanto, a maioria das aplicações orientadas a dados empregam esse tipo de arquitetura.

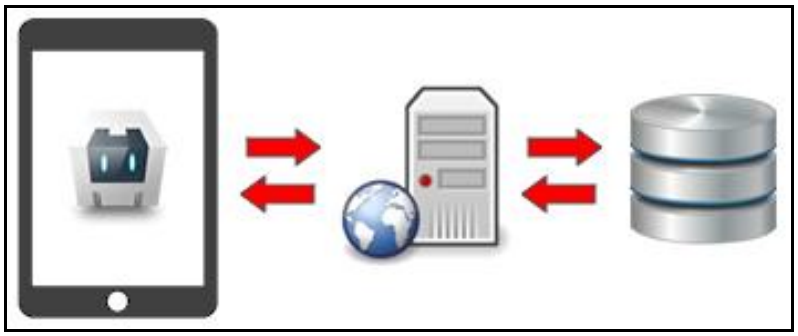

Figura 4 - Arquitetura Orientada a Dados[43].

2.2.3 – Integração PhoneGap com Web Services

Tal como o nome sugere *Web Service ou* serviços Web é um tipo de aplicação *webified* (aplicativo que usa a web o tempo todo ou que tem uma interface Web opcional, quando os usuários não estão em suas máquinas), ou seja, uma aplicação tipicamente entregue sobre HTTP (*Hyper Text Transport Protocol*) (KALIN, 2013).

[37] http://tomcat.apache.org/

[38] http://technet.microsoft.com/pt-br/library/cc753433(v=ws.10).aspx

[39] http://www.adobe.com/br/products/coldfusion-family.html

[40] http://www.oracle.com/technetwork/pt/java/javase/downloads/index.html?ssSourceSiteId=otnes

[41] http://msdn.microsoft.com/pt-br/vstudio/aa496123.aspx

[42] http://www.php.net/

[43] http://www.mobiltec.com.br/blog/index.php/projetando-um-sistema-multiplataforma-baseado-em-phonegap/

Serviços Web são, portanto, uma aplicação distribuída, cujos componentes podem ser utilizados e executados em dispositivos distintos. Serviços Web podem consistir de vários componentes de código, cada um hospedado em um servidor de nível empresarial separado. O serviço web pode ser consumido tanto em PCs como em dispositivos moveis (KALIN, 2013).

As aplicações *PhoneGap* geralmente não se comunicam diretamente com um banco, pois os dados são encaminhados através de um servidor de aplicativos. O cliente realiza a comunicação com o servidor, a partir de solicitações HTTP padrão, para conteúdo de HTML, XML serviços RESTfull, serviços JSON, ou SOAP. Estas são as mesmas técnicas que são utilizadas em aplicações *web Desktop* baseados em AJAX.

Na arquitetura ao lado do cliente é utilizado um modelo de aplicação de uma única página, a qual a lógica de aplicação está dentro da página HTML. Esta página não é descarregada da memória. Todos os dados deverão ser exibidos por atualização do HTML DOM, enquanto serão recuperados do servidor, utilizando técnicas de AJAX e as variáveis serão mantidas na memória dentro do *JavaScript.*

Clientes Multi-page podem ser arquitetados e suportados pelo *PhoneGap*, mas não são recomendados pelo próprio *framework*, pois o sistema pode perder variáveis em memória ao carregar uma nova página separada.

A Figura 5 mostra um exemplo de uma aplicação *web* básica com três camadas. Seria a *Interface*, a lógica de negócio junto a uma camada de padrões como SOAP[44] ou *Restful*[45], para que possamos realizar futuramente uma fácil comunicação com o projeto móvel e a DAO, a qual é utilizada para buscar dados no banco de dados, na persistência de um exemplo de Banco de Dados relacional. Com esses componentes definidos temos um sistema *web* completo.

[44] http://www.linhadecodigo.com.br/artigo/38/soap-e-webservices.aspx

[45] http://www.infoq.com/br/articles/rest-introduction

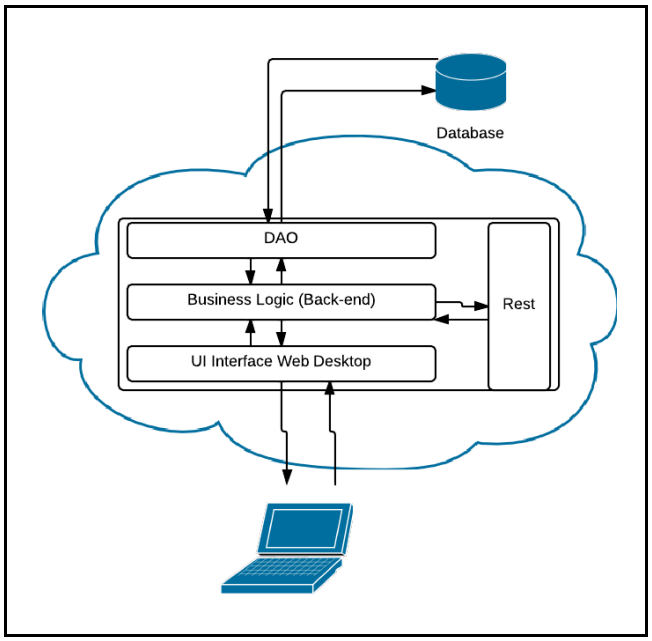

Figura 5 - Sistema Baseado em Camadas.[46]

A Figura 6 ilustra as camadas básicas de um sistema móvel, a primeira camada é basicamente a camada de usuário, essa interface se comunica com a camada de lógica de negócio local desenvolvida com *JavaScript* (é necessário conter regras de negócio no dispositivo, como cálculos e regras do sistema, pois o sistema pode funcionar mesmo sem conexão à internet). A regra de negócio móvel *(Business Logic Mobile)* é responsável por comunicar-se com a API do *PhoneGap*, que tem como responsabilidade fazer a comunicação com os componentes nativos e o banco interno dos smartphones SQL Lite[47].

[46] http://www.mobiltec.com.br/blog/index.php/projetando-um-sistema-multiplataforma-baseado-em-phonegap/

[47] http://www.sqlite.org/

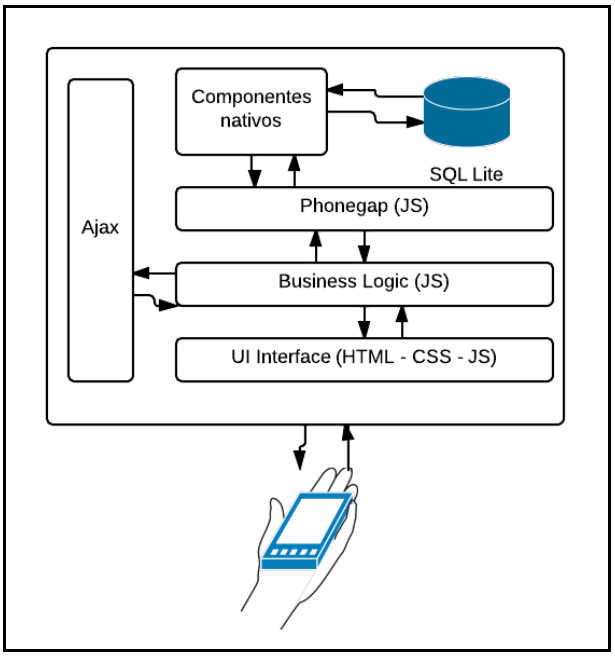

Figura 6 - Modelo de Camadas Mobile.[48]

Com base nos exemplos mostrados anteriormente nas figuras 5 e 6 é possível criar uma integração de dispositivos distintos, compartilhando os mesmos dados através dos serviços *web*.

A Figura 7 ilustra a integração de dois dispositivos se comunicando através de serviços *Web*. Observe que quando o usuário faz uma solicitação no aplicativo do *smartphone*, o processo se inicia na camada de interface do usuário. A lógica e as regras de negócio são realizadas através da camada *JavaScript* e enviadas ao banco local do aparelho, geralmente um SQLlite. Isso possibilita que os aplicativos possam trabalhar off-line mesmo que não haja conexão à Internet, e sem correr o risco de perder informações; possibilitando que essas informações possam ser enviadas assim que houver uma conexão disponível. Logo após os dados serem enviadas ao banco local, ele retorna para a camada de lógica de negócio, onde essa por sua vez, fica responsável pelo envio dos dados para a camada AJAX, que se comunica com a camada *Rest* da aplicação de serviços web. Essas duas camadas são as responsáveis por fazer a integração dos dispositivos. Observe também que o banco de dados não está localizado em nenhuma das aplicações, geralmente fica em um servidor separado e isso possibilita um dos

[48] http://www.mobiltec.com.br/blog/index.php/projetando-um-sistema-multiplataforma-baseado-em-phonegap/

fundamentos dos serviços *web* que é o consumo e produção dos dados em qualquer dispositivo, seja PCs ou dispositivos móveis.

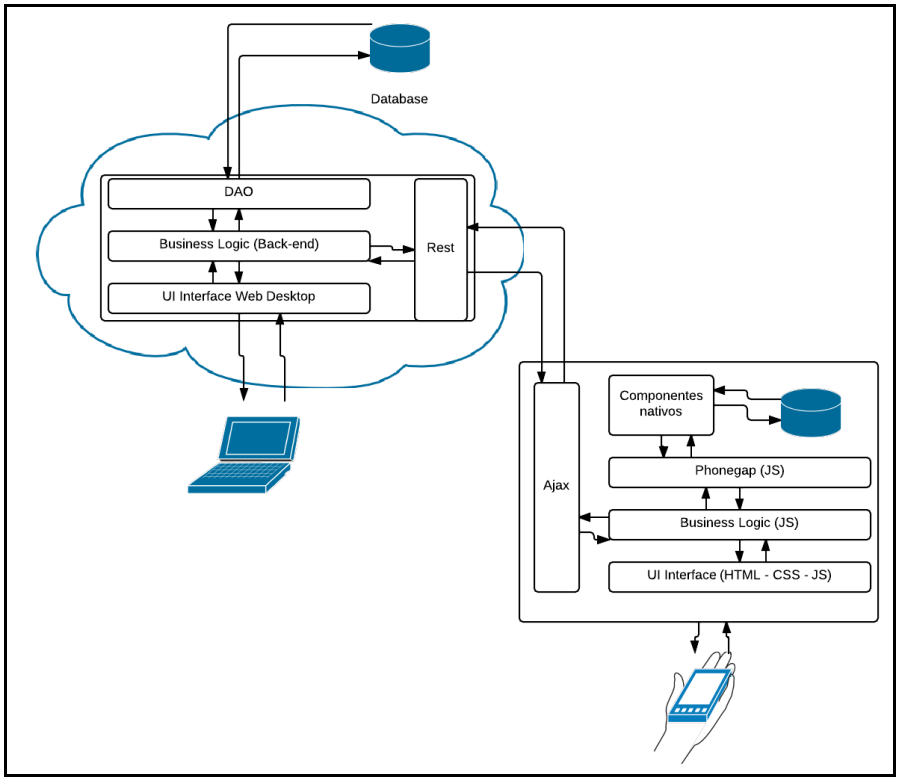

Figura 7 - Comunicação entre Dispositivos.[49]

2.2.4 – PhoneGap Build

As aplicações criadas com *PhoneGap* podem ser distribuídas para várias lojas de aplicativos (*Ovis tore, Google Play, Black Barry World,* e *App Store)* e serem instaladas no dispositivo de um usuário final, como qualquer outro aplicativo nativo.

Mesmo que o *PhoneGap* forneça uma maneira intuitiva e amigável para a criação das *apps* móveis, os ambientes de desenvolvimento móvel ainda requer um conhecimento técnico avançado.

[49] http://www.mobiltec.com.br/blog/index.php/projetando-um-sistema-multiplataforma-baseado-em-phonegap/

Cada fornecedor oferece um conjunto de ferramentas diferentes, e a cada lançamento o *PhoneGap* é compatível com um conjunto específico de ferramentas.

PhoneGap Build[50] é um serviço de nuvem que permite que aplicações desenvolvidas com tecnologias (HTML5, CSS3 e JS) passem a ser uma aplicação específica de cada sistema operacional, como, *Android, iOS, Ubuntu Phone, Windows Phone, Firefox OS, BlackBerry OS,* entre outras. Isso possibilita que a aplicação possa ser disponibilizada nas lojas de cada plataforma. O *PhoneGap Build* é projetado para tornar o desenvolvimento de aplicações móveis mais rápido e mais fácil. Tudo o que se precisa fazer é o *upload* do projeto desenvolvido com o *PhoneGap,* e a aplicação estará pronta para o mercado, em diferentes plataformas.

O *PhoneGap Build* também possibilita que os aplicativos possam ser baixados diretamente do site , sem ter a necessidade de enviá-los para as lojas.

A figura 8 ilustra o processo de transformação de uma aplicação web para um aplicativo nativo de uma plataforma especifica, para *smartphone* (*iOS, Android, Windows Phone e etc.*).

