UNIVERSIDADE FEDERAL DO PIAUÍ – UFPI

CURSO DE BACHARELADO EM SISTEMAS DE INFORMAÇÃO

AUTOMATIZANDO ATENDIMENTO EM BARES E RESTAURANTES COM O USO DE DISPOSITIVOS MÓVEIS

Melksedek Amorim Santos Góis

PICOS – PIAUÍ

2016

Melksedek Amorim Santos Góis

Automatizando atendimento em bares e restaurantes com o uso de Dispositivos Móveis

Monografia submetida ao Curso de Bacharelado em Sistemas de Informação como requisito parcial para obtenção de grau de Bacharel em Sistemas de Informação.

Orientador: Prof. Ivenilton Alexandre de Souza Moura.

PICOS – PIAUÍ

2016

FICHA CATALOGRÁFICA
Serviço de Processamento Técnico da Universidade Federal do Piauí
Biblioteca José Albano de Macêdo

G616a Góis, Melksedek Amorim Santos.
Automatizando atendimento em bares e restaurantes com o uso de dispositivos móveis / Melksedek Amorim Santos Góis. – 2016.
CD-ROM : il.; 4 ¾ pol. (57 f.)

Monografia(Bacharelado em Sistemas de Informação) – Universidade Federal do Piauí, Picos, 2016.
Orientador(A): Profº. Esp. Ivenilton Alexandre de Souza Moura
.

1. Web Service. 2. Android. 3. Dispositivos Móveis. I. Título.

CDD 005.3

AUTOMATIZANDO ATENDIMENTO EM BARES E RESTAURANTES COM O USO DE DISPOSITIVOS MÓVEIS

MELKSEDEK AMORIM SANTOS GOIS

Monografia aprovada como exigência parcial para obtenção do grau de Bacharel em Sistemas de Informação.

Data de Aprovação

Picos – PI, 19 de fevereiro de 2016

Profª. Esp. Ivenilton Alexandre de Souza Moura
Orientador

Prof. Me. Frank César Lopes Véras
Membro

Jonnison Lima Ferreira
Membro

Dedico este trabalho primeiramente a Deus, por ser essencial em minha vida, autor de meu destino, meu guia, socorro presente na hora da angústia, ao meu pai Oberto Prata Góis, minha mãe Iracilda Amorim dos Santos Góis, meus irmãos Rebeca e Vinicius, ao meu avô materno, Raimundo Gomes dos Santos, de quem recebi carinhosamente o titulo de “Dr. Melk”. E a todos os familiares e amigos que direta ou indiretamente contribuíram para que isso fosse possível.

AGRADECIMENTOS

Primeiramente a Deus que permitiu que tudo isso acontecesse, ao longo de minha vida, e não somente nestes anos como universitário, mas que em todos os momentos é o maior mestre que alguém pode conhecer.

Aos meus avós paternos José de Góis Filho e Laudicena Ribeiro Prata, e materno Raimundo Gomes dos Santos e Iraci Amorim de Carvalho por me darem muito amor, carinho e exemplos valiosos de vida, contribuindo na minha formação como pessoa.

Aos meus pais, Oberto Prata Góis e Iracilda Amorim dos Santos Góis, que sempre transformaram o impossível em possível para dar a mim e meus irmãos o melhor. Obrigado pela família, pelo amor, pelo carinho e por sempre acreditarem em mim, até quando não acreditei em mim mesmo. Vocês são meu porto seguro.

Obrigado! Tios e primos pela contribuição valiosa.

Meus agradecimentos aos amigos Israel, Weder e João Mariano, companheiros de trabalhos e irmãos na amizade que fizeram parte da minha formação e que vão continuar presentes em minha vida.

Ao meu orientador Ivenilton Alexandre de Souza Moura, pelo suporte no pouco tempo que lhe coube, pelas suas correções e incentivos.

A esta universidade, seu corpo docente, direção e administração que oportunizaram a janela que vislumbro um horizonte superior, eivado pela acendrada confiança no mérito e ética aqui presentes.

A todos que direta ou indiretamente fizeram parte da minha formação, o meu muito obrigado.

“Empreendedores são aqueles que entendem que há uma pequena diferença entre obstáculos e oportunidades e são capazes de transformar ambos em vantagem.”

Nicolau Maquiavel

RESUMO

O presente trabalho apresenta um sistema de automatização para estabelecimentos como bares e restaurantes, desenvolvido na plataforma móvel com sistema operacional *Android* que se comunica através da rede com um Web Service que trata toda comunicação com a base de dados, sendo esse Web Service e a própria base de dados executados em um servidor local ou na nuvem. O objetivo é substituir o clássico cardápio impresso, diminuindo assim, a necessidade do tratamento com o garçom e, consequentemente, possíveis erros do mesmo na abordagem ao cliente. O sistema é projetado para atender as solicitações do cliente na hora que ele desejar, agilizando o pedido e dando conforto ao cliente.

Palavras-chave: *Web Service, Android,* Atendimento ao Cliente.

ABSTRACT

This paper presents an automation system for establishments such as bars and restaurants, developed on the mobile platform with Android operating system that communicates over the network with a Web service that handles all communication with the database, and this Web Service itself database running on a local server or in the cloud. The objective is to replace the classic printed menu, thereby reducing the need for treatment with the waiter and therefore possible errors in the approach to the same customer. The system is designed to meet the client requests the time he desired, streamlining the application and giving comfort to the client.

Keywords: Web Service, Android, Customer Service.

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE QUADROS

LISTA DE ABREVIATURAS E SIGLAS

ADT	Android Development Tools
AUTOCOM	Feira e congresso Internacionais de Automação Comercial e Tecnologia para o Varejo
HD	High Definition
HTML 5	Hypertext Markup Language, versão 5.
HTTP	HyperText Transfer Protocol
IDE	Integrated Development Environmet
NCE	Núcleo de Computação Eletrônica
NFC	Near Field Communication
SDK	Software Development Kit
SOA	Service Oriented Architecture
SOAP	Simple Object Acces Protocol
UFRJ	Universidade Federal do Rio de Janeiro
UML	Unified Modeling Language
W3C	Word Wide Web Consortium
WSDL	Web Service Description Language
XML	Extensible Markup Language
XSD	XML Schema Definition

SUMÁRIO

1 INTRODUÇÃO

Nos dias de hoje, as empresas estão disputando arduamente os espaços no mercado. Atualmente, é difícil um negócio ter um mercado novo e com potencial grande o suficiente para que empresas possam trabalhar sem entrar no mercado da outra, e cada detalhe que elas proporcionam ou não, pode ser o que vai decidir se ela sobrevive no mercado.

> "Para as organizações, é fundamental ter as competências exigidas pelos desafios do terceiro milênio: o enfrentamento de um mercado extremamente exigente e competitivo. Precisam ter qualidade em vários segmentos, como saúde econômica, tecnologia e produtos diferenciados, entre outros, mas, sobretudo, ter excelência no atendimento." (BENTES, 2012, p. 27).

Seguindo o raciocínio aplicado por Bentes (2012), no segmento do mercado alimentício também não é diferente e um detalhe que uma empresa deste tipo deve ter é o atendimento ao cliente. Atendimento esse que é a principal função de um garçom, mas de acordo com notícia publicada pela AUTOCOM, 15º Feira e Congresso Internacionais de Automação Comercial e Tecnologia para o Varejo, no site da *MASTRAN BUSINESS FAIRS*, estima-se que, um garçom consegue atender de 4 a 5 mesas em um restaurante não automatizado. Ficando com apenas 40% do seu tempo disponível para realizar atendimento e vendas. Desta forma, tendo o garçom como um atendente e vendedor, percebe-se, que a empresa não está disponível ao cliente na maior parte do tempo. Surge ai um problema, como a empresa vai conseguir aumentar sua disponibilidade para venda sem ter que aumentar o efetivo de garçons?

Em busca da solução para este problema, é apresentada neste trabalho, uma forma de saná-los através do uso da tecnologia computacional. A resposta em questão trata-se de um aplicativo para *tablets*, que é capaz de realizar as vendas para o estabelecimento, proporcionando assim um atendimento mais cômodo ao cliente na hora de realizar um pedido e uma disponibilidade maior da empresa para venda.

O aplicativo funciona em qualquer *tablet* que possua o Sistema Operacional *Android* 4.0 ou versões superiores, e trabalha em conjunto com outro aplicativo também para *tablets* com mesmo Sistema Operacional citado anteriormente. Proporcionando uma comunicação rápida entre o cliente e o estabelecimento.

Utilizando como base as informações publicadas no site da AUTOCOM, citadas anteriormente, tomemos o caso de um estabelecimento em que os ocupantes de cada mesa gastem R$ 100,00 a cada hora, e que das 8h de trabalho do garçom, ele esta disponível para

venda e atendimento apenas 3,2h (40% das 8h), seu total de venda será de R$ 320,00 por mesa a cada 8h.

Agora tomamos outro caso, onde a proposta de automação do atendimento seja implantada, utilizando dos mesmos dados da hipótese anterior em que uma mesa arrecada a cada hora R$ 100,00. Temos 100% do tempo disponível para venda e atendimento, graças ao sistema proposto, o que da uma arrecadação total no final das 8h de R$ 800,00.

Levando em consideração que na primeira hipótese cada mesa é de responsabilidade de um garçom e que na segunda hipótese cada mesa é de responsabilidade do aplicativo em um *tablet*, existe um aumento potencial de 150% da arrecadação de cada mesa com o uso do sistema.

1.1 OBJETIVO GERAL

O objetivo deste trabalho é desenvolver um sistema capaz de oferecer um autoatendimento a clientes de bares e restaurantes, permitindo rapidez no atendimento e maior disponibilidade do estabelecimento para venda.

1.2 OBJETIVOS ESPECIFICOS

Este trabalho tem os seguintes objetivos específicos:

- Estudar o sistema operacional para dispositivos móveis *Android*, assim como sua comunicação com base de dados externa;
- Compreender a importância do atendimento ao cliente em organizações e utilizar da tecnologia para auxiliar no desenvolvimento deste aspecto;
- Estudar as tecnologias dos Web Services e suas aplicações.