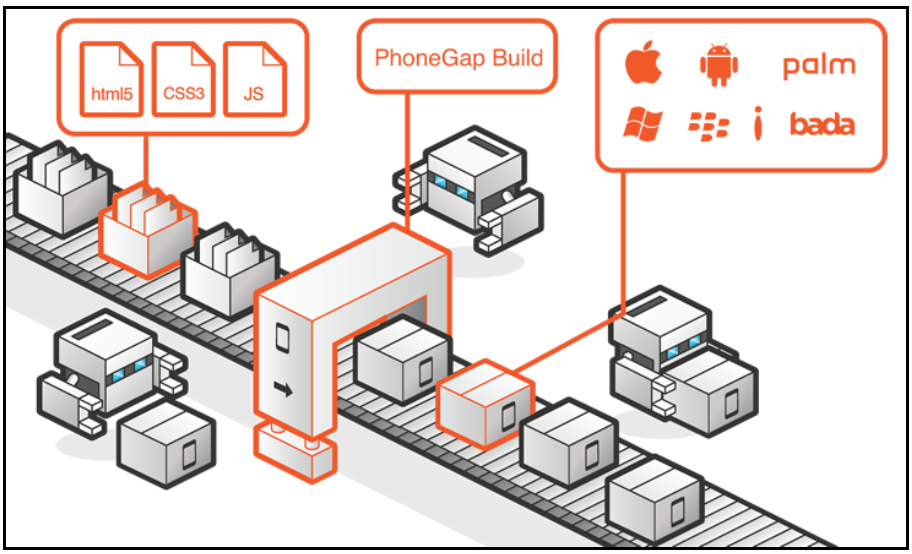

Figura 8 – PhoneGap Build.[51]

[50] https://build.phonegap.com/

[51] https://build.phonegap.com/

2.3 - HTML5

De acordo com o W3C[52] a *Web* é baseada em três pilares:

- Um esquema de nomes para localização de fontes de informação na *Web*. Esse esquema chama-se URL;
- Um Protocolo de acesso para acessar estas fontes, o HTTP[53];
- Uma linguagem de *Hypertexto*, para uma fácil navegação entre as fontes de informação: o HTML.

HTML é uma abreviação de *Hypertext Markup Language* - Linguagem de Marcação de *Hypertexto*. Resumindo em uma frase: o HTML é uma linguagem para publicação de conteúdo (texto, imagem, vídeo, áudio e etc.) na *Web* (FERREIRA, EIS, 2014).

O HTML baseou-se no *Standard Generalized Markup Language (*SGML)[54], Linguagem de Marcação Generalizada Padrão, um sistema de processamento de documentos bem maior.

Desenvolvido originalmente por *Tim Berners-Lee* o HTML ganhou popularidade quando o *Mosaic* (*browser* desenvolvido por *Marc Andreessen* na década de 1990) ganhou força. A partir daí, desenvolvedores e fabricantes de *browsers* utilizaram o HTML como base, compartilhando as mesmas convenções.

O HTML é baseado no conceito de *Hipertexto*, que são conjuntos de elementos, ou nós, ligados por conexões. Estes elementos podem ser palavras, imagens, vídeos, áudios, documentos, e etc. Conectados, eles formam uma grande rede de informação.

Segundo Silva (2011), podemos resumir *hipertexto*, como sendo todo o conteúdo inserido em um documento para a *web*, e que tem como principal característica a possibilidade de se interligar a outros documentos da *web*. O que torna possível a construção de *hipertextos* são os *links*, presentes nas páginas dos *sites* que estamos acostumados a visitar, quando entramos na *internet*.

---

[52] http://www.w3.org/standards/webdesign/htmlcss

[53] http://imasters.com.br/artigo/11513/redes-e-servidores/afinal-o-que-e-http/

[54] http://codigosdown.wordpress.com/2011/11/25/sgml-xml/

Resumindo em uma frase: o HTML é uma linguagem para publicação de conteúdo (texto, imagem, vídeo, áudio e etc.) na *Web*.

Desde a invenção da *web* por *Tim Berners-Lee*, a HTML evoluiu por sete versões que são:

- HTML;
- HTML +;
- HTML 2.0;
- HTML 3.0;
- HTML 3.2;
- HTML 4.0;
- HTML 4.01 ;
- HTML5 .

HTML5[55] em muitos aspectos representa o melhor de todos os *Worlds*, com uma grande quantidade de novas potencialidades jogadas em boa medida (CLARK; STUDHOLME; MURPHY; MANIAN, 2012).

Em maio de 2007, o W3C reconsiderou sua decisão de encerrar o desenvolvimento da HTML em favor da XHTML e tornou pública sua decisão de retomar os estudos para o desenvolvimento da HTML5, tomando como base o trabalho que já vinha sendo desenvolvido pelo WHATWG.

No dia 19 de janeiro de 2011, *Ian Hickson*, editor da HTML5, publicou no blog da WHATWG[56], uma matéria informando que a especificação para a HTML5 continuaria a ser desenvolvida exclusivamente pelo W3C, ficando sob-responsabilidade do WHATWG a continuidade do desenvolvimento.

A especificação para a HTML5 está estruturada em 10 sessões, a saber:

1. Infraestrutura comum: define terminologia, classes, algoritmos, sintaxes e partes comuns das especificações;

[55] http://www.w3schools.com/html/html5_intro.asp

[56] http://blog.whatwg.org/

2. Semântica, estrutura e APIs para documentos HTML: definem as funcionalidades do DOM[57] HTML e dos elementos HTML em geral;

3. Elementos HTML: explicam o significado de cada um dos elementos HTML. São estabelecidas regras de uso dos elementos na marcação, bem como diretrizes de manipulação deles pelos agentes de usuário;

4. Microformatos: a especificação para a HTML5 prevê um mecanismo para marcar informações sobre o documento, no formato nome/valor, para serem lidas por máquinas. Essa sessão descreve esse mecanismo e os algoritmos capazes de converter documentos HTML em outros formatos. Adicionalmente, nessa sessão definem-se alguns vocabulários para Microformatos: informações de contato, calendário de eventos e licenças;

5. Carregamento de páginas *web*: documentos HTML não aparecem do nada, essa sessão define as muitas funcionalidades relacionadas ao tratamento de páginas *web* pelos diferentes dispositivos.

6. APIs para aplicações *web*: descrevem as funcionalidades básicas para desenvolvimento de *scripts* em aplicações HTML;

7. Interação com o usuário: descreve os diferentes mecanismos de interação do usuário com um documento HTML;

8. APIs para comunicação: tratam dos mecanismos de comunicação entre aplicações HTML rodando em domínios diferentes e no mesmo cliente;

9. Sintaxe HTML: descreve a sintaxe HTML;

10. Sintaxe XHTML: descreve a sintaxe XHTML.

A especificação para a HTML5 iniciada no ano de 2004 e incorporada pelo W3C no ano de 2007 tem como objetivo geral estudar e resolver problemas relacionados à implementação de um HTML contemporâneo, e ao mesmo tempo compatível com conteúdos existentes. Para cumprir esse objetivo, considera os seguintes pontos:

- Definição de uma linguagem única denominada HTML5 que pode ser escrita tanto com a sintaxe HTML quanto com a sintaxe XML;

[57] http://www.w3c.br/cursos/html5/conteudo/capitulo16.html

- Definição de modelos de processamento detalhados com vista a promover a interoperabilidade da linguagem;
- Aperfeiçoamento da marcação dos documentos;
- Criação de marcação e APIs para novas tecnologias, tais como as aplicações *Web*.

2.4 - CSS3

CSS é a abreviação para o termo em inglês *Cascading Style Sheet*, traduzido para o português como folhas de estilo em cascata (SILVA, 2011).

Em setembro de 1994, surgiu a primeira proposta para implementação das CSS. Até então, o próprio Tim Berners-Lee, considerava que a estilização era uma questão a ser resolvida pelo navegador, razão pela qual não se preocupou em publicar a sintaxe usada para criar a folha de estilo padrão do seu navegador. No mês de dezembro de 1996, as CSS1 foram lançadas como uma Recomendação oficial do W3C (SILVA, 2011).

A definição mais precisa e simples para folha de estilo encontra-se na *homepage* das CSS no site do W3C e diz:

"Folha de estilo em cascata é um mecanismo simples para adicionar estilos (por exemplo: fontes, cores, espaçamentos) aos documentos web".

O CSS formata a informação entregue pelo HTML. Essa informação pode ser qualquer coisa: imagem, texto, vídeo, áudio ou qualquer outro elemento criado. Essa formatação na maioria das vezes é visual, mas não necessariamente.

2.5 - JAVASCRIPT

JavaScript[58] também conhecido como JS, foi criado pela *Netscape*[59] em parceria com a *Sun Microsystems*[60], com a finalidade de fornecer um meio de adicionar interatividade a uma

[58] http://www.w3schools.com/js/DEFAULT.asp

página *web*. A primeira versão, denominada *JavaScript* 1.0, foi lançada em 1995 e implementada em março de 1996 no navegador Netscape *Navigator* 2.0 quando o mercado era dominado pela *Netscape* (Silva, 2010).

JavaScript é parte da tríade de tecnologias que todos os desenvolvedores *Web* devem aprender: HTML para especificar o conteúdo de páginas da *web*, CSS para especificar a apresentação de páginas e *JavaScript*, para especificar o comportamento das páginas (FLANAGAN, 2011).

O nome *"JavaScript"* na verdade é um pouco enganador. Com exceção de uma sintática superficial semelhante, ele é completamente diferente da linguagem de programação Java, e é há muito tempo já superadas suas raízes no idioma de script, para tornar-se uma linguagem de propósito geral, robusta e eficiente. A mais recente versão da linguagem define novos recursos para desenvolvimento de software em grande escala (FLANAGAN, 2011).

Segundo Silva, Osmar J (2000), *JavaScript* é uma linguagem de programação usada por 99% dos *Webmasters* e desenvolvedores de páginas *web* em geral. Como toda linguagem de programação, o *JavaScript* obedece a uma série de convenções. Ele é uma linguagem próxima do *Delphi, Visual Basic* e C++, mas ainda assim completamente diferente na maneira de lidar com suas características e resultados.

De Acordo com Silva (2010) *JavaScript* é uma linguagem desenvolvida para executar no lado do cliente, isto é, a interpretação e o funcionamento da linguagem dependem de funcionalidades hospedadas no navegador do usuário. Isso é possível porque existe um interpretador *JavaScript* hospedado no navegador.

JavaScript é uma linguagem de *script*, que você pode adicionar a uma página HTML para torná-la mais interativa e conveniente para o usuário. Por exemplo, você pode escrever um código *JavaScript* que inspecione os valores digitados em um formulário, para garantir que sejam válidos, ou pode fazer um código *JavaScript*, para mostrar ou esconder elementos de uma página, dependendo de onde o usuário selecione. O *JavaScript* pode até contatar o

---

[59] http://channels.netscape.com/forum_center/?404=http%3a%2f%2fcommunity.netscape.com%2fn%2fpfx%2fforum.aspx%3ftsn%3d1%26nav%3dmessages%26webtag%3dws-nscpbrowser%26tid%3d10422

[60] http://www.oracle.com/us/sun/index.html

servidor *web* para executar alterações no banco de dados, sem atualizar a página *web* atual (STARK, 2012).

A linguagem *JavaScript* suporta programação orientada a objetos – *Object-Oriented Programming* (OOP). É mais apropriado dizer que *JavaScript* é uma linguagem capaz de simular muitos dos fundamentos de OOP, embora não plenamente alinhada com todos os conceitos de orientação a objetos (SILVA, 2010).

Em 2010, muitos dispositivos móveis em massa no mercado passaram a aceitar o ECMAScript-MP[61] ou o *JavaScript* móvel. O *JavaScript* móvel é uma ferramenta fantástica para criar experiências *Web Mobile* interativas. Testar o *JavaScript* em dispositivos móveis é fundamental para o desenvolvimento eficiente, pois os emuladores podem não descobrir os problemas de sintaxe e performance.

O *JavaScript* móvel reduz os conjuntos de caracteres suportados e exclui os elementos da linguagem que requer muita computação. Difere de seu correspondente do desktop no nível de seu DOM e suporte de eventos no navegador móvel. O DOM e o suporte de eventos podem variar entre os revendedores do navegador e versão.

O *Script* no lado do cliente também pode reduzir o desempenho da navegação *Web Mobile.* Os usuários de dispositivos móveis podem desativar a execução *JavaScript.* Por isso, até a marcação designada para os dispositivos móveis que suportam o *JavaScript* deve adaptar-se com elegância a um ambiente sem *script*. O *desing* flexível da *Web Mobile* programa primeiro a marcação e a aperfeiçoa interativamente com *script* no lado do cliente.

2.6 - MAPA MENTAL

O surgimento da técnica de Mapa Mental ocorreu na década de 1960, e foi proposto pelo psicólogo e escritor inglês Tony Buzan.

Mapa mental, ou mapa da mente é o nome dado para uma técnica de organizar pensamentos e ideias, construindo um tipo de diagrama (SOMMERVILLE, 2003).

[61] http://technical.openmobilealliance.org/Technical/release_program/docs/Browsing/V2_3-20050118-C/OMA-WAP-ESMP-V1_0-20040709-C.pdf

De acordo com Buzan (2005), o Mapa Mental é um poderoso método que visa auxiliar no armazenamento, organização, e priorização de informações. O método consiste em utilizar palavras e imagens chave, para facilitar a lembrança de ideias específicas.

Segundo Buzan (2005), o nosso cérebro conta com um enorme potencial, onde se pode criar uma vasta quantidade de imagens, ideias e conceitos. Para o autor, o Mapa Mental funciona da mesma maneira. À medida que se alocam informações parecidas com a forma em que o cérebro age naturalmente, maior será a destreza para se recordar dos acontecimentos.

Buzan (2005) destaca que a técnica utiliza muitas cores, desenhos, símbolos, imagens e setas, pois isso o torna estimulante. Basicamente, para a elaboração de um Mapa Mental é preciso apenas folhas de papel em branco e sem pautas, canetas, e lápis coloridos.

Os Mapas Mentais podem ser aplicados à diversas atividades, nos mais distintos assuntos, como: na disposição das tarefas escolares, em gerenciamentos de projetos, revisão de conteúdo, na priorização de funções, nas compras do dia a dia, na vida familiar, na comunicação e incontáveis outros meios.

2.8 – UML

UML (*Unified Modeling Language*) é uma linguagem para especificação, construção, visualização e documentação de artefatos de um sistema de software. É promovido pelo Object Management Group (OMG), com contribuições e direitos de autoria das seguintes empresas: *Hewlett-Packard, IBM, ICON Computing, i-Logix, IntelliCorp, Electronic Data Services, Microsoft, ObjecTime, Oracle, Platinum, Ptech, Rational, Reich, Softeam, Sterling, Taskon A/S* e *Unisys* (SILVA; VIDEIRA , 2001).

A ênfase do UML é na definição de uma linguagem de modelagem, por conseguinte, o UML é independente das linguagens de programação, e das ferramentas CASE, bem como dos processos de desenvolvimento. O objetivo do UML é que, dependendo do tipo de projeto, da ferramenta de suporte, ou da organização envolvida, devem ser adaptados diferentes processos/metodologias, mantendo-se, contudo a utilização da mesma linguagem de modelagem (SILVA; VIDEIRA, 2001).

A seguir a definição dos diagramas UML, que são: Caso de Uso, Atividade, Sequência, Classe e Modelo de Diagrama entidade relacionamento (DER).

2.8.1 - Casos de uso

Um diagrama de caso de uso descreve a relação entre atores e casos de utilização de um dado sistema. Este é um diagrama que permite dar uma visão global e de alto nível do sistema, sendo fundamental a definição correta da sua fronteira.

Estes diagramas são utilizados preferencialmente na fase de especificação de requisitos e na modelagem de processos de negócio. Estes diagramas são equivalentes aos homólogos existentes no método OOSE de Ivar Jacobson (SILVA; VIDEIRA, 2001).

2.8.2 - Diagrama de atividades

Os diagramas de atividades são um caso particular dos diagramas de transição de estado, nos quais a maioria dos estados é substituída pelos conceitos correspondentes a ações ou atividades, e as transições são desencadeadas devido à conclusão de ações nos estados originais.

O objetivo deste diagrama é representar os fluxos conduzidos por processamento interno, em contraste com eventos externos representados tipicamente nos diagramas de estado. Este tipo de diagrama pode ser encarado como um caso particular dos diagramas de estado, e inspirado nos diagramas de *workflow*, fluxogramas ou naqueles baseados em SDL (*Specification and Description Language*) (SILVA; VIDEIRA, 2001).

2.8.3 - Diagrama de Sequência

Os diagramas de sequência ilustram interações entre objetos num determinado período de tempo. Em particular, os objetos são representados pelas suas "linhas de vida" e interagem por troca de mensagens ao longo de um determinado período de tempo.

Este tipo de diagrama baseia-se numa variedade de diagramas homólogos existentes em distintos métodos OO, segundo diferentes designações, como, por exemplo, *interaction diagrams*, *message sequence charts*, *message trace*, *event trace*, etc (SILVA; VIDEIRA, 2001).

2.8.4 - Diagrama de Classes

Os diagramas de classes descrevem a estrutura estática de um sistema, em particular, as entidades existentes, as suas estruturas internas, e relações entre si.

Um diagrama de objetos descreve um conjunto de instâncias compatíveis com determinado diagrama de classes. Permitem ilustrar os detalhes de um sistema em determinado momento, ao providenciarem cenários de possíveis configurações (SILVA; VIDEIRA, 2001).

2.8.5 - Modelagem de Entidade e Relacionamento

Diagrama entidade relacionamento (DER) é um modelo diagramático, que descreve o modelo de dados de um sistema com alto nível de abstração. Ele é a principal representação gráfica do Modelo de Entidades e Relacionamentos (MER). É usado para representar o modelo conceitual do negócio.

- MER: Conjunto de conceitos e elementos de modelagem que o projetista de banco de dados precisa conhecer. O Modelo é de Alto Nível.
- DER: Resultado do processo de modelagem executado pelo projetista de dados que conhece o MER.

2.9 - PLANEJAMENTO DE PROJETO

Segundo o PMI[62], o gerenciamento de projetos é a aplicação de conhecimentos, habilidades, ferramentas e técnicas para projetar atividades, que visam atingir os requisitos do projeto.

Para facilitar o gerenciamento do projeto, ele deve ser dividido em fases que constituem seu ciclo de vida.

O Planejamento do Projeto pode ser dividido nas seguintes fases: *Work Breakdown Structure,* Sequenciamento de Atividades, Cronograma, Especificação de Recursos e Especificação de Custos.

2.9.1 - Work Breakdown Structure – WBS

A WBS é um agrupamento de elementos do projeto, que organiza e define total abrangência do projeto. Ela é apresentada na forma de diagrama, sendo que cada nível inferior representa um acréscimo no detalhamento dos elementos.

2.9.2 - Sequenciamento de Atividades

Sequenciamento de Atividades envolve a identificação e a documentação dos relacionamentos lógicos entre as atividades do Cronograma. As atividades podem ser sequenciadas logicamente usando as relações de precedência adequada.

[62] http://brasil.pmi.org/

2.9.3 - Cronograma (estimativa de tempo) – Diagrama de Gantt

O Cronograma do projeto inclui, para cada atividade, no mínimo as datas de início e de término esperado para cada atividade. Se as datas de início de término não forem realizadas é provável que o projeto termine além do tempo planejado.

2.9.4 - Especificação de Recursos

Determinação de quais recursos físicos (como pessoas, equipamentos, materiais, etc) e as quantidades de cada um são necessárias para desenvolver as atividades do projeto.

2.9.5 - Especificação de Custos (orçamento)

Três variáveis envolvidas nos custos:

- Custos de *hardware e Software*;
- Custos de viagens e treinamento;
- Custo relativo ao esforço empregado.

3 - ANÁLISE E ESPECIFICAÇÃO DO SISTEMA

Este capítulo apresenta as análises e especificações do sistema, com os principais requisitos levantados para o desenvolvimento da proposta de trabalho. Também apresenta o mapa mental dos principais diagramas UML (Casos de Uso, classe, sequência e atividade) e o diagrama Entidade-Relacionamento.

3.1 - MAPA MENTAL

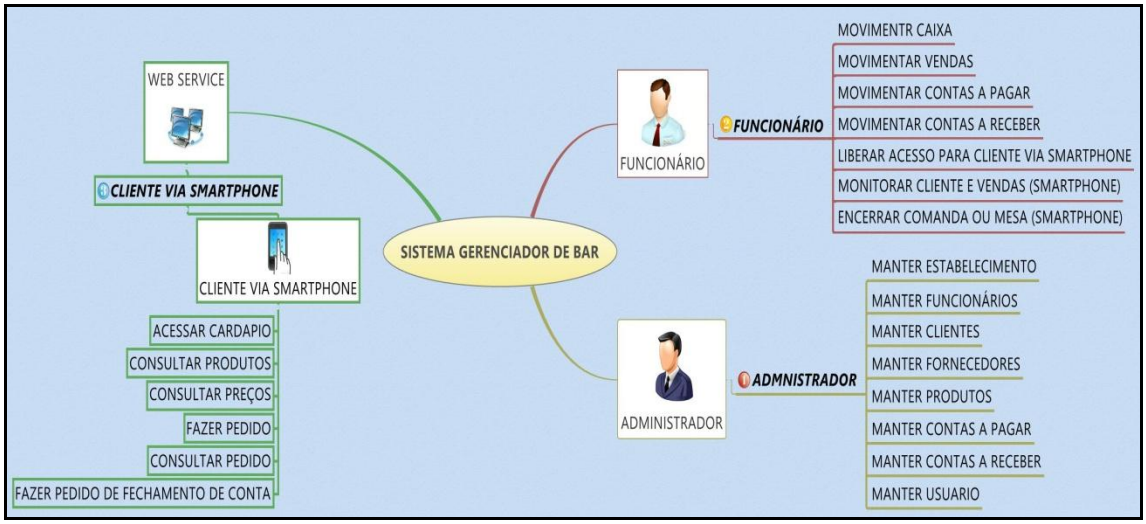

Figura 9 - Mapa Mental

3.2 – LISTA DE EVENTOS

1. Efetuar Cadastro
2. Efetuar Acesso
3. Consultar Cardápio
4. Efetuar Pedido
5. Consultar Pedido
6. Consultar Preço
7. Solicitar Fechamento de Conta
8. Movimentar Caixa
9. Movimentar Vendas
10. Movimentar Contas a Pagar
11. Movimentar contas a Receber
12. Cadastrar Cliente
13. Monitorar Cliente e Vendas
14. Encerrar Conta
15. Manter Estabelecimento
16. Manter Funcionários
17. Manter Cliente
18. Manter Fornecedor
19. Manter Produto
20. Manter Contas a Pagar
21. Manter Contas a Receber

3.3 - CASOS DE USO

A Figura 10 descreve o caso de uso geral do administrador do sistema de controle que será desenvolvido para a demonstração da integração com o aplicativo para *smartphone* (*Virtual Bartender*).

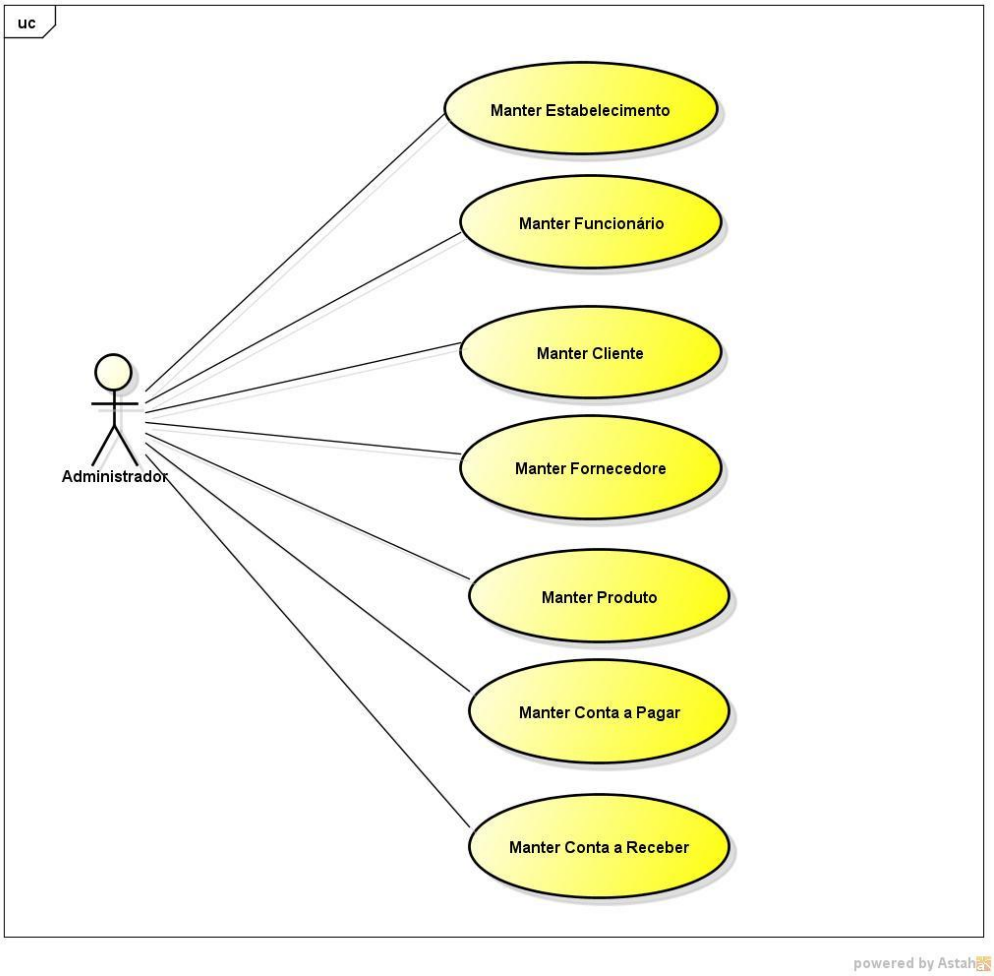

Figura 10 - Caso de Uso Administrador.

A figura 11 descreve o caso de uso geral do funcionário responsável pelo atendimento ao cliente. É ele quem efetua o cadastro e acompanha os pedidos feitos com o aplicativo.