1.3 ORGANIZAÇÃO DO TRABALHO

Este trabalho está organizado em 5 capítulos. No capítulo 2, Referencial Teórico, encontra-se o embasamento teórico para o desenvolvimento do trabalho. São mostrados conceitos relacionados ao Sistema Operacional *Android*, *Web Service* e Engenharia de *Software*. No capítulo 3, Sistema Desenvolvido, é apresentado o sistema proposto para alcançar o objetivo deste trabalho. Abordando a análise de requisitos e diagramas produzidos

para o desenvolvimento do sistema, além de mostrar o funcionamento do mesmo. No capítulo 4, Testes e Resultados, é mostrado o teste realizado no sistema. Simulando um ambiente real, proporcionando dados para análise de desempenho do sistema em relação ao objetivo proposto. No capítulo 5, Conclusões e Trabalhos Futuros, é apresentada a conclusão do trabalho e indicações de trabalhos futuros.

2 REFERÊNCIAL TEÓRICO

2.1 ANDROID

Desenvolvido pela *Android Inc* e adquirido pelo *Google* em julho de 2005, o *Android* é um sistema operacional baseado em Linux projetado principalmente para dispositivos móveis com tela sensível ao toque como *smartphones* e *tablets*. Em novembro de 2007, foi formado a *Open Handset Alliance*, um consórcio composto inicialmente de 34 empresas.

"A primeira geração de telefones Android foi lançado em outubro de 2008. De acordo com a Gartner, as vendas de telefones baseados em Android nos Estados Unidos aumentaram 707% no primeiro trimestre de 2010, em relação ao ano anterior." (DEITEL et al.,2013, p. 4).

O sucesso do *Android* se evidencia com o passar do tempo, pois já em abril de 2011, 36% da fatia do mercado de *smartphones* nos Estados Unidos era do *Android*, deixando seus concorrentes *iPhone* da *Apple* com 26% e o *Blackberry* com 23% (NIELSEN, 2015). E segundo (HAMANN, 2014), em 2013 o *Android* possuía 84,70% do mercado global, tendo um bilhão de usuários ativos.

Assim como o *Android* foi evoluindo nas vendas ele também evoluiu em funcionalidades a cada versão.

2.1.1 VERSÕES DO ANDROID

O *Android* já teve diversas versões desde seu lançamento em 2008, sem contar as pequenas variações entre as versões.

Segundo (SANTINO, 2013) a primeira versão comercial, *Android* 1.0, lançada em setembro de 2008 já possuía aplicativos do *Google* e vários recursos básicos que na época eram inovadores, como um Media Player, navegador e suporte a *Wi-Fi* e *Bluetooth*.

A primeira atualização do sistema, *Android* 1.1, corrigiu falhas e *bugs* da versão anterior, mas não trouxe grandes inovações. Entre as novidades estava o detalhamento e exibição de *reviews* de locais quando o usuário fizesse uma busca no *Maps*, além de melhorias na interface para realizar chamadas.

Na versão *Android* 1.5, foi incluído gravação e reprodução de vídeos em formato *MPEG-4* e *3GP*. Também foi incluso os *Widgets*, que até hoje é marca registrada do sistema.

Conforme as versões seguintes, (1.6, 2.0 a 2.1, 2.2 a 2.2.3, 2.3 a 2.3.7), foram surgindo outros recursos foram adicionados ao *Android*, como por exemplo o suporte ao *HTML5* no navegador na versão 2.0 e suporte a resolução *HD* (*Higt Definition*) e tecnologia *NFC* (*Near Field Communication*) nas versões 2.3 a 2.3.7.

Porém, até a versão 3.0 o *Android* tinha certas dificuldades em executar em *tablets*. Por esse motivo, decidiram criar uma versão que atendesse melhor ao tamanho maior de tela dos *tablets* e outras características particulares desse tipo de dispositivo, foi ai que surgiu o *Android* 3.0, que é uma versão feita particularmente para *tablets*. Porem, na versão 4.0, o *Android* passou a não fazer mais distinção de dispositivos tornando a nova versão capaz de ser executado tanto em *tablets* como em *smartphones*. Informações estas que estão disponíveis nas páginas oficiais das versões.

Da versão 4.0 a versão atual 5.0 o que se destaca é o desempenho acelerado multitarefa que surgiu na versão 4.4.

"O Android 4.4 eleva o desempenho do sistema a um novo patamar, otimizando a memória e aperfeiçoando a tecnologia toutchscreen para que responda com mais rapidez e precisão do que nunca." (ANDROID, 2015).

2.1.2 DISTRIBUIÇÃO DE DISPOSITIVOS ATIVOS POR VERSÃO

Nesta seção vemos como está a distribuição das versões do *Android* pelos dispositivos ativos até 7 dias antes a 4 de Agosto de 2015.

Na tabela 1 é possível ver que a versão 4.0 mesmo sendo mais recente já possibilita um grande volume de dispositivos compatíveis, uma vez que um aplicativo desenvolvido com a versão mínima 4.0 executa em versões superiores também.

Version	Codename	API	Distribution
2.2	Froyo	8	0.3%
2.3.3 - 2.3.7	Gingerbread	10	4.6%
4.0.3 - 4.0.4	Ice Cream Sandwich	15	4.1%
4.1.x	Jelly Bean	16	13.0%
4.2.x		17	15.9%
4.3		18	4.7%
4.4	KitKat	19	39.3%
5.0	Lollipop	21	15.5%
5.1		22	2.6%

Tabela 1 – Distribuição das Versões *Android* por Dispositivos Ativos.
Fonte: (DEVELOPER.ANDROID, 2015a).

A Tabela 1 fornece dados importantes a quem deseja desenvolver algum tipo de aplicativo para *Android*. Outro ponto importante para um desenvolvedor *Android* é a IDE (*Integrated Development Environmet*) que ele utilizará.

2.1.3 IDE'S DE DESENVOLVIMENTO DE APLICATIVOS PARA ANDROID

A IDE, sigla em inglês de Ambiente de Desenvolvimento Integrado, é um *software* que apoia o desenvolvimento de outros *softwares* objetivando agilização deste processo. Reúne ferramentas e características que visam a maior produtividade dos desenvolvedores.

"Android Studio é a IDE oficial para o desenvolvimento de aplicativos Android, baseado na IntelliJ IDEA." (DEVELOPER.ANDROID, 2015b).

Possui editor de código poderoso e inteligente capaz de realizar, refatorar e analisar códigos avançados que auxilia na maior produtividade do desenvolvedor além de ter um gerenciador de dispositivos virtual que fornece perfis de dispositivos pré-definidos para dispositivos Android comuns. (DEVELOPER.ANDROID, 2016).

Antes do surgimento do *Android Studio*, a IDE recomendada para desenvolvimento com *Android* era o *Eclipse*. Como mostra (DEITEL et al., 2013).

Ainda segundo (DEITEL et al., 2013), para o uso da IDE *Eclipse*, é necessário a instalação do SDK (*Software Development Kit*) do *Android* e o *plugin* ADT (*Android Development Tools*) para *Eclipse*. Sendo que o SDK fornece as ferramentas necessárias para desenvolver, testar e depurar os aplicativos *Android*, e o *plugin* ADT, permite usar as ferramentas do SDK do *Android* para desenvolver aplicativos *Android* no IDE Eclipse.

Com a chegada do *Android Studio*, o desenvolvedor *Android* ganha facilidades no desenvolvimento dos seus trabalhos, permitindo dar mais atenção na lógica do seu sistema.

2.2 WEB SERVICE

Utilizado para integrar sistemas e possibilitar a comunicação entre aplicações diferentes, o *Web Service* veio como uma tentativa de organizar o caos gerado pela grande quantidade de aplicativos, fornecedores e plataformas diferentes.

O *Web Service* possibilita que sistemas desenvolvidos em linguagens diferentes, se comuniquem através das redes de computadores.

> "As bases para a construção de um Web service são os padrões XML e SOAP. O transporte dos dados é realizado normalmente via protocolo HTTP (o padrão não determina o protocolo de transporte). Os dados são transferidos no formato XML, encapsulados pelo protocolo SOAP." (OFICINADANET, 2007).

Como o Analista de Suporte Sérgio Cruz, do NCE/UFRJ (Núcleo de Computação Eletrônica da Universidade Federal do Rio de Janeiro) explica em artigo publicado no site do NCE/UFRJ. O *Web Service* é descrito como uma arquitetura baseada em serviços, conhecido como SOA (*Service Oriented Architecture*). Os componentes dessa arquitetura representam uma coleção de serviços que se comunicam através da troca de mensagens XML (*Extensible Markup Language*).

2.2.1 COMPONENTES DA ARQUITETURA DO WEB SERVICE

O Primeiro Componente citado pelo autor no artigo é o Provedor de Serviço, ele é responsável pela descrição e publicação do serviço no registro dos serviços, é responsável também por descrever as informações da chamada do serviço. Estas informações estão

contidas em um documento XML escritas na linguagem padrão WSDL (*Web Service Description Language*).

Segundo componente, chamado de Consumidor do Serviço, fica responsável por descobrir um serviço, obter a sua descrição e usá-la para conectar a um provedor a fim de invocar um serviço *web*.

Terceiro componente é o Registro dos Serviços, que é onde se mantem um diretório com informações sobre os serviços.

2.2.2 WEB SERVICES DESCRIPTION LANGUAGE (WSDL)

"[...] é um padrão de mercado para descrever Web Services de forma a eliminar ao máximo a necessidade de comunicação entre as partes envolvidas em uma integração de dados." (OLIVEIRA, 2015).

Submetida a W3C (*Word Wide Web Consortium*), uma das principais organizações de padronização da *WEB*, o WSDL é o arquivo XML padronizado com todas as informações necessárias para o uso do *Web Service*.

A estrutura principal de um WSDL, como mostra publicação do site w3schools, parece com o código na Figura 1 a seguir.

```
<definitions>

<types>
  data type definitions........
</types>

<message>
  definition of the data being communicated....
</message>

<portType>
  set of operations......
</portType>

<binding>
  protocol and data format specification....
</binding>

</definitions>
```

Figura 1 – Exemplo de código WSDL. Fonte: (W3SCHOOLS, 2015).