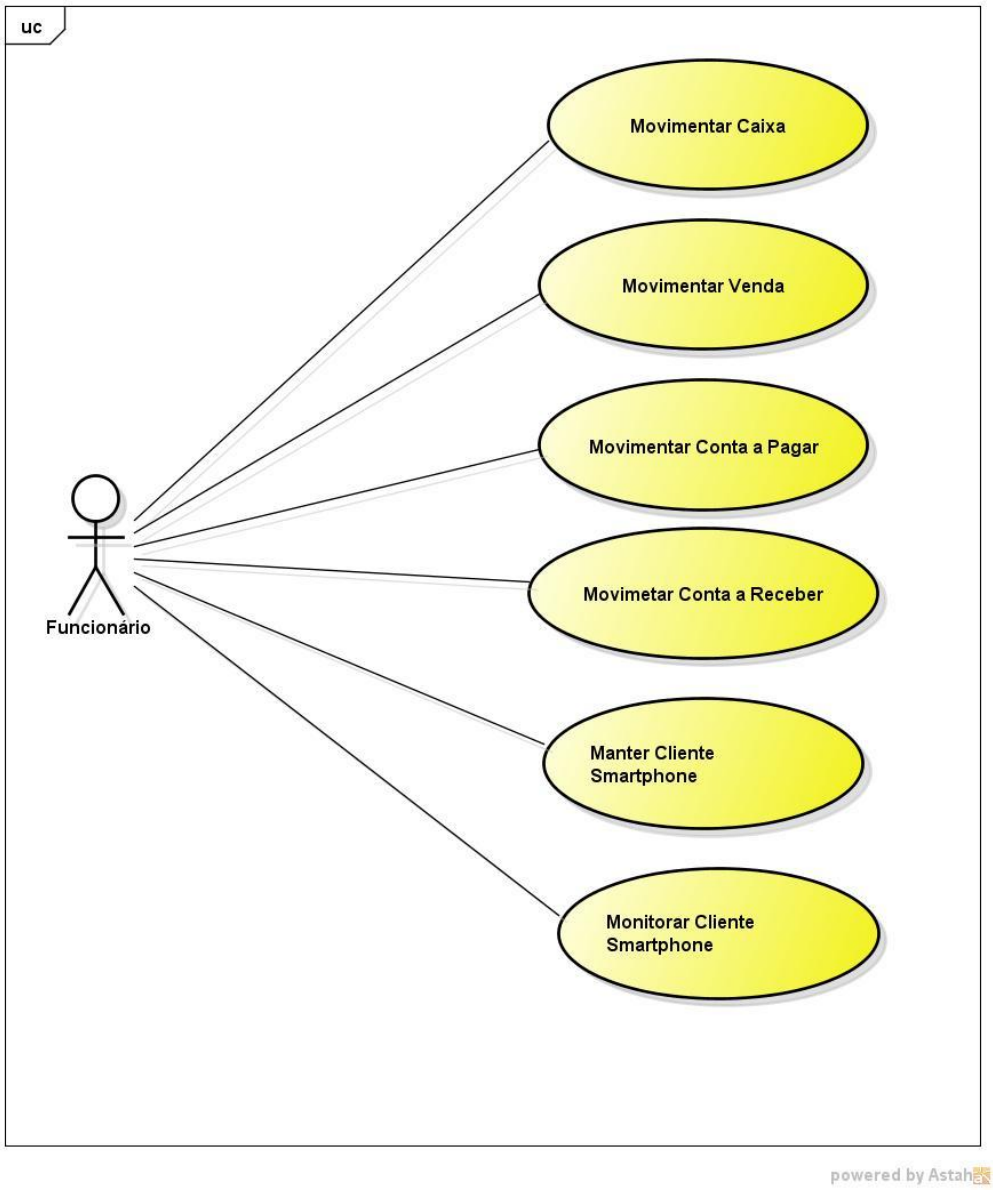

Figura 11 - Caso de Uso Funcionário.

A Figura 12 ilustra o Caso de Uso geral do cliente utilizando o aplicativo para *smartphone*.

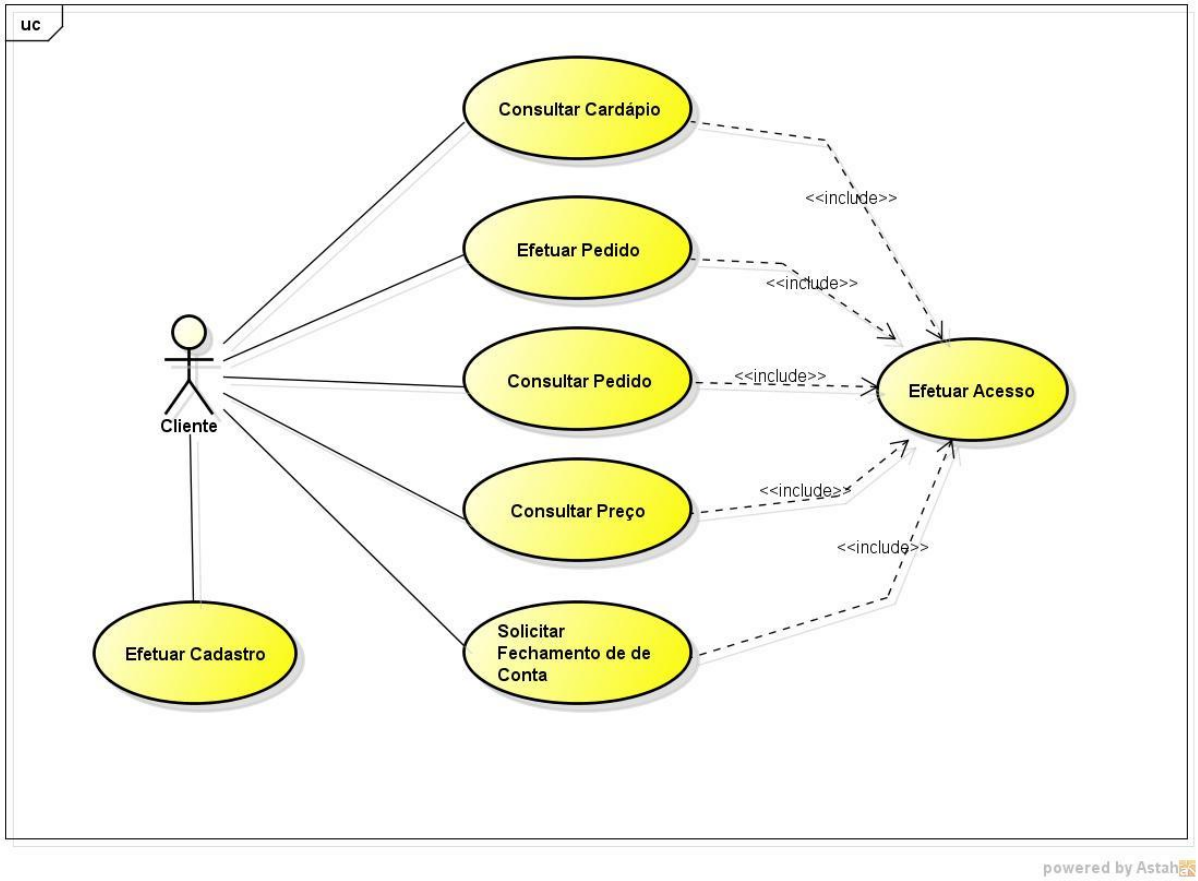

Figura 12 - Caso de Uso Cliente.

3.4 - DIAGRAMA DE ATIVIDADES

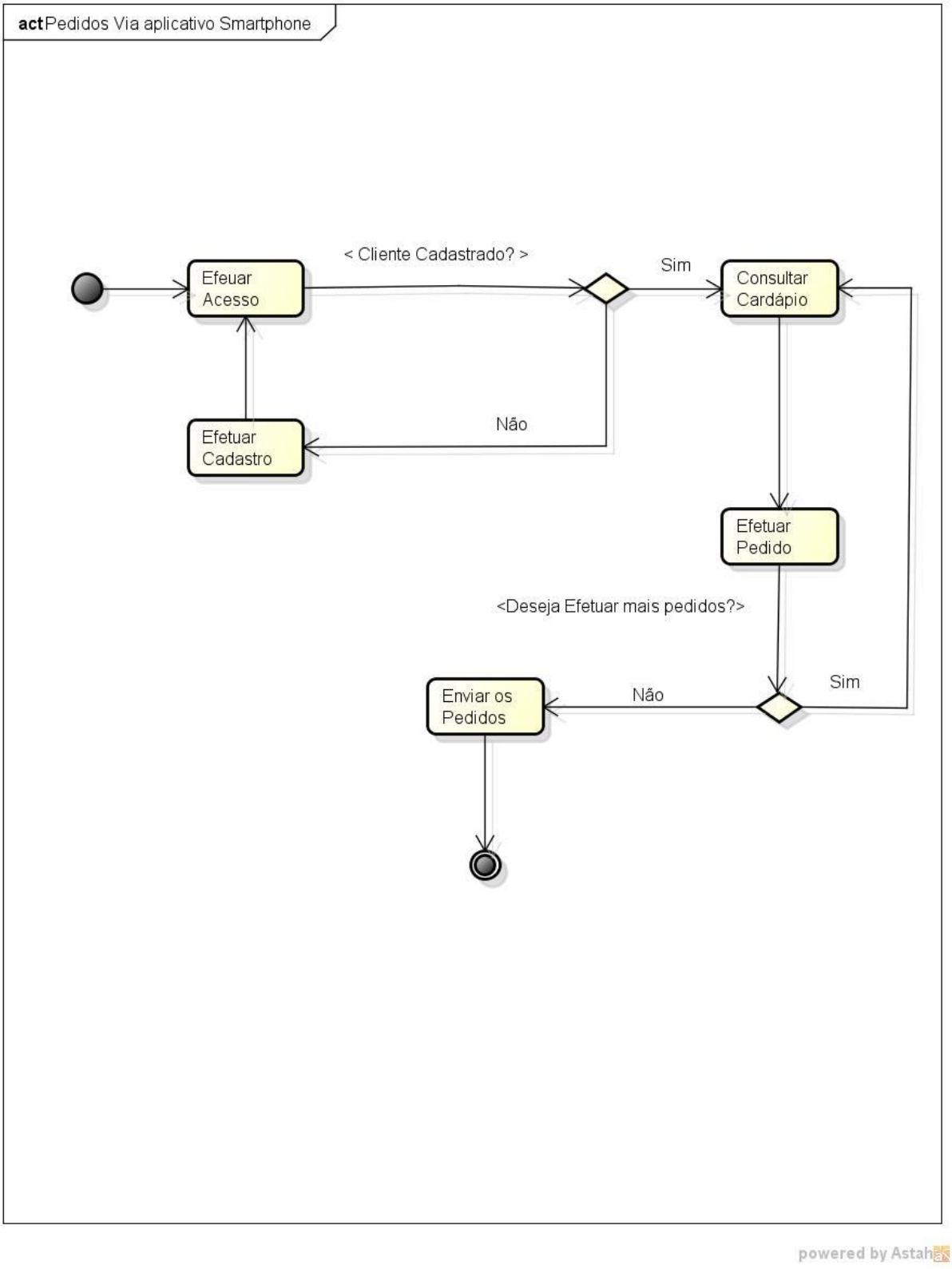

Figura 13 - Diagrama de Atividade Pedido via Smartphone.

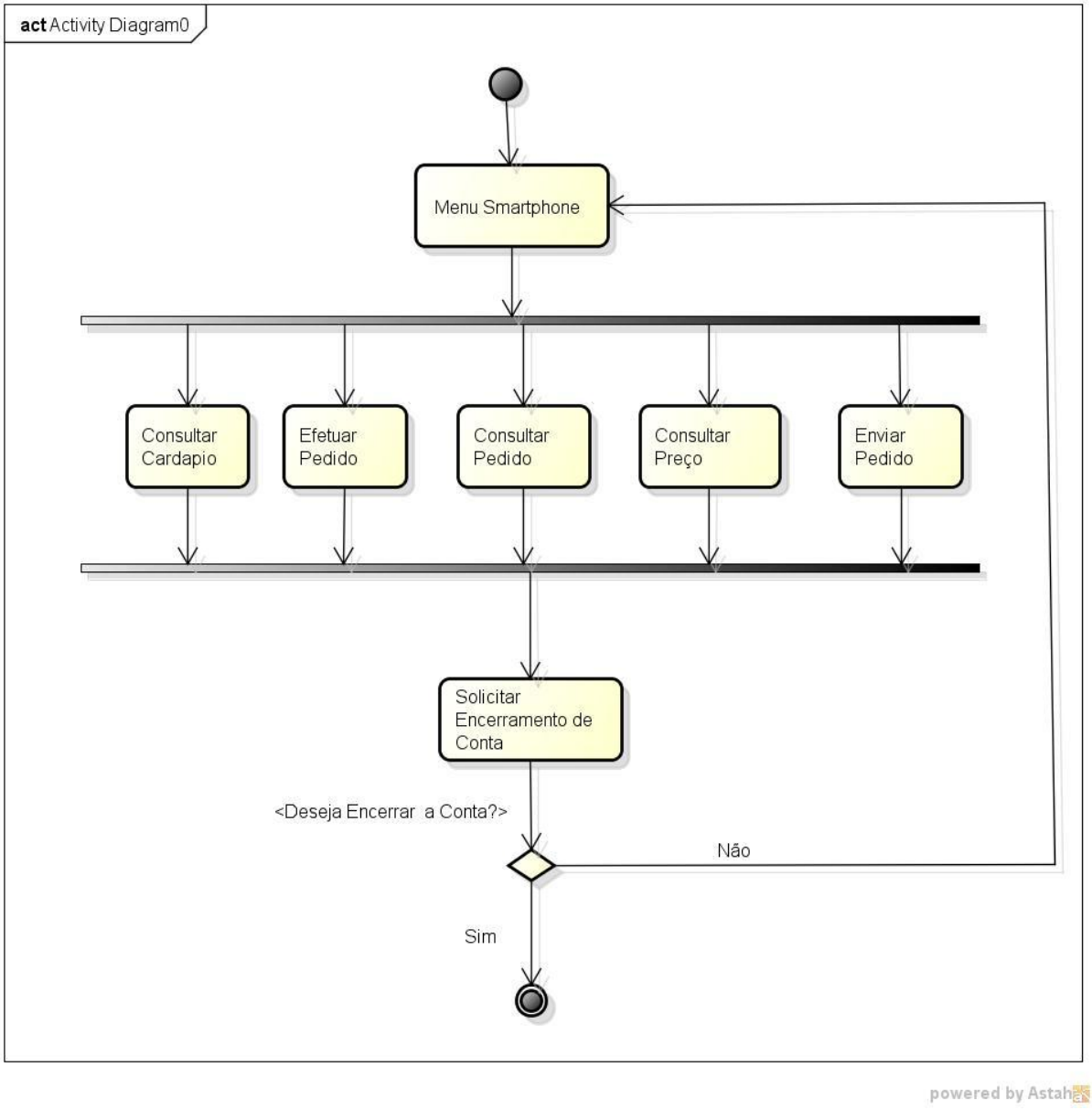

Figura 14 – Diagrama de Atividade Menu Principal do Aplicativo.

3.5 - DIAGRAMA DE SEQUÊNCIAS

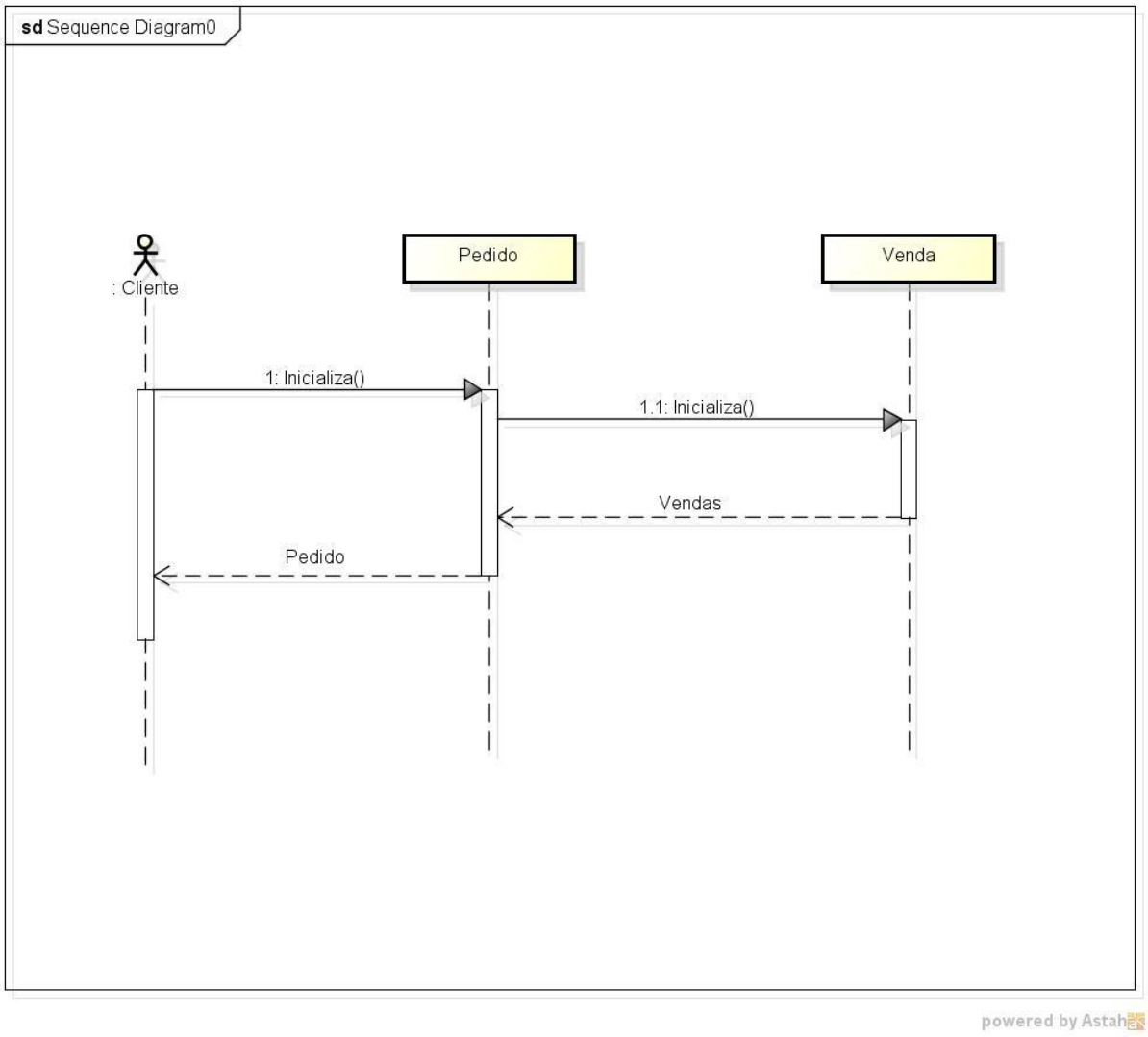

Figura 15 - Diagrama de Sequencia Efetuar Pedido.

3.6 - DIAGRAMA DE CLASSES

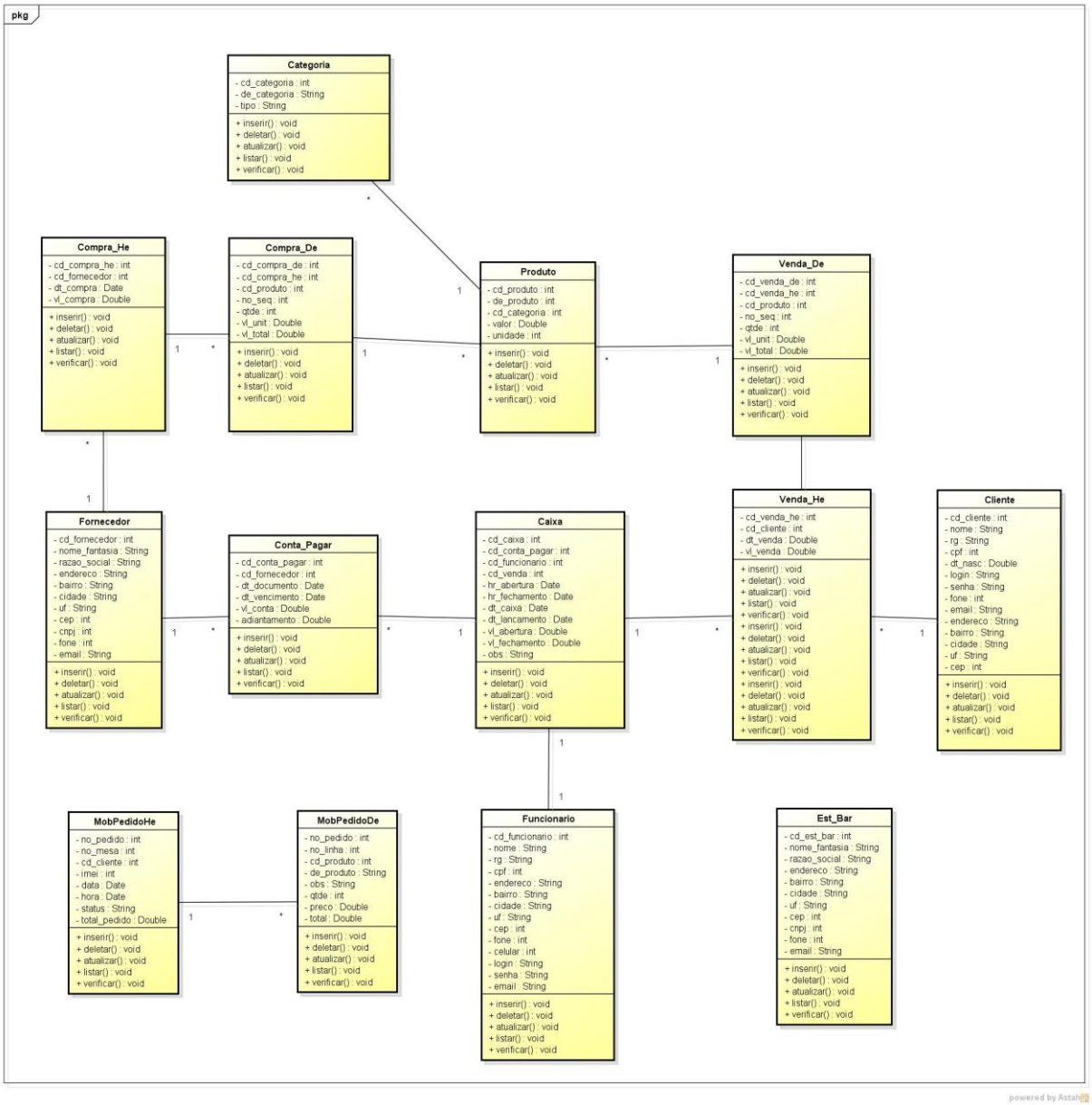

Figura 16 - Diagrama de Classe Aplicação Web.

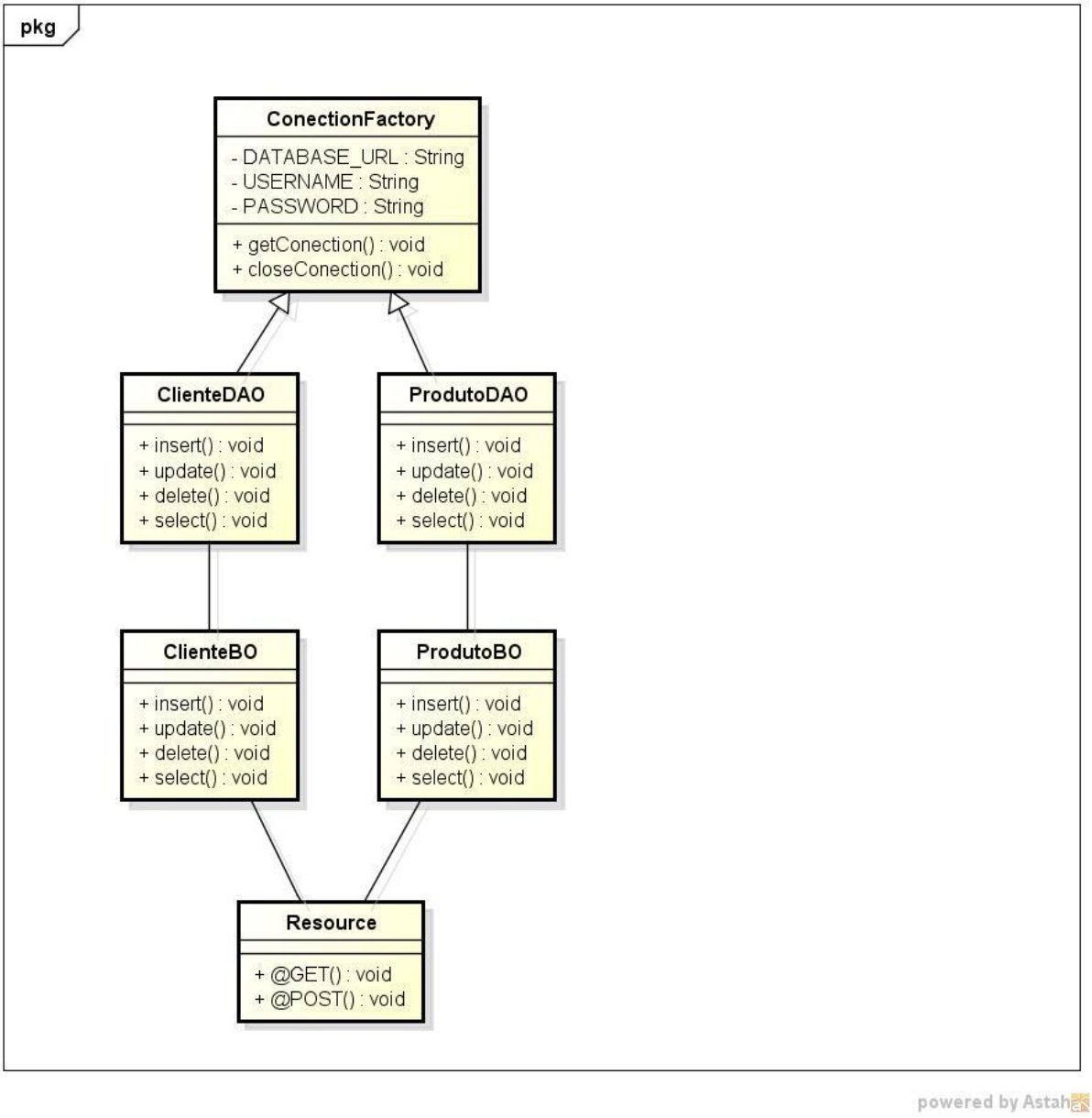

Figura 17 - Diagrama de Classe - Serviço Web (Web Service).

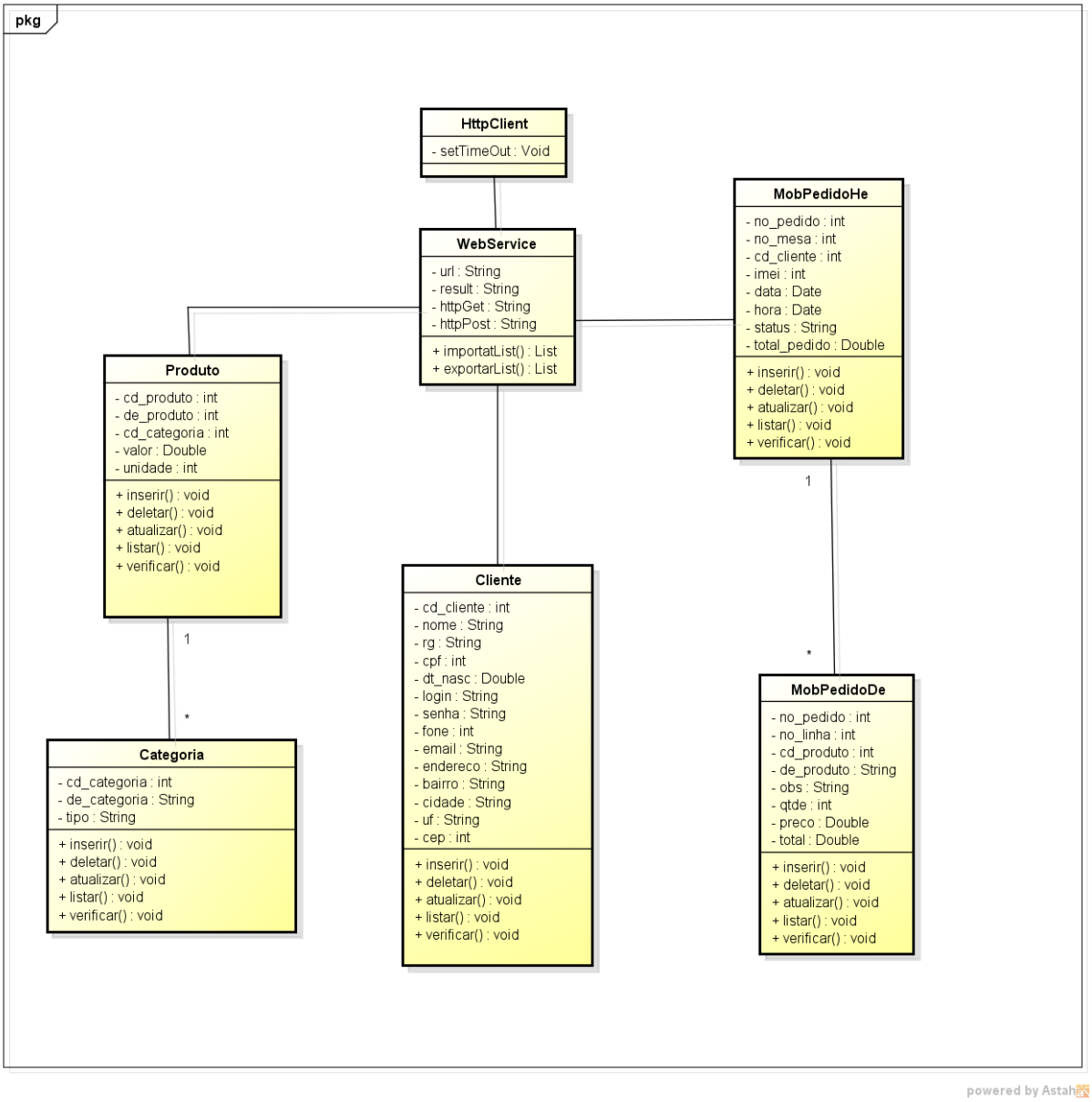

Figura 18 - Diagrama de Classe Aplicativo Smartphone.