2.2.3 EXTENSIBLE MARKUP LANGUAGE (XML)

"XML, do inglês eXtensible Markup Language, é uma linguagem de marcação recomendada pela W3C para a criação de documentos com dados organizados hierarquicamente, tais como textos, banco de dados ou desenhos vetoriais." (PEREIRA, 2009).

O XML está em um nível de abstração onde é considerada uma metalinguagem, ou seja, ela é capaz de permitir a criação de outras linguagens. Esta criação se da pela definição de novos conjuntos de tags e pelas regras que um documento XML deve seguir. (DÉCIO, 2015).

Como visto nas seções anteriores, o XML tem papel importantíssimo no funcionamento do *Web Service*, pois ele está presente não somente no WSDL que define o *Web Service*, mas também na troca de mensagens entre o *Web Service* e o cliente. Mensagens estas que são protocoladas pelo SOAP.

2.2.4 SIMPLE OBJECT ACCESS PROTOCOL (SOAP)

"SOAP é um protocolo elaborado para facilitar a chamada remota de funções via internet, permitindo que dois programas se comuniquem de uma maneira tecnicamente muito semelhante à invocação de páginas WEB." (SANTANA, 2015).

Segundo (ROMMEL, 2003) o SOAP é um dos elementos fundamentais do *Web Service*, mesmo não sendo necessário o conhecimento do funcionamento do mesmo para criar e consumir um *Web Service*. O entendimento geral do protocolo é útil para lidar com situações de erros e problemas com a interoperabilidade de plataformas no uso de *Web Service*.

A estrutura do protocolo é bem simples, (DANTAS, 2007) divide em três elementos básicos, sendo eles:

- Envelope. Deve estar presente em toda mensagem SOAP, é o elemento raiz do documento XML, ele contem declarações de *namespace* e também atributos adicionais como o que define o estilo de codificação (*encoding style*).
- *Header*. É um cabeçalho opcional, carrega informações adicionais como por exemplo se a mensagem deve ser processada por um determinado nó intermediário. Quando utilizado, deve ser o primeiro elemento do Envelope.

- *Body*. Elemento obrigatório que contem a informação a ser transportada para o destino final, este elemento pode conter um elemento opcional chamado de *Fault*, usado para carregar mensagens de status e erros retornados pelos nós ao processarem a mensagem.

"A especificação do SOAP define o suporte aos tipos de dados baseado no XSD, a especificação do XML Schema Definition." (ROMMEL, 2003).

> "Tipos definidos pelo usuário também são aceitos, de forma que é possível formar estruturas de dados a partir dos dados primitivos oferecidos pela XSD. Mais uma grande vantagem do uso do SOAP, que aceita qualquer tipo que possa ser representado por um XSD Schema." (ROMMEL, 2003).

Para entender melhor o papel do SOAP e do WSDL no *Web Service*, é necessário ver como essas definições e protocolos trabalham na execução do *Web Service*.

2.2.5 FUNCIONAMENTO DO WEB SERVICE

Segundo (SANTOS, 2015) o ciclo de funcionamento do *Web Service* que utiliza o SOAP se faz da seguinte maneira. Um cliente envia uma requisição HTTP através da rede para o servidor que disponibiliza o *Web Service*, então, este servidor devolve para o cliente o arquivo WSDL. Uma vez que o cliente processa o XML do WSDL, ele envia uma requisição HTTP contendo um arquivo XML encapsulado pelo SOAP com as informações necessárias para que o servidor execute a operação desejada. Após a execução da operação solicitada, o servidor envia de volta ao cliente um arquivo XML encapsulado pelo SOAP informando o resultado da operação que deve conter desde uma simples mensagem até um conjunto complexo de informações. Na Figura 2 este ciclo é melhor visualizado.

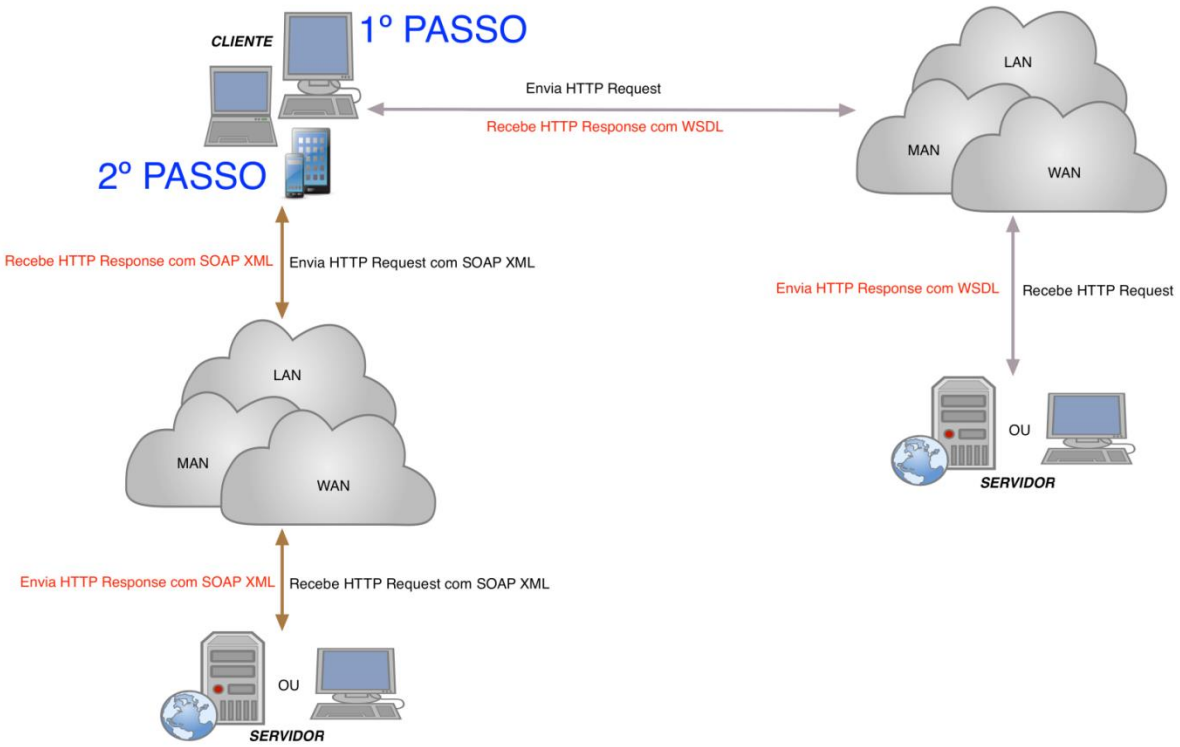

Figura 2 - Diagrama de Funcionamento do Web Service. Fonte: (SANTOS, 2015).

2.3 ENGENHARIA DE SOFTWARE

"O software desempenha um papel fundamental em várias áreas de negócios seja para controle e segurança de informações, controle de processos, apoio a decisões como, também, entretenimento." (HIRAMA, 2012, p.2).

Distribuindo o produto mais importante de nossa era, a informação, o Software transforma dados pessoais de modo que possa ser mais úteis num determinado contexto. Gerenciando as informações comerciais para aumentar a competitividade por exemplo. (PRESSMAN, 2011).

Com os softwares se tornando cada vez mais complexos, problemas de qualidade e atendimento de prazos e custos foram aparecendo. Da necessidade de tratar o software como um produto de engenharia, para fazer frente aos novos desafios, nasceu a Engenharia de Software. (HIRAMA, 2012).

De acordo com (SOMMERVILE, 2005), a Engenharia de Software se ocupa de todos os aspectos da produção de software, indo dos estágios iniciais de especificações do sistema até quando o sistema entra em operação.

"É importante notar que a Engenharia de Software segue os mesmos princípios de uma disciplina de engenharia tradicional, baseada em uma relação adequada de custo/benefício do produto, que não falhe e que seja eficiente." (HIRAMA, 2012, p. 7).

A Engenharia de *Software* nos oferece apoio no desenvolvimento de *software*, com técnicas e modelos que permite aos desenvolvedores uma melhor organização do seu trabalho e uma melhor gerência do tempo.

2.3.1 METODOLOGIAS PARA DESENVOLVIMENTO DE SOFTWARE

Metodologias no desenvolvimento de *software* podem ser consideradas como conjuntos de métodos e ou processos que devem ser seguidos na construção de um *software*.

Segundo (CAETANO, 2009), as metodologias servem para não tornar a construção do *software* um verdadeiro caos. Porém, o problema é que dependendo do projeto, os métodos tradicionais podem deixar os desenvolvedores amarrados a requisitos desatualizados que não corresponde as reais necessidades do cliente. A flexibilidade e a facilidade de mudar o rumo são qualidades muito valiosas para serem deixadas de lado em um momento de crise econômica ou mercados altamente competitivos.

> "Também conhecidas como Metodologias orientadas a planejamento, as Metodologias Clássicas dominaram a forma de desenvolvimento de softwares até o inicio da década de 90, Entretanto, estas metodologias devem ser aplicadas apenas em situações em que os requisitos do sistema são estáveis e os requisitos futuros são previsíveis." (REIS, 2015).

Existem outros tipos de metodologias, são as chamadas metodologias ágeis. (CAETANO, 2009) explica que ao contrário das metodologias tradicionais, essas metodologias ágeis, oferecem ao desenvolvedor total flexibilidade e aproximam a equipe de tecnologia da informação a o usuário final do software, seja ele um cliente interno ou externo. Com esse tipo de tecnologia, a homologação do projeto é feita em etapas, resultando em tempos de espera mais curtos e capacidade de promover alterações rapidamente.

Segundo (ESTEFFEN, 2012), as metodologias ágeis *Extreme Programming* (XP), *Scrum, Lean Development, Feature-Driven Development* (FDD), *Kanban*, RUP *e OpenUP* são as metodologias ágeis mais comuns. E que o *Scrum* é de longe o *framework* mais utilizado por ser o mais simples e de fácil adoção e adaptação.

2.3.2 SCRUM

O *Scrum* tem como ideia principal, controlar processos empíricos mantendo o foco na entrega de valor de um negócio no menor tempo possível. É utilizado para o gerenciamento de projetos ágeis, que apesar de muito comum na área de desenvolvimento de *software*, pode ser utilizado também para o planejamento, gerenciamento e desenvolvimento de qualquer produto, principalmente por ser um *framework* iterativo e incremental. (CRUZ, 2015).

Por mais que o *Scrum* forneça ferramentas e maneiras de se conduzir o desenvolvimento ele não pode ser considerado uma metodologia.

> "A diferença entre metodologia e framework é que uma metodologia fornece praticamente tudo que é necessário para condução de um projeto. Ela é mais completa que um framework, já que diz o que fazer e como fazer. Já o framework apenas indica qual a trajetória, mas não indica exatamente como fazer, existindo a possibilidade de ser empregado juntamente com outros processos e técnicas. Funciona praticamente como um "esqueleto", que fornece uma estrutura básica como suporte." (BARBOSA, 2015).

De acordo com (DESENVOLVIMENTOAGIL, 2015), os projetos no *Scrum* são divididos em ciclos tipicamente mensais chamados de *Sprint*. O *Sprint* representa uma caixa do tempo, onde no qual um conjunto de atividades deve ser executado. As funcionalidades a serem implementadas em um projeto são mantidas em uma lista que é conhecida como *Product Backlog*. Ao iniciar cada *Sprint*, faz-se uma reunião de planejamento (*Sprint Planning Meeting*) na qual o *Product Owner* prioriza os itens do *Product Backlog* e a equipe seleciona as atividades que ela será capaz de implementar durante o *Sprint* que se inicia. As tarefas alocadas em um *Sprint* são transferidas do *Product Blacklog* para o *Sprint Blacklog*. A cada dia de uma *Sprint* a equipe faz uma breve reunião chamada de *Daily Scrum* com o objetivo de disseminar o conhecimento sobre o que foi feito no dia anterior, identificar impedimentos e priorizar o trabalho do dia que se inicia. No final de uma *Sprint* a equipe apresenta as funcionalidades implementadas em uma *Sprint Review Meeting*, finalmente faz-se uma *Sprint Retrospctive* e a equipe parte para o planejamento do próximo *Sprint*, reiniciando o ciclo.

Na Figura 3 é possível visualizar este funcionamento.

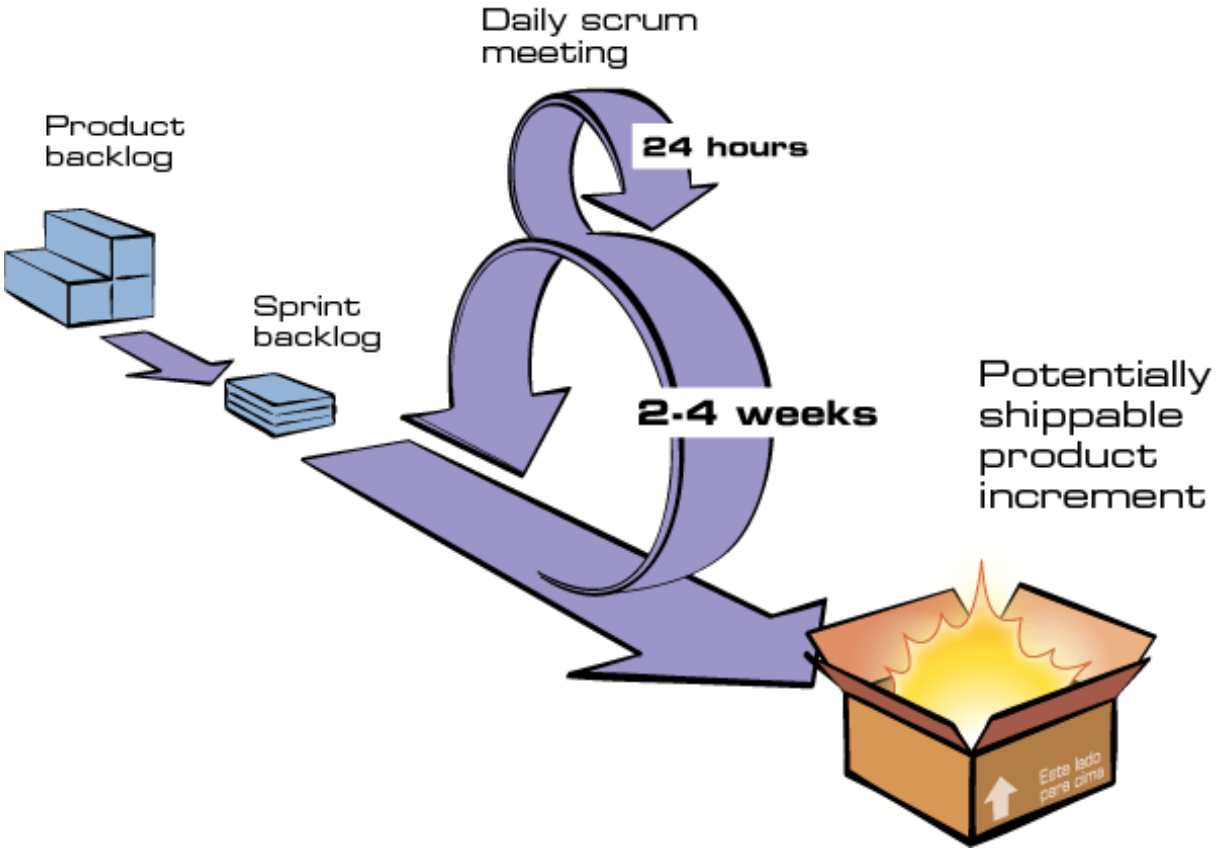

Figura 3 – Ciclo de uma Sprint no *Scrum*. Fonte: (DESENVOLVIMENTOAGIL, 2015).

Além das metodologias de desenvolvimento, existem outros recursos proporcionados pela Engenharia de *Software* que auxiliam na construção de *software*. Como por exemplo, a UML (*Unified Modeling Language*).

2.3.3 UML (Unified Modeling Language)

"A UML (Unified Modeling Language, ou, traduzindo, Linguagem de Modelagem Unificada) é uma linguagem que serve para especificar, construir, visualizar e documentar os artefatos de um sistema de software." (RAMOS, p.8, 2006).

Esta linguagem de modelagem não é um método de desenvolvimento, ela tem o papel de auxiliar a visualizar o desenho e comunicação entre objetos. Permitindo que desenvolvedores visualizem os produtos de seu trabalho em diagramas padronizados.

Segundo (RAMOS, 2006), a UML possui as seguintes particularidades:

- Semântica e notação para tratar um grande número de tópicos atuais de modelagem.

- Semântica para tratar tópicos de modelagem futura, relacionados em particular com a tecnologia de componentes, computação distribuída, *frameworks* e Internet.
- Mecanismos de extensão que permitem que futuras aproximações e notações de modelagem possam continuar a se desenvolver sobre UML.
- Semântica e sintaxe que facilitam a troca de modelos entre ferramentas distintas

A UML providencia alguns diagramas que são divididos em estruturais e comportamentais. Dentre os comportamentais se destacam os diagramas de Caso de Uso e o Diagrama de Atividades.

2.3.4 DIAGRAMA DE CASOS DE USO

Usado para documentar o que o sistema faz de acordo com o ponto de vista do usuário, o diagrama de casos de uso em outras palavras descreve as principais funcionalidades do sistema e a interação dessas funcionalidades com os usuários do mesmo.

> "O modelo de casos de uso permite capturar os requisitos de um sistema por meio do detalhe de todos os cenários que os usuários podem acessar. Os casos de uso, mas do que iniciar a modelagem de requisitos de um sistema, conduzem todo processo de desenvolvimento." (RAMOS, p.65, 2006).

Os casos de usos podem ser aplicados para captar o comportamento pretendido do sistema que está sendo desenvolvido, sem ser necessário especificar como esse comportamento é implementado.

"Um caso de uso é representado graficamente mediante uma forma oval com uma frase que representa uma determinada ação. A frase deve ser escrita na voz ativa, com um verbo infinitivo." (RAMOS, p.65, 2006). Como por exemplo, "Cadastrar Produto", "Manter Pedidos", "Validar usuário" e "Excluir produto".

2.3.5 DIAGRAMA DE ATIVIDADE

Sendo um gráfico de fluxo, o diagrama de atividade mostra o fluxo de controle de uma atividade para outra.

Segundo (BEZERRA, 2015), um diagrama de atividade é um tipo especial de diagrama de estados de um objeto. Ao contrário dos diagramas de estados, que são orientados

a eventos, diagramas de atividades são orientados a fluxos de controle. Seus elementos podem ser divididos em dois grupos: os utilizados para representar fluxos de controle sequenciais e os utilizados para representar fluxos de controle paralelos.

Para (PRESSMAN, 2011), o diagrama de atividades complementa o caso de uso através de uma representação gráfica do fluxo de interação em um cenário específico. Este diagrama usa retângulos com cantos arredondados para representar determinada função do sistema, setas para representar o fluxo através do sistema, losangos para representar uma decisão com ramificação e as linhas horizontais cheias indicam as atividades paralelas que estão ocorrendo.

Na Figura 4, podemos visualizar os elementos descritos anteriormente.

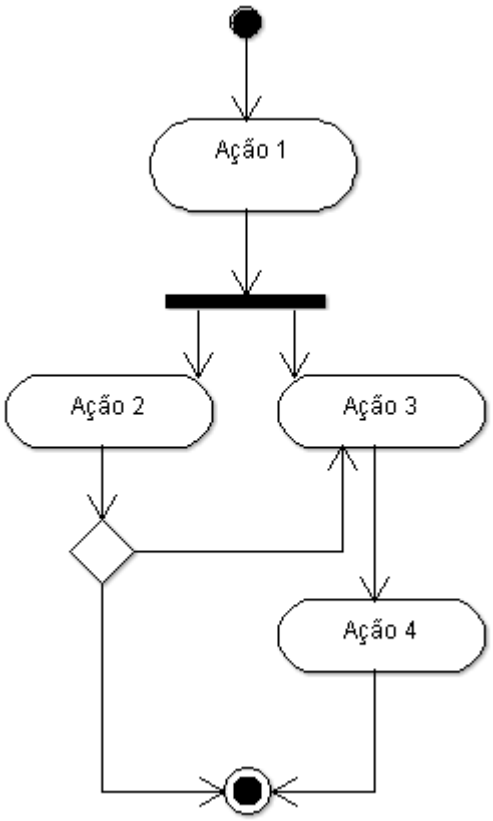

Figura 4 – Exemplo Diagrama de Atividade. Fonte: (O AUTOR).