3.7 - MODELAGEM DE ENTIDADE E RELACIONAMENTO

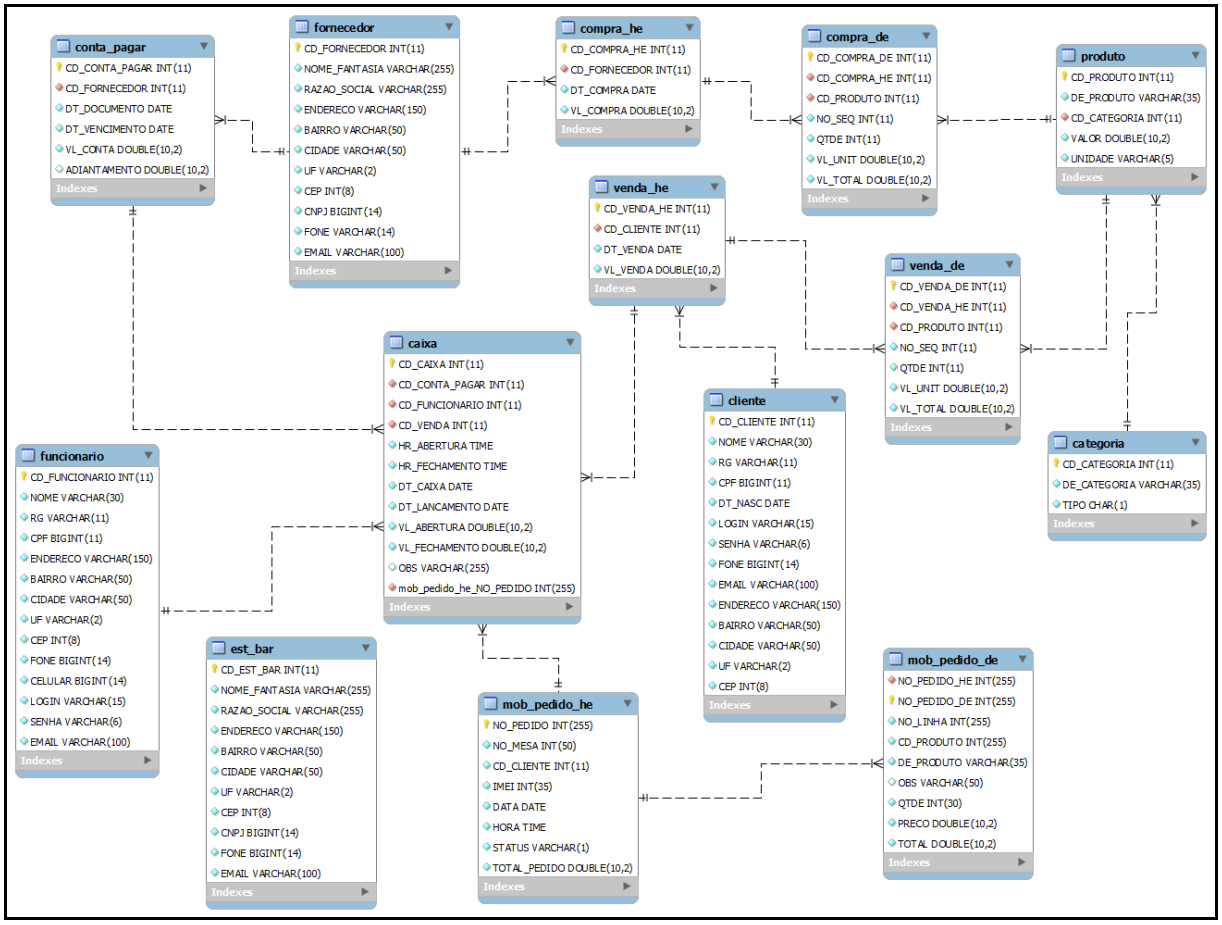

Figura 19 - Modelo Entidade-Relacionamento.

4 - ESTRUTURA DO PROJETO

A metodologia de desenvolvimento utilizada para o trabalho é especificada em fases e etapas. Este capítulo define: a *Work Breakdown Structure* (WBS), conhecida como estrutura analítica de trabalho; o diagrama de sequenciamento de atividades; e o cronograma para acompanhamento das atividades realizadas.

4.1 - WORK BREAKDOWN STRUCTURE – WBS

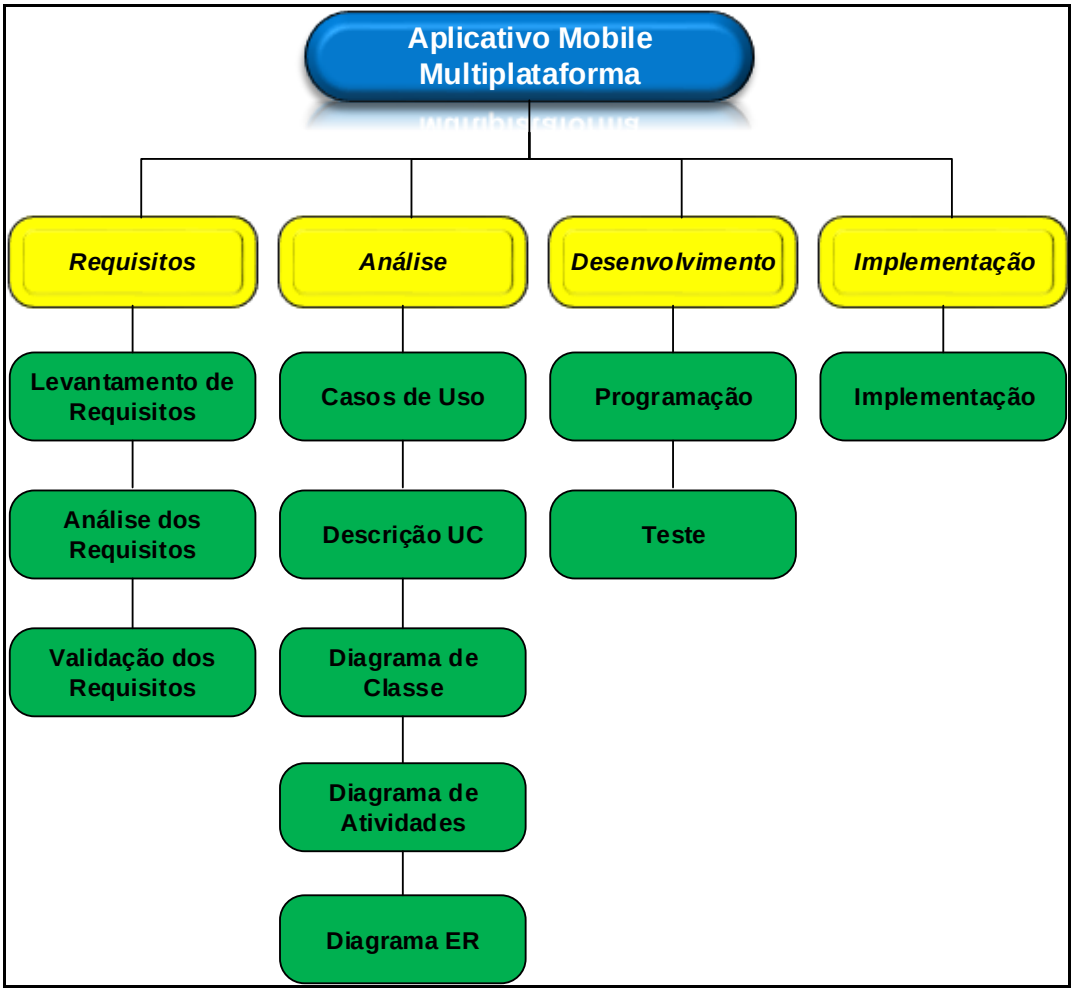

Figura 20 - Work Breakdown Structure (WBS).

4.2 - SEQUENCIAMENTO DE ATIVIDADES

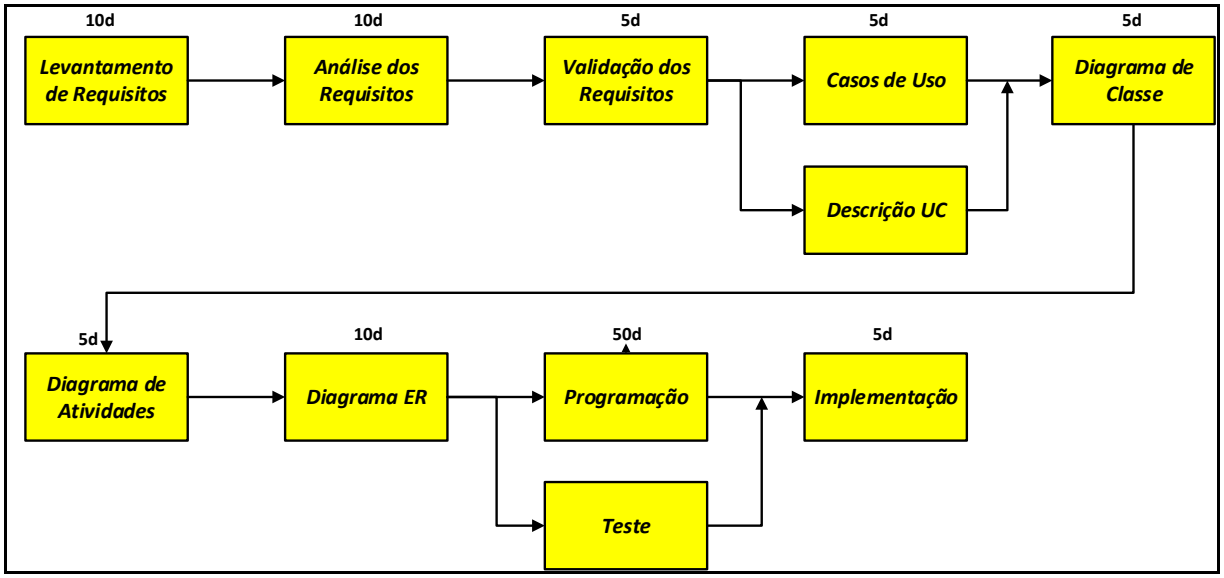

Figura 21 - Sequenciamento de Atividades.

4.3 – CRONOGRAMA

ATIVIDADES	INÍCIO	TÉRMINO	TOTAL DE DIAS
Elaboração da Qualificação	04/11/2013	17/03/2014	133
Exame de Qualificação	07/04/2014	17/04/2014	10
Levantamento de Requisitos	21/04/2014	01/05/2014	10
Análise dos Requisitos	02/05/2014	12/05/2014	10
Validação dos Requisitos	13/05/2014	18/05/2014	5
Casos de Uso e Descrição UC	19/05/2014	24/05/2014	5
Diagrama de Classe	25/05/2014	30/05/2014	5
Diagrama de Atividades	31/05/2014	05/06/2014	5
Diagrama ER	06/06/2014	16/06/2014	10
Programação e Testes	17/06/2014	06/08/2014	50
Implementação	07/08/2014	12/08/2014	5
Defesa final	08/09/2014	13/09/2014	11
INÍCIO DO PROJETO:	04/11/2013		
TÉRMINO DO PROJETO:	12/08/2014		
TOTAL DE DIAS GERAL:	105		

5 – IMPLEMENTAÇÃO DA SOLUÇÃO

Para a implementação do aplicativo *Virtual Bartender* foi utilizada a seguinte arquitetura de software mostrada na figura 22.