3. SISTEMA DESENVOLVIDO

Neste capitulo, é apresentado o sistema desenvolvido para alcançar o objetivo do trabalho.

3.1 VISÃO GERAL DO SISTEMA

O sistema é dividido em 3 módulos, sendo eles o APP Cliente, o APP Cozinha e o *Web Service*.

Os APP Cliente e APP Cozinha são aplicativos para dispositivos móveis, mais especificamente para *tablets*. No mercado existe alguns vários Sistemas Operacionais para estes tipos de dispositivos, os mais populares são o *iOS* da *Apple* e o *Android*. Sendo que o *iOS* está presente nos dispositivos da *Apple*, enquanto o *Android* está presente em vários modelos de dispositivos de diferentes fabricantes.

"Apesar de ter entrado no mercado um pouco depois do iOS, o Android é o sistema operacional mais utilizado em portáteis." (HAMANN, 2014).

Essa popularidade do sistema operacional *Android* amplia a quantidade de aparelhos capazes de executar os aplicativos desenvolvidos, permitindo uma variedade maior de dispositivos proporcionando ao cliente um conjunto de opções capaz de variados custos na aquisição dos mesmos para implantação do sistema em seu estabelecimento.

3.1.1 FUNCIONAMENTO DO SISTEMA

Os pedidos feitos pelo APP Cliente são enviados através da rede local ao *Web Service* que processará a informação e deixará disponível para o APP Cozinha utilizar.

A Figura 5 mostra de maneira bem simples como o sistema funciona, toda solicitação enviada pelo APP Cliente ou APP Cozinha é respondida pelo *Web Service*.

Figura 5 – Ilustração Simples do Funcionamento do Sistema. Fonte: (O AUTOR).

Sistemas como esse, geram certo grau de dificuldade no planejamento das atividades a serem realizadas. Porém, existem modelos e técnicas utilizadas que agiliza o processo de desenvolvimento e a criação de funcionalidades no sistema.

3.2 METODOLOGIA DE DESENVOLVIMENTO E REQUISITOS DO SISTEMA

Pela necessidade de um desenvolvimento rápido do sistema, a metodologia utilizada no desenvolvimento do mesmo foi o *Scrum*. Apesar de alguns autores afirmarem que o *Scrum*, por ser um *framework*, não pode ser considerado uma metodologia. Ele possibilita que o desenvolvimento do sistema possa ser dinâmico e com grande capacidade de se adaptar a mudanças, motivo esse que nos levou a sua utilização.

Por ter uma lógica de negócio não comum a outros sistemas existentes no mercado, a metodologia de desenvolvimento escolhida foi de suma importância para a conclusão do sistema. Pois possibilitou que as funcionalidades fossem testadas na medida em que o sistema era desenvolvido, permitindo a verificação da necessidade de outras funções que completassem as mesmas.

Tendo em vista que o sistema possivelmente cresceria durante seu desenvolvimento e até mesmo após a conclusão da versão de apresentação, ficou decidido que os levantamentos de requisitos funcionais e não funcionais, fossem originados de uma base idealizada para suprir as necessidades do atendimento automatizado em bares e restaurantes.

No Quadro 1 podemos conhecer os requisitos funcionais com seu identificador, descrição e as dependências que esses possuem.

Quadro 1 – Requisitos Funcionais. Fonte: (O AUTOR).

Identificador	Descrição	Dependente de
RF01	O sistema deve permitir o cadastro de Produto.	
RF02	O sistema deve permitir abrir comanda, informando a mesa referente.	RF07
RF03	O sistema deve permitir alterar disponibilidade de um produto.	RF01
RF04	O sistema deve ser capaz de realizar pedidos, registrando a mesa e o produto, além de gerenciar e alterar o status desse pedido.	
RF05	O sistema deve calcular o consumo de cada comanda.	RF02
RF06	O sistema deve informar ao cliente o valor atualizado de seu consumo, assim como informações dos produtos consumidos.	RF05
RF07	O sistema não deve permitir o cadastro de mesas, uma vez que a quantidade de mesas é pré-fixada na base de dados.	
RF08	O sistema deve permitir o cancelamento do pedido pelo cliente, desde que o status do mesmo seja “aguardando”.	RF04
RF09	O sistema deve permitir que o cliente encerre sua comanda, desde que não tenha pedido em aberto.	RF04, RF05, RF08

O Quadro 2 mostra os requisitos não funcionais cada um com seu identificador, uma descrição, a categoria e escopo a qual o requisito pertence e suas dependências.

Quadro 2 – Requisitos não Funcionais. Fonte: (O AUTOR).

Identificador	Descrição	Categoria	Dependente de
RNF01	O sistema evitará que uma comanda seja aberta com um número de mesa já utilizada por outra comanda.	**Confiabilidade**	
RNF02	A persistência das informações deve ser implementada em um Sistema Gerenciador de Bancos de Dados Relacionais (SGBDR) livre.	**Manutenibilidade**	
RNF03	O sistema deve ser restrito ao estabelecimento.	**Operacionais**	

O Quadro 3 mostra as regras de negócios, cada um com seu identificador, uma descrição e suas dependências.

Quadro 3 – Regras de Negócios. Fonte: (O AUTOR).

Identificador	Descrição	Dependente de
RN01	Os produtos serão tipificados em dois tipos, refeição e bebidas.	

Após levantamento dos requisitos funcionais, não funcionais e regras de negócio. É possível construir diagramas comportamentais da UML (*Unified Modeling Language*) que servem para demonstrar melhor o sistema.

3.3 UML (*Unified Modeling Language*)

A UML consiste de certo número de elementos gráficos que se combinam para formar diagramas com o objetivo de apresentar múltiplas visões do sistema.

Dentre os principais diagramas, se destacam os de casos de uso e de atividades.

3.3.1 DIAGRAMA DE CASO DE USO

O sistema apresentado neste trabalho diferencia os usuários em dois tipos: cliente e estabelecimento.

O Quadro 4 mostra os atores identificados no sistema e uma breve descrição de quem são e o que podem fazer.

Quadro 4 – Atores. Fonte: (O AUTOR).

Ator	**Descrição**
Cliente	São os usuários que utilizarão o sistema como forma de autoatendimento. Este usuário consegue visualizar os produtos disponíveis, realizar pedidos, acompanhar seus pedidos, cancelar pedido e encerrar comanda.
Estabelecimento	Este usuário é um funcionário da empresa que utilizara o sistema para gerenciar produtos e pedidos realizados pelo cliente.

O diagrama de caso da Figura 6 mostra as funcionalidades do sistema de forma geral, com os dois tipos de usuários possíveis representados pelos atores cliente e estabelecimento.

No diagrama, os atores são representados por bonecos, casos de uso por elipses e a associação entre eles é representada por uma linha.

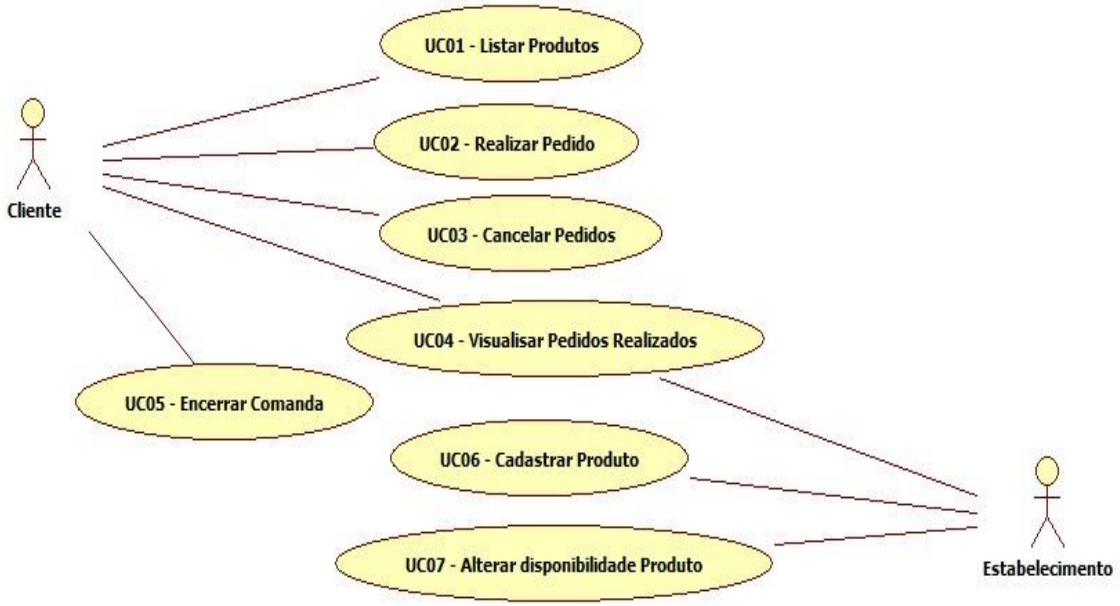

Figura 6 – Diagrama de Caso de Uso. Fonte: (O AUTOR).

A seguir é mostrado detalhes dos casos de uso da Figura 6, explicitando os atores que estão associados ao caso de uso, uma descrição e os requisitos que esses estão relacionados.

- **UC01 - Listar Produtos:**
  - **Ator:** Cliente.
  - **Descrição:** Pode-se visualizar os produtos disponíveis para pedido.
  - **Requisitos:** RF03, RN01.
- **UC02 - Realizar Pedido:**
  - **Ator:** Cliente.
  - **Descrição:** Pode-se efetuar um pedido de um produto desejado.
  - **Requisitos:** RF04.
- **UC03 - Cancelar Pedidos:**
  - **Ator:** Cliente.
  - **Descrição:** Pode-se cancelar um pedido que esteja com seu status em "aguardando".
  - **Requisitos:** RF08.
- **UC04 - Visualizar Pedidos Realizados:**
  - **Ator:** Cliente, Estabelecimento.

- **Descrição:** Para o cliente é possível visualizar seus pedidos, para o estabelecimento é possível visualizar todos os pedidos em aberto.
- **Requisitos:** RF04,

- **UC05 - Encerrar Comanda:**
  - **Ator:** Cliente.
  - **Descrição:** Pode-se pedir o fechamento da comanda, desde que não tenha pedidos em aberto.
  - **Requisitos:** RF09.
- **UC06 - Cadastrar Produto:**
  - **Ator:** Estabelecimento.
  - **Descrição:** Pode-se cadastrar produtos informando nome, descrição, valor e tipo.
  - **Requisitos:** RF01.
- **UC07 - Alterar Disponibilidade Produto:**
  - **Ator:** Estabelecimento.
  - **Descrição:** Pode-se alterar a disponibilidade do produto, permitindo o cliente visualizar ou não este produto.
  - **Requisitos:** RF03.

Além do diagrama de casos de uso, pode-se explicar melhor o fluxo de informação dentro do sistema através do diagrama de atividades.

3.3.2 DIAGRAMA DE ATIVIDADE

O objetivo deste diagrama é mostrar o fluxo de atividades em um único processo e como uma atividade depende da outra.

Serão apresentados alguns diagramas de atividade que representam as principais atividades dos dois tipos de usuários do sistema, o usuário “cliente” e o usuário “estabelecimento”.

A Figura 7 mostra o diagrama de atividade que representa a realização de um pedido. Seguindo esses passos:

1. Listagem dos Produtos disponíveis, essa ação mostra ao usuário todos os produtos que ele pode pedir.

2. Após a escolha do produto na lista, o próximo passo é selecionar esse produto.
3. Com o produto selecionado, é confirmada a realização do pedido, dando fim a esta atividade.

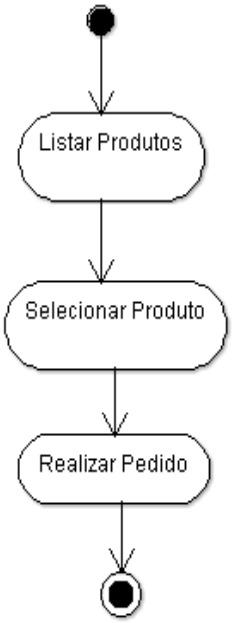

Figura 7 – Diagrama de Atividade Realiza Pedido. Fonte: (O Autor).

Na Figura 8 é possível visualizar o diagrama de atividade que representa o cancelamento de um pedido. Está atividade é realizada pelo usuário cliente seguindo os seguintes passos:

1. Primeiramente chama-se a opção cancelar pedido.
2. Logo em seguida é listado todos os pedidos feitos pelo usuário.
3. Após ver a lista dos pedidos o usuário seleciona o pedido que deseja cancelar.
4. Quando selecionado o pedido, o sistema verificará o status do pedido.
5. Caso o status do pedido seja “aguardando”, ele poderá ser cancelado.
6. Caso o status do pedido não seja “aguardando”, ele não poderá ser mais cancelado.

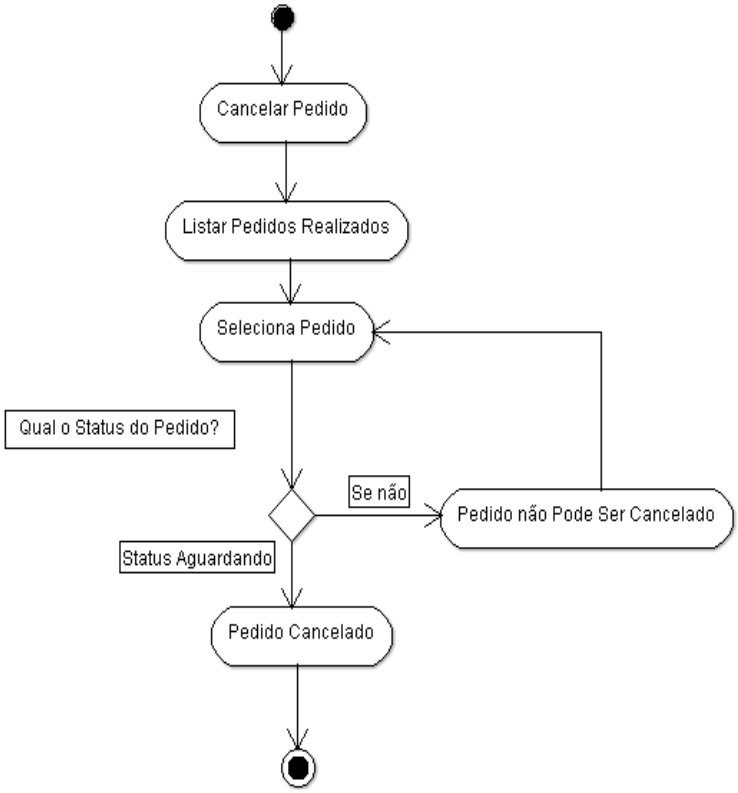

Figura 8 – Diagrama de Atividade Cancelar Pedido. Fonte: (O AUTOR).

A Figura 9 mostra o diagrama de atividade que representa o gerenciamento do pedido pelo usuário "estabelecimento". Para realizar essa atividade, o usuário segue os seguintes passos:

1. Acessar a lista com os pedidos realizados pelos clientes.
2. Seleciona o pedido que ele deseja alterar seu status.
3. O sistema verificará qual o status atual do pedido.
4. Caso for aguardando, é alterado o status do pedido para "em preparo".
5. Caso o status do pedido seja "em preparo", é alterado o status para "finalizado". O que indica que o pedido já foi atendido.

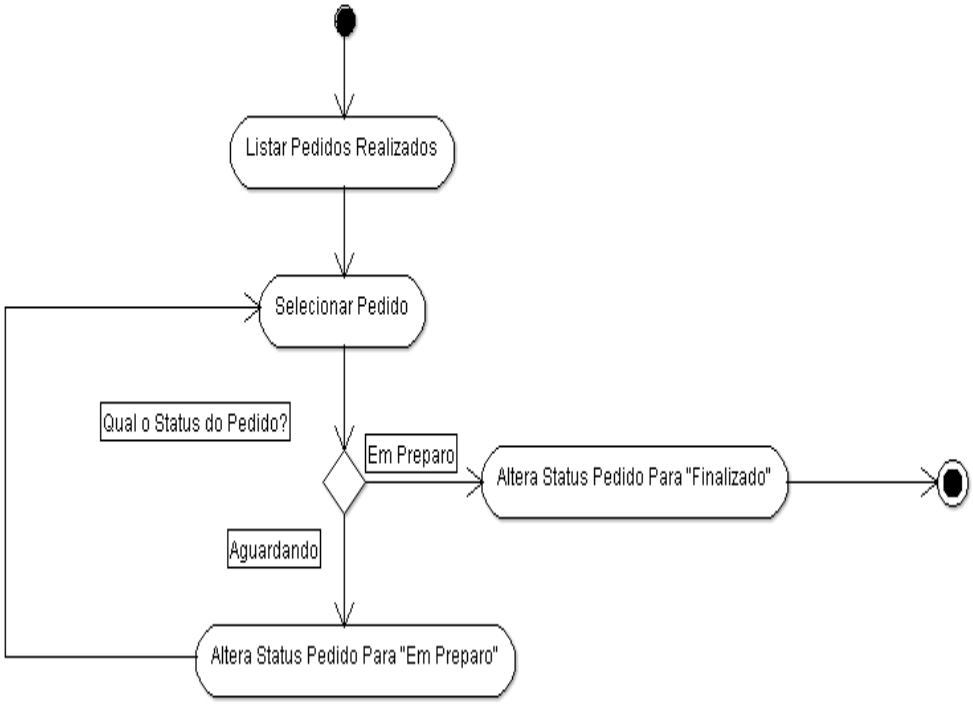

Figura 9 – Diagrama de Atividade Gerenciamento de Pedidos. Fonte: (O AUTOR).

Esses diagramas apresentados dão uma visão do que o sistema é capaz de fazer, permitindo conhecer detalhes do sistema.

3.4 APP CLIENTE

Ao ser iniciado, a primeira tela que ira aparecer solicitara o número da mesa correspondente aquele dispositivo. O garçom ao levar o dispositivo ate a mesa já terá feito esse comando, como mostra a figura 10.

Figura 10 – Tela de Informe da Mesa. Fonte: (O AUTOR).

Sendo o responsável pelo trato com o cliente, atentamos para que o aplicativo fosse de fácil compreensão pelo cliente desde o primeiro contato. Disponibilizando opções simples e objetivas no sistema.

A Figura 11 mostra a tela principal do sistema, onde no canto inferior esquerdo já é possível visualizar o valor atual do consumo e na parte superior acessar as opções do sistema com as funcionalidades que vão auxiliar o estabelecimento a ter um atendimento melhor ao cliente, permitindo o cliente ter informações sobre seu consumo, produtos e pedidos já realizados por ele instantaneamente.