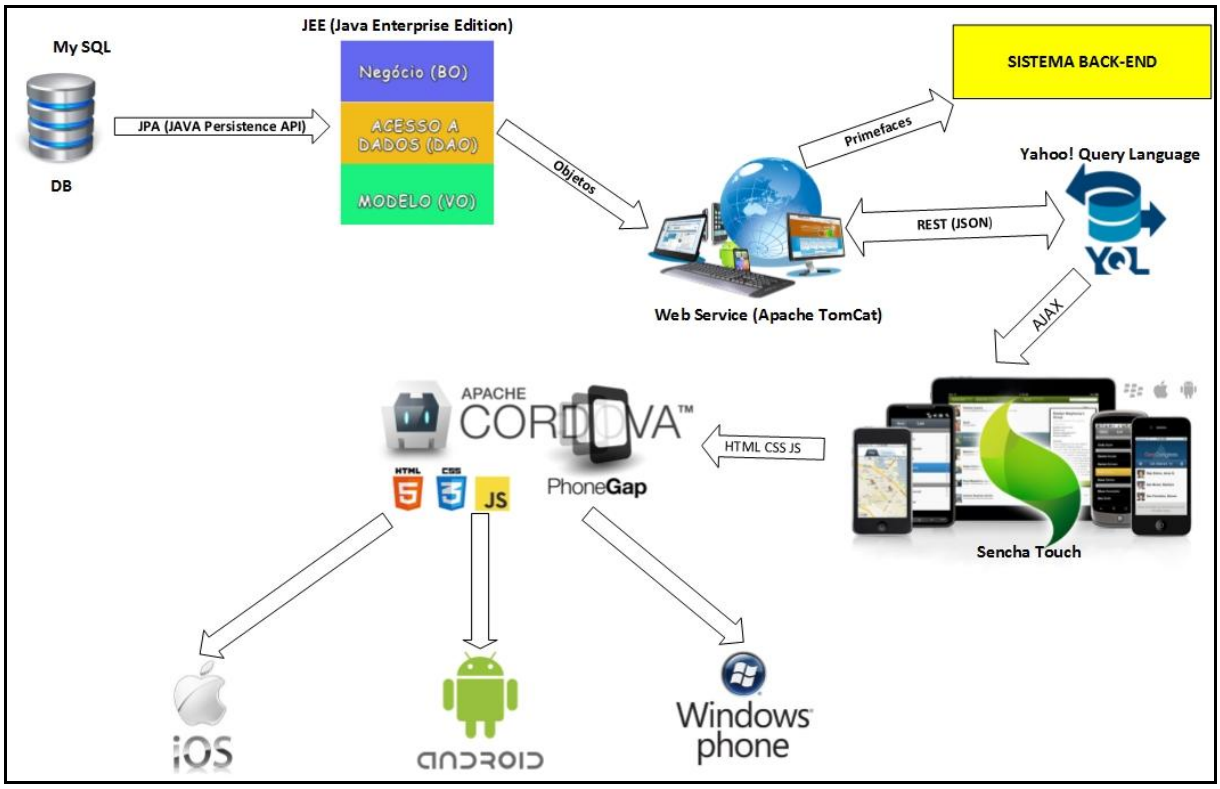

Figura 22 - Arquitetura da Solução

Essa arquitetura contêm todas as tecnologias envolvidas para o desenvolvimento da solução proposta, e para o estudo de caso sobre a integração de diferentes tecnologias com foco principal nos frameworks *Sencha Touch* e *PhoneGap*.

5.1 – BASES DE DADOS

Como base de dados foi utilizado o banco MySQL[63], um servidor e gerenciador de banco de dados (SGBD) relacional.

De licença dupla (sendo uma delas de software livre), projetado inicialmente para trabalhar com aplicações de pequeno e médio porte, mas hoje atendendo às aplicações de grande porte e com mais vantagens do que seus concorrentes. Possui todas as características que um banco de dados de grande porte precisa, sendo reconhecido por algumas entidades como o banco de dados *open source* com maior capacidade para concorrer com programas similares de código fechado, tais como SQL Server (da Microsoft) e Oracle.

Para alimentar o aplicativo móvel foram necessárias as seguintes tabelas: mob_pedido_he; mob_pedido_de; produto; categoria e cliente.

Mob_pedido_he é uma tabela de cabeçalho, que contêm as informações sobre o cliente como, número do pedido, número da mesa, data e hora do pedido entre outras informações. A tabela mob_pedido_de contém o detalhe do pedido como: produto, quantidade e valor.

As outras tabelas que foram apresentadas são para o sistema web *back-end* e serão usadas por completo em trabalhos futuros.

5.2 – PERSISTÊNCIA DOS DADOS

Para realizar a persistência dos dados com a aplicação web, utilizou-se o JPA[64] (*Java Persistence API*). As classes de entidade (modelo) foram criadas a partir das tabelas do banco com as *annotations*. O JPA fornece um modelo de persistência POJO[65] para mapeamento objeto-relacional. O JPA foi desenvolvido pelo grupo de peritos em software EJB 3.0 como parte do JSR 220, mas seu uso não se limita aos componentes de software EJB. Ele também

---

[63] http://www.mysql.com/

[64] https://docs.jboss.org/hibernate/orm/3.5/reference/pt-BR/html/persistent-classes.html

[65] https://docs.jboss.org/hibernate/orm/3.5/reference/pt-BR/html/persistent-classes.html

pode ser usado diretamente por aplicações web e clientes de aplicativos, e até mesmo fora da plataforma Java EE, por exemplo, em aplicações Java SE.

O código 1 apresenta a classe gerada a partir das tabelas pelo *framework* JPA.

Código 1 - Classe de Entidade.

```
package br.com.bartender.entity;
import java.io.Serializable;
import javax.persistence.*;
import java.util.List;
/**
 * The persistent class for the categoria database table.
 *
 */
@Entity
@NamedQuery(name="Categoria.findAll", query="SELECT c FROM Categoria c")
public class Categoria implements Serializable {
	private static final long serialVersionUID = 1L;

	@Id
	@GeneratedValue(strategy=GenerationType.IDENTITY)
	@Column(name="CD_CATEGORIA")
	private int cdCategoria;

	@Column(name="DE_CATEGORIA")
	private String deCategoria;

	private String tipo;

	//bi-directional many-to-one association to Produto
	@OneToMany(mappedBy="categoria")
	private List<Produto> produtos;

	public Categoria() {
	}
	public int getCdCategoria() {
		return this.cdCategoria;
	}
	public void setCdCategoria(int cdCategoria) {
		this.cdCategoria = cdCategoria;
	}
	public String getDeCategoria() {
		return this.deCategoria;
	}
	public void setDeCategoria(String deCategoria) {
		this.deCategoria = deCategoria;
	}
	public String getTipo() {
		return this.tipo;
	}
	public void setTipo(String tipo) {
		this.tipo = tipo;
	}
	public List<Produto> getProdutos() {
		return this.produtos;
	}
	public void setProdutos(List<Produto> produtos) {
		this.produtos = produtos;
	}
	public Produto addProduto(Produto produto) {
		getProdutos().add(produto);
		produto.setCategoria(this);

		return produto;
	}

	public Produto removeProduto(Produto produto) {
		getProdutos().remove(produto);
		produto.setCategoria(null);

		return produto;
	}

}
```

5.3 – MÓDULO DE INTEGRAÇÃO

Para o desenvolvimento da aplicação *Web Service* foi utilizado como solução de integração entre sistemas, uma aplicação do tipo *Rest*[66] *Java EE*. *Rest* é uma arquitetura usada para integrar sistemas distribuídos.

A aplicação *Web Service* será responsável por acessar a base de dados, pois o aplicativo por sua vez não consegue conectar-se diretamente com a base de dados. O *Web Service* fará toda a integração entre as tecnologias. Ele é quem vai enviar (POST) ou receber (GET) as informações para o aplicativo *Virtual Bartender*, e o sistema retaguarda (*back-end*) através de arquivos do tipo JSON[67], pois os arquivos desse tipo costumam ser bem mais leves do que os arquivos do tipo XML[68].

A código 2 apresenta um modelo de arquivo JSON gerado pela aplicação de serviço, a partir da tabela Categoria do banco de dados.

Código 2 - Arquivo JSON.

```
[
        {
                "cd_categoria":1,
                "de_categoria":"LANCHES",
                "tipo":"A"
        },
        {
                "cd_categoria":2,
                "de_categoria":"PORÃ‡Ã•ES",
                "tipo":"A"
        },
        {
                "cd_categoria":3,
                "de_categoria":"REFRIGERANTES",
                "tipo":"B"},
        {
                "cd_categoria":4,
                "de_categoria":"CHOPP E CERVEJAS",
                "tipo":"B"
        }
        {
                "cd_categoria":5,
                "de_categoria":"Ã?GUA",
                "tipo":"B"
        },
]
```

[66] http://docs.oracle.com/javaee/6/tutorial/doc/gijqy.html

[67] http://json.org/json-pt.html

[68] http://www.xml.com/pub/a/98/10/guide0.html

Como servidor de aplicações web Java foi utilizado o servidor open source Apache Tomcat[69], um servidor com um bom desempenho e com baixo consumo de hardware, especifico para aplicações do tipo JEE[70]. Ele mantém arquivos compactados conhecidos como *war*.

Para a integração do aplicativo móvel com o *Web Service* será usado a API YQL[71] (*Yahoo Query language*), que funciona de maneira similar a um banco de dados, onde os serviços são disponibilizados por tabelas, que podemos fazer consultas através de uma sintaxe semelhante a do SQL. Com o YQL é possível filtrar e juntar dados de *Web Services*. O motivo de ter que optar pelo serviço dele é o fato de que o *Sencha Touch* não consegue comunicar-se diretamente com o *Web Service.* E a única forma é um intermédio entre a camada AJAX do *Sencha Touch* com o YQL. Para trabalhos futuros será uma boa pesquisa encontrar uma melhor forma de integrar o *Sencha Touch* com *Web Service.*

5.4 – TIPOS DE CONEXÃO

Para a conexão do aplicativo móvel com o servidor web será utilizada a conexão do tipo Wi-Fi. Um conjunto de especificações de redes locais sem fio (WLAN - *Wireless Local Área Network*) baseado no padrão IEEE 802.11[72], tendo como entidade responsável pelo padrão de licenciamento a Wi-Fi Alliance[73].

Com essa tecnologia Wi-Fi é possível conectar computadores e outros dispositivos como *smartphones*, *tablets*, impressoras, vídeo games entre outros dispositivos compatíveis com essa tecnologia.

Esse tipo de rede está cada vez mais comum, não só nos ambientes domésticos e corporativos, mas também em locais públicos (bares, lanchonetes, shoppings, livrarias, aeroportos, etc).

---

[69] http://tomcat.apache.org/

[70] http://www.oracle.com/technetwork/java/javaee/overview/index.html

[71] https://developer.yahoo.com/yql/

[72] http://www.ieee802.org/11/

[73] http://www.wi-fi.org/

5.5 – APLICAÇÃO WEB JEE

Para o desenvolvimento da aplicação de serviço web e sistema retaguarda (*back-end*) foi utilizada a IDE (*Integrated Development Environment*) ou Ambiente Integrado de Desenvolvimento Eclipse.

A figura 23 apresenta a estrutura desenvolvida para uma melhor organização da aplicação web, dividida em pacotes.

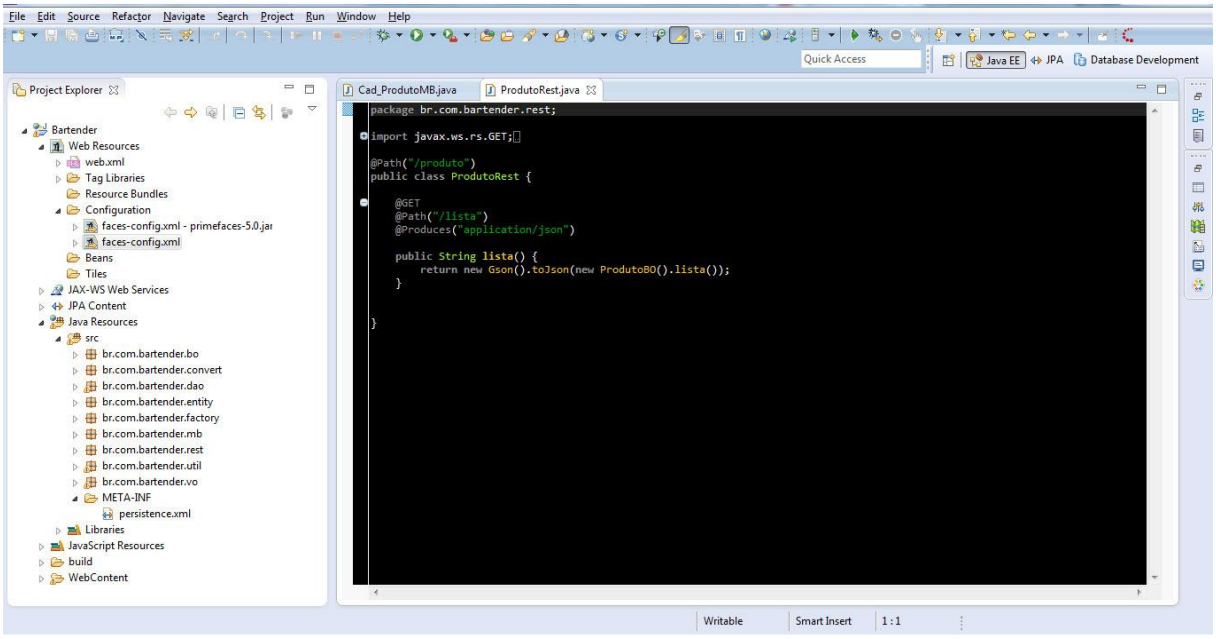

Figura 23- IDE Eclipse

Para uma melhor organização do projeto *web service* foi utilizado o padrão de projeto Java contendo os pacotes: bo (*Business*) classe das regras de negócio; dao (*Data access object*) classe de acesso a dados; e vo (*Value Object*), classe modelo contendo os valores dos objetos.

A código 3 apresenta a classe modelo. É a responsável pelo transporte dos valores do objeto entre as classes da aplicação.