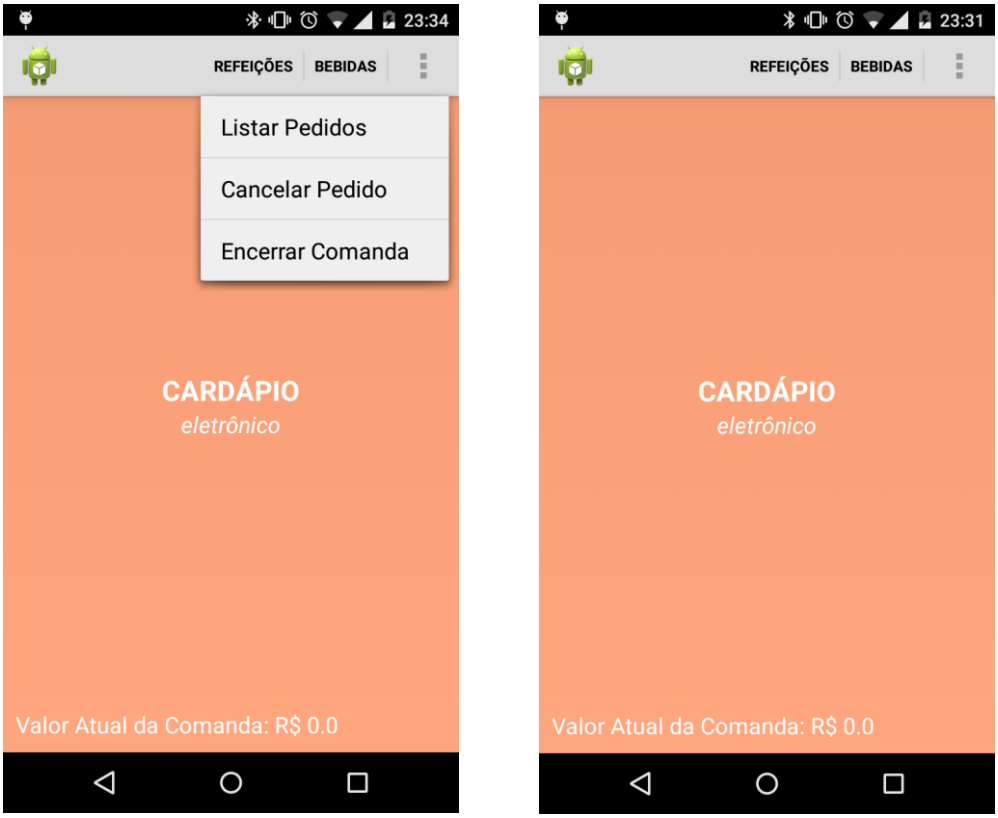

Figura 11 – Tela Principal. Fonte: (O AUTOR).

3.4.1 FUNCIONALIDADES DO APLICATIVO

Acessando o menu “Refeições” ou “Bebidas”, o sistema mostrará ao cliente as opções do cardápio disponíveis. Uma vez que o produto só é visível no cardápio caso haja disponibilidade confirmada pelo APP Cozinha que veremos adiante.

Na tela de apresentação do cardápio, os produtos são expostos em uma lista contendo as informações do mesmo, sendo elas o nome, descrição e preço dos produtos. Os produtos são divididos nas categorias Refeição e Bebidas, como mostra a Figura 12.

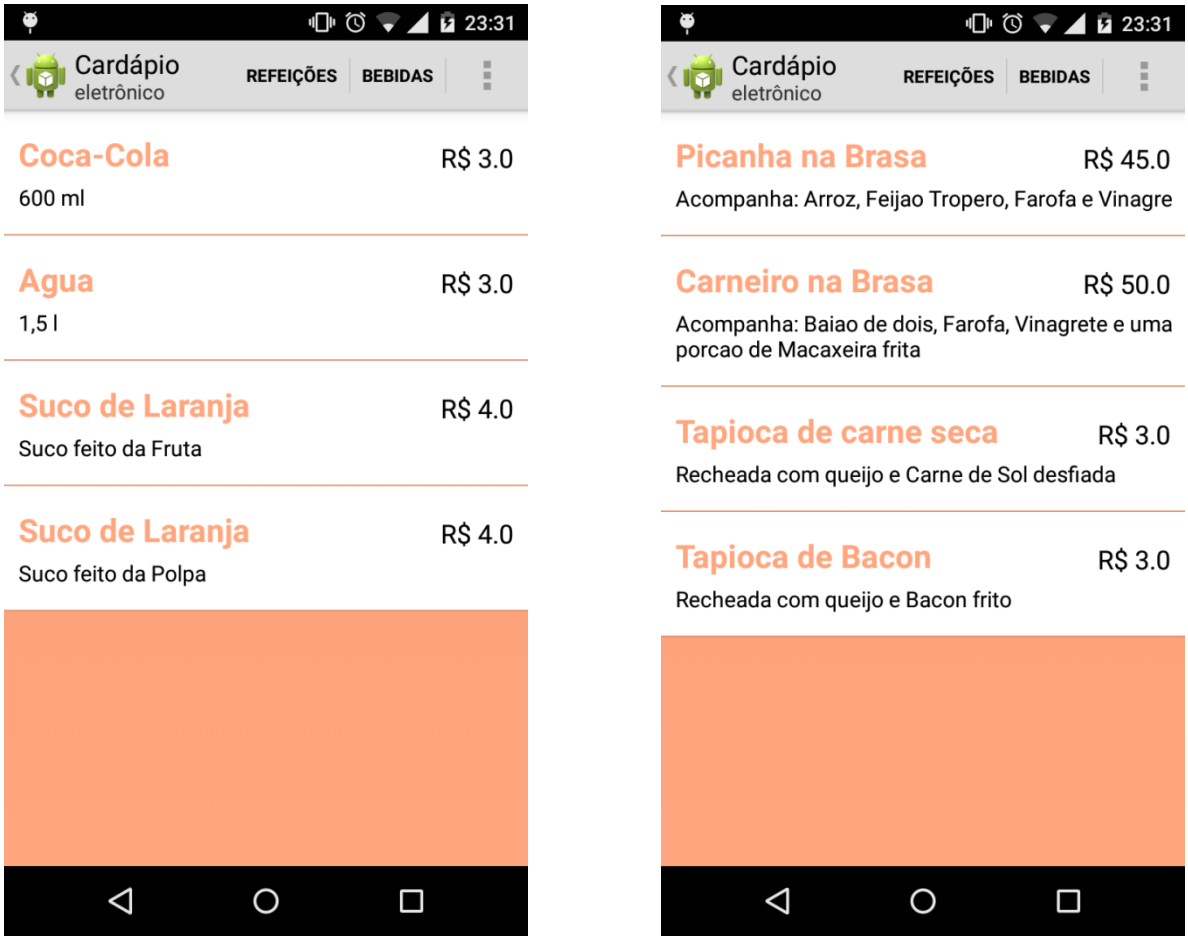

Figura 12 – Telas de Apresentação dos dois Tipos de Produtos. Fonte: (O AUTOR).

No menu “Listar Pedidos” o cliente visualizara as informações de todos seus pedidos feitos, inclusive o status do pedido que no sistema é organizado em 3 estados, “Aguardando”, “Em Preparo” e “Finalizado”.

Para cancelar um pedido ao acessar a opção “Cancelar Pedido” no menu, este pedido tem que estar com seu status em aguardando, pois a partir do momento que o mesmo entrar em preparo não a mais como o cliente cancelar.

Ao escolher a opção encerrar comanda o sistema verificara se não existe pedidos pendentes, caso haja, o cliente só poderá encerrar a comanda quando esses pedidos forem finalizados.

3.5 APP COZINHA

Dentre outras funções, este aplicativo é responsável principalmente pelo monitoramento da lista de pedidos, ao ser inicializado já exibirá na tela a lista de todos os pedidos feitos, possibilitando o usuário alterar o status de cada pedido com um simples toque. Como mostra a Figura 13.

Figura 13 – Tela Pedidos Realizados. Fonte: (O AUTOR).

Na Figura 13 vemos o aplicativo aberto com a lista de pedidos, ao tocar em um item da lista o usuário mudará o status do pedido e para melhor usabilidade é alterado a cor do item passando a ficar azul, sinalizando que aquele pedido está em preparo como mostram as Figura 14 e Figura 15.

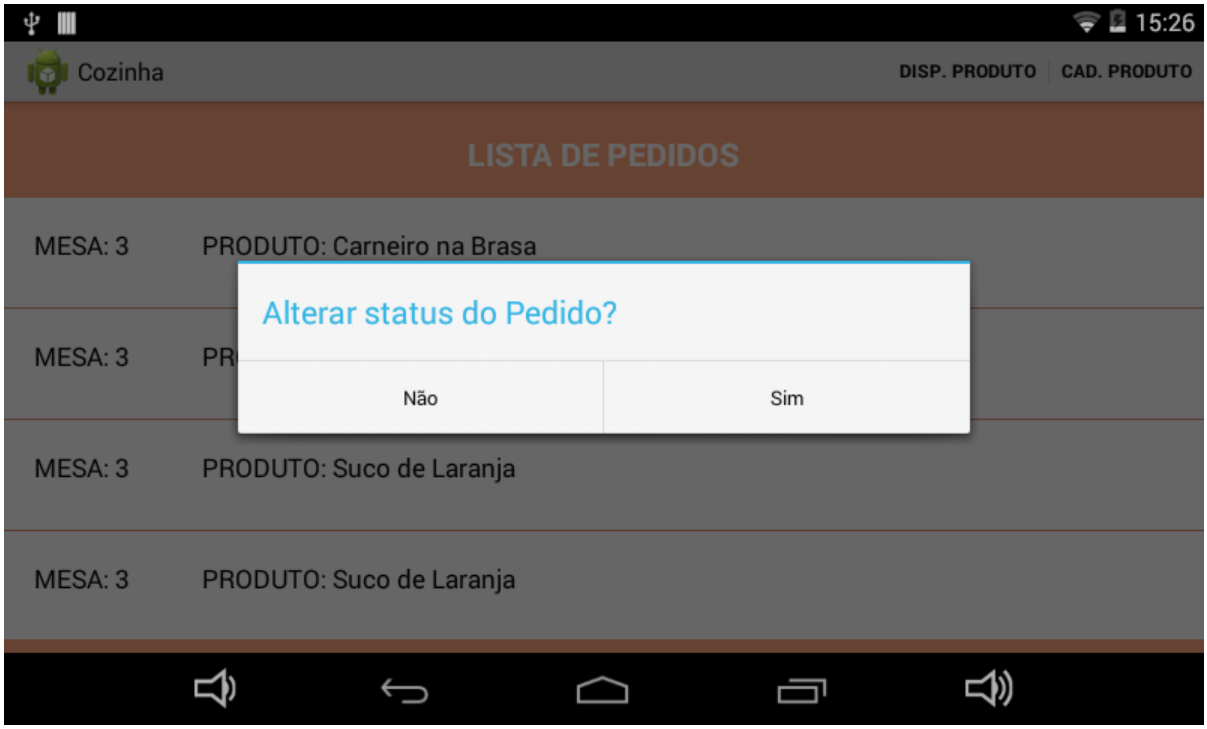

Figura 14 – Mudando Status do Pedido. Fonte: (O AUTOR).