Código 3 - Classe VO.

```
package br.com.bartender.vo;
import java.io.Serializable;

public class CategoriaVO implements Serializable {

	/**
	 *
	 */
	private static final long serialVersionUID = -3580911986840817043L;
	private Integer cd_categoria;
	private String de_categoria;
	private String tipo;

	public CategoriaVO() {
		super();
		// TODO Auto-generated constructor stub
	}

	public CategoriaVO(Integer cd_categoria, String de_categoria, String tipo) {
		super();
		this.cd_categoria = cd_categoria;
		this.de_categoria = de_categoria;
		this.tipo = tipo;
	}

	public Integer getCd_categoria() {
		return cd_categoria;
	}

	public void setCd_categoria(Integer cd_categoria) {
		this.cd_categoria = cd_categoria;
	}

	public String getDe_categoria() {
		return de_categoria;
	}

	public void setDe_categoria(String de_categoria) {
		this.de_categoria = de_categoria;
	}

	public String getTipo() {
		return tipo;
	}

	public void setTipo(String tipo) {
		this.tipo = tipo;
	}

	@Override
	public int hashCode() {
		final int prime = 31;
		int result = 1;
		result = prime * result
				+ ((cd_categoria == null) ? 0 : cd_categoria.hashCode());
		result = prime * result
				+ ((de_categoria == null) ? 0 : de_categoria.hashCode());
		result = prime * result + ((tipo == null) ? 0 : tipo.hashCode());
		return result;
}
```

O código 4 apresenta a classe Categoria DAO, que faz o acesso aos dados do MySQL. Nessa classe contém o método responsável por listar os atributos da tabela categoria.

Código 4 - Classe DAO.

```
 package br.com.bartender.dao;

import java.sql.Connection;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.util.ArrayList;

import br.com.bartender.factory.ConnectionFactory;
import br.com.bartender.vo.CategoriaVO;
import br.com.bartender.vo.ProdutoVO;

public class CategoriaDAO extends ConnectionFactory{

      public ArrayList<CategoriaVO> lista() {

            ArrayList<CategoriaVO> lista = null;
            Connection conn = null;
            ResultSet resultSet = null;
            PreparedStatement statement = null;
          conn = getConnection();

            try {

                  String sql = "SELECT * FROM CATEGORIA ORDER BY CD_CATEGORIA";

                  statement = conn.prepareStatement(sql);
                  resultSet = statement.executeQuery();
                  lista = new ArrayList<CategoriaVO>();

                  while (resultSet.next()) {


                        CategoriaVO vo = new CategoriaVO();
                        vo.setCd_categoria(resultSet.getInt("CD_CATEGORIA"));
                        vo.setDe_categoria(resultSet.getString("DE_CATEGORIA"));
                        vo.setTipo(resultSet.getString("TIPO"));


                        System.out.println(vo.getDe_categoria());
                        lista.add(vo);
                  }

            } catch (Exception e) {
                  System.out.println("Erro de gerar lista: " + e);
                  lista = null;

            } finally {
                  closeConnection(conn, statement, resultSet);

            }

            return lista;
      }

}
```

O código 5 apresenta a classe BO, nela contém o método que retorna uma lista de objeto categoria, que foi carregados pela classe CategoriaDAO. Essa classe faz toda a lógica de negócio da aplicação:

Código 5 - Classe BO.

```
 package br.com.bartender.bo;
import java.util.ArrayList;
import br.com.bartender.dao.CategoriaDAO;
import br.com.bartender.vo.CategoriaVO;

public class CategoriaBO {
	public ArrayList<CategoriaVO> lista() {
		return new CategoriaDAO().lista();
	}
}
```

Por fim temos também o código 6 que apresenta a classe CategoriaRest. Ela que fica responsável por listar ou receber os objetos JSON.

Código 6 - Classe Rest.

```
package br.com.bartender.rest;

import javax.ws.rs.GET;
import javax.ws.rs.Path;
import javax.ws.rs.Produces;

import br.com.bartender.bo.CategoriaBO;

import com.google.gson.Gson;

@Path("/categoria")
public class CategoriaRest {

	@GET
	@Path("/lista")
	@Produces("application/json")

	public String lista() {
		return new Gson().toJson(new CategoriaBO().lista());
	}

}
```

5.6 - APLICATIVO MÓVEL

O desenvolvimento do aplicativo móvel foi dividido em duas etapas. Na primeira foi utilizado como ferramenta para desenvolvimento, o editor de texto Sublime[74]. Nessa o foco principal foi o framework JS *Sencha Touch*. Tal framework utiliza-se da biblioteca Sencha conhecida como Ext JS. Toda a aplicação foi totalmente desenvolvida com recursos web como HTML, JS, e CSS. Nesse primeiro momento, o aplicativo pode ser considerado um app web móvel.

Como padrão de desenvolvimento foi utilizado o padrão de camadas conhecido como MVC (*Model View Controller*) ou Modelo Visão e Controlador.

A camada *model* ou modelo gerencia o comportamento e os dados da aplicação, responde aos pedidos da *view* e as instruções de mudança de estado dos controladores.

Visão é responsável por: gerar a interface gráfica, receber as informações dos controladores e modelos, e apresentar o resultado nas marcações finais HTML. Nela encontram-se os formulários e etc.

Controlador, como o próprio nome diz, controla a aplicação, o cérebro; nele encontra toda a regra de negócio do aplicativo.

[74] http://www.sublimetext.com/

A Figura 24 mostra a estrutura do aplicativo no editor de texto sublime, usando o padrão de projeto MVC.

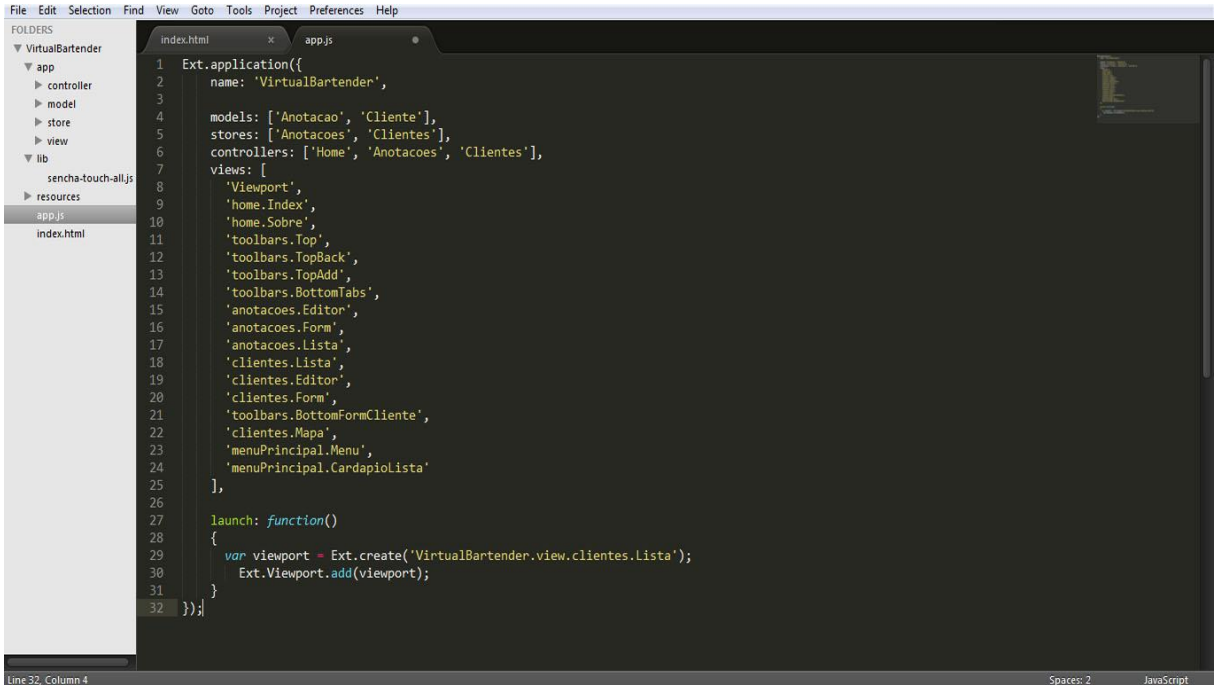

Figura 24 - Estrutura do Aplicativo Virtual Bartender

A segunda etapa teve como foco o *framework PhoneGap*, este é responsável por pegar a aplicação web (HTML, JS, CSS) e gerar o aplicativo binário nativo de cada plataforma. Com o *PhoneGap* a aplicação final será para as plataformas iOS, Windows Phone, Android e BlackBarry Os, entre outras. O *PhoneGap* oferece duas opções de desenvolvimento, o programador pode gerar a aplicação individual em cada plataforma, usando a biblioteca do *PhoneGap*. Nesse caso é necessário ter a IDE e o SDK de cada plataforma, ou também é possível gerar com o *PhoneGap build*. Ele por sua vez, faz todo o processo automaticamente, sem precisar do desenvolvedor. O único trabalho é enviar o aplicativo app web para o *PhoneGap build* para ter uma aplicação final nas plataformas que escolher. Alem disso é possível disponibilizar o aplicativo direto do site do *PhoneGap build*, sem a necessidade de enviar para uma app store (loja de aplicativos).

6 – CONCLUSÃO

Com a análise das ferramentas e tecnologias levantadas no presente trabalho, pode-se concluir que o desenvolvimento de aplicativos utilizando as tecnologias aqui apresentadas tem muitas vantagens. Uma delas sem dúvida é a facilidade de portar o aplicativo para qualquer plataforma móvel. Outra vantagem é a opção de distribuição do aplicativo; o desenvolvedor pode optar pelas lojas de aplicativos ou simplesmente disponibilizar no *Build PhoneGap* um serviço de distribuição gratuita, sem ter que pagar para manter seu aplicativo.

Com as pesquisas levantadas durante o trabalho, foi alcançado um crescimento pessoal e profissional muito satisfatório, além dos conhecimentos adquiridos pelas tecnologias aqui apresentadas.

Com base nas pesquisas e análises, verificou-se que o aplicativo tem uma grande possibilidade de comercialização.

Na continuação do trabalho serão aplicados os conhecimentos adquiridos com os estudos e referências levantadas, para o desenvolvimento do aplicativo *Virtual Bartender*.

REFERÊNCIAS BIBLIOGRÁFICAS

ALLEN, Sarah; GRAUPERA, Vidal; LUNDRIGAN, Lee. **Desenvolvimento profissional multiplataforma para smartphone: iPhone, Android, Windows Mobile e BlackBerry.** 1. Ed. Tradução de Frank Coelho de Alcantara. Rio de Janeiro: Alta Books, 2012.

CLARK, John E; JOHNSON, Bryan P. **Sencha Touch Mobile JavaScript Framework: Build web applications for Apple iOS and Google Android touchscreen devices with this first HTML5 mobile framework.** 1. Ed. Birmingham - Mumbai: Packt Publishing Ltd, 2012.

FERREIRA, Elcio; EIS, Diego. **HTML5: Curso W3C Escritório Brasil.** Disponivel em: < http://www.w3c.br/pub/Cursos/CursoHTML5/html5-web.pdf >. Acesso em: 04. mar. 2014.

FLANAGAN, David. **JavaScript: The Definitive Guide.** 6. Ed. Sebastopol, CA: O'Reilly, 2012.

FREDERICK, Gail Rahn; LAL, Rajesh. **Dominando o desenvolvimento Web para Smartphone: Construindo aplicativos baseado no JavaScript, CSS, HTML, e Ajax para iPhone, Android, Palm Pre, BlackBerry, Windows Mobile e Nokia S60.** 1. Ed. Tradução de Eveline Machado. Rio de Janeiro: Alta Books, 2011.

GATOL, Rohit; PATEL, Yogesh. **Beginning PhoneGap: Mobile Web Framework for JavaScript and HTML5**. 1. Ed. New York, NY: Apress, 2012.

KALIN, Martin. **Java Web Services: Up and Running.** 1. Ed. Sebastopol, CA: O'Reilly, 2013.

KOSMACZEWSKI**,** Adrian. **Sencha Touch 2 Up and Running.** 1. Ed. Sebastopol, CA: O'Reilly, 2013.

NATILI, Giorgio. **PhoneGap 3 Beginner's Guide 2013.** 2. Ed. Birmingham - Mumbai: Packt Publishing Ltd, 2013.

OEHLMAN, Damon; BLANC, Sébastien. **Aplicativos Web Pro Android: Desenvolvimento Pro Android Usando HTML5, CSS3 e JavaScript.** 1. Ed. Tradução de Kleber Rodrigo de Carvalho. Rio de Janeiro: Editora Ciência moderna Ltda., 2012.

SILVA, Alberto Manuel Rodrigues da; VIDEIRA, Carlos Alberto Escaleira. **UML: Metodologias e Ferramentas CASE: Linguagem de Modelação UML, Metodologias e Ferramentas CASE na Concepção e Desenvolvimento de Software.** 1. Ed. Porto - Lisboa Portugal: Centro Atlântico, 2001.

SILVA, Maurício Samy. **HTML5: A linguagem de marcação que revolucionou a Web.** 1. Ed. São Paulo: Novatec Editora, 2010.

SILVA, Maurício Samy. **JavaScript: guia do programador.** 1. Ed. São Paulo: Novatec Editora, 2010.

SILVA, Maurício Samy. **CSS3: Desenvolva aplicações web profissionais com uso dos poderosos recursos de estilização das CSS3.** 1. Ed. Tradução de Rafael Zanolli. São Paulo: Novatec Editora, 2012.

SILVA, Osmar J. **Java Script: guia prático do Webmaster.** 2. Ed. Revisada. São Paulo: Érica, 2000.

STARK, Jonathan; JEPSON, Brian. **Construindo aplicativos Android com HTML, CSS e JavaScript.** 1. Ed. Tradução de Acauan Fernades. São Paulo: Novatec Editora; Sebastopol, CA: O'Reilly, 2012.

SOMMERVILLE, Ian. **Engenharia de Software**. 6. Ed. São Paulo: Person, 2003.