Figura 15 – Pedido com Status Alterado. Fonte: (O AUTOR).

3.5.1 FUNCIONALIDADES DO APLICATIVO

As funcionalidades do aplicativo são bem simples, além de gerenciar os pedidos que estão sendo feitos pelos clientes, o aplicativo também conta com as funções de informar quais produtos estão disponíveis para pedidos e realizar também o cadastro dos produtos.

Para alterar a disponibilidade do produto, basta um toque na opção “Disp. Produto”, que será possível visualizar na tela todos os produtos cadastrados. E com mais um simples toque no produto que deseja mudar a disponibilidade, o produto não será mais visto pelos clientes.

Na opção “Cad. Produto” é possível cadastrar os produtos que o estabelecimento fornece informando o nome, descrição, valor e o tipo. Conforme mostra a Figura 16.

Figura 16 – Tela de Cadastro de Produtos. Fonte: (O AUTOR).

Os APP’s Cliente e Cozinha, conseguem compartilhar informações graças ao Web Service, que disponibiliza serviços na rede.

3.6 WEB SERVICE

O papel do *Web Service* neste trabalho é fazer a comunicação entre os aplicativos e a base de dados, uma vez que o Sistema Operacional *Android* não faz comunicação direta com uma base de dados externa.

Escrito na linguagem de programação Java, nosso *Web Service* é hospedado em um servidor *Tomcat* na rede local.

Para acessar o *Web Servise*, precisamos do endereço do WSDL, que é o documento XML que contem as informações do que está disponível no *Web Service* e como os dados estão dispostos.

Na Figura 17, veremos um trecho do WSDL relacionado aos Produtos, neste pedaço de código XML vemos a definição do tipo complexo de dados que esse *Web Service* trabalha.

Figura 17 – Trecho 1 Código WSDL. Fonte: (O AUTOR).

A Figura 18 mostra outro trecho do mesmo código XML do WSDL citado anteriormente. Neste trecho, vemos como um dos métodos do *Web Service* é descrito para que possa ser utilizado.

Figura 18 – Trecho 2 Código WSDL. Fonte: (O AUTOR).

O código da Figura 18 mostra que para o método “cadastraProduto”, os parâmetros de entrada e seus respectivos tipos são “nome” do tipo *String*, “desc” do tipo *String*, “valor” do tipo *Float* e “tipo” que é *String* também. Além disso, informa o tipo de retorno deste método que no caso é *Boolean*.

3.6.1 COMUNICAÇÃO ENTRE OS APLICATIVOS E O WEB SERVICE

A comunicação entre os aplicativos e o Web Service é encapsulada pelo protocolo SOAP e enviada na rede pelo protocolo HTTP.

Veremos na Figura 19 um exemplo de uma mensagem SOAP enviada ao Web Service.

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/" xmlns:ws="http://WS.tcc.com.br">
   <soapenv:Header/>
   <soapenv:Body>
      <ws:cadastraProduto>
         <!--Optional:-->
         <ws:nome>Suco Abacaxi</ws:nome>
         <!--Optional:-->
         <ws:desc>Feito da fruta / 250ml</ws:desc>
         <!--Optional:-->
         <ws:valor>4</ws:valor>
         <!--Optional:-->
         <ws:tipo>bebida</ws:tipo>
      </ws:cadastraProduto>
   </soapenv:Body>
</soapenv:Envelope>
```

Figura 19 – Código Da Mensagem SOAP Enviada. Fonte: (O AUTOR).

No código da Figura 19 é mostrado a mensagem SOAP enviada pelo aplicativo ao *Web Service* para o cadastro de um produto. Após esta mensagem ser enviada, o aplicativo aguardará a resposta do *Web Service* se o método tiver retorno. No caso deste exemplo é aguardada uma resposta do tipo *Boolean* para informar se o cadastro foi efetuado com sucesso ou não.

A Figura 20 mostra a resposta SOAP dada pelo *Web Service*, que é uma valor *Boolean* indicando que o cadastro foi efetuado com sucesso.

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/">
   <soapenv:Body>
      <ns:cadastraProdutoResponse xmlns:ns="http://WS.tcc.com.br">
         <ns:return>true</ns:return>
      </ns:cadastraProdutoResponse>
   </soapenv:Body>
</soapenv:Envelope>
```

Figura 20 – Código Da Resposta da Mensagem SOAP . Fonte: (O AUTOR).

4. TESTES E RESULTADOS

O sistema foi testado em laboratório de maneira que pudesse proporcionar uma aproximação do mundo real.

Com a ajuda de 4 voluntários aleatórios, criou-se um restaurante hipotético. Onde os 4 voluntários atuaram como clientes do estabelecimento, representados pelos pseudônimos cliente A, cliente B, cliente C e cliente D. Um 5º personagem foi adicionado ao teste, onde atuou como o funcionário do estabelecimento, recebendo o pseudônimo cozinheiro.

Aos clientes A, B, C e D, foi dada a liberdade de decisão de em que momento eles sentariam nas mesas.

Não se especificou tempo para duração do teste, pois assim como eles tiveram a liberdade de sentarem, também tinham a liberdade de decidirem em que momento iriam embora.

Para simular o tempo de atendimento do pedido, foi dado um tempo padrão de 20 minutos a produtos do tipo refeição e 05 minutos a produtos do tipo bebidas.

Ao final do teste foi feito questionário com os participantes que atuaram como clientes. O Quadro 5 representa o modelo do questionário aplicado.

Quadro 5 – Modelo Questionário Pós Teste. Fonte: (O AUTOR).

Cliente:			
O que achou de fazer um pedido sem a necessidade de chamar um garçom?	( ) Ruim	( ) Indiferente	( ) Bom
Quanto à aparência do sistema. No que se refere a facilidade de uso. O Que achou?	( ) Ruim	( ) Indiferente	( ) Bom
Você como cliente. Qual seu sentimento quanto ao atendimento recebido?	( ) Ruim	( ) Indiferente	( ) Bom
Você gostaria de encontrar esse sistema algum dia em uma situação real?	( ) Não	( ) Indiferente	( ) Sim
Você achou que o teste proposto se aproximou de uma situação real?	( ) Não	( ) Prefiro não opinar	( ) Sim

Com os dados colhidos através do questionário puderam-se criar gráficos onde foi possível visualizar níveis de satisfação dos clientes.

O primeiro gráfico que podemos visualizar é representado pela Figura 21, nesta figura verificamos que 75% dos participantes definiram como “Bom” a experiência de ser atendido sem ter que esperar alguém para atendê-los.

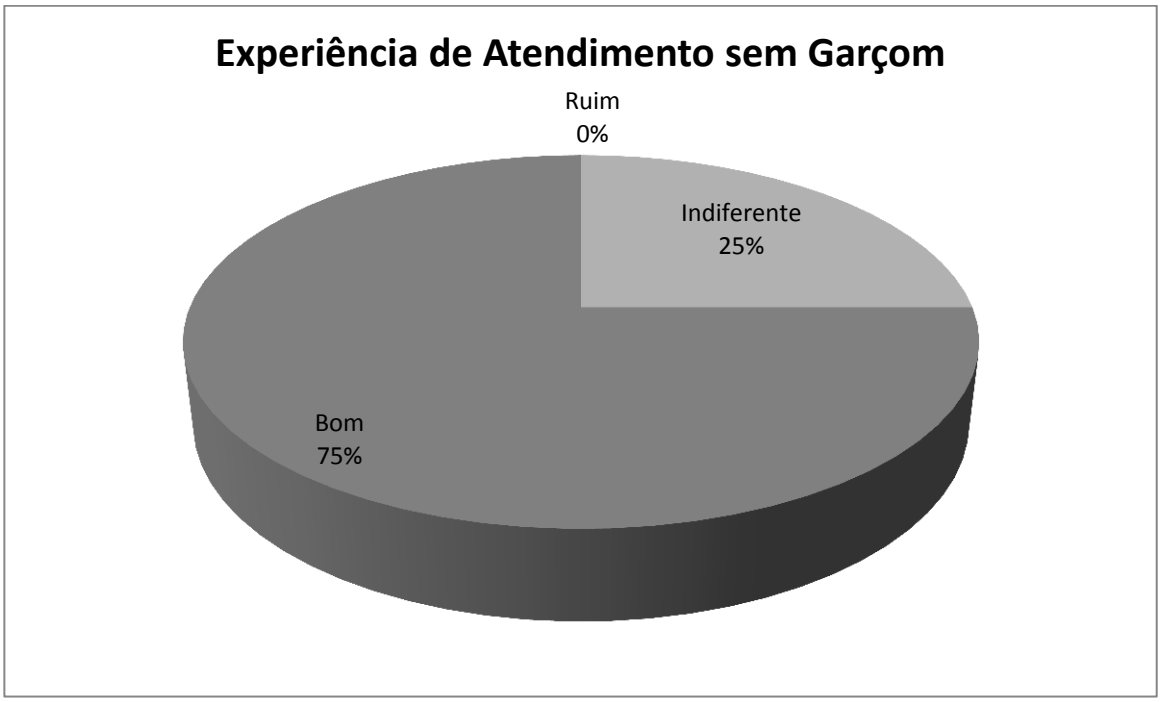

Figura 21 – Gráfico Experiência de Atendimento sem Garçom. Fonte: (O AUTOR).

Na Figura 22 pode-se verificar a experiência dos usuários no que se refere às questões de facilidade de compreensão e uso do sistema. Onde 50% dos participantes afirmaram estar indiferentes a esta questão, 25 % afirmaram que o sistema é de fácil compreensão e 25% afirmaram que o sistema não é tão fácil de usar.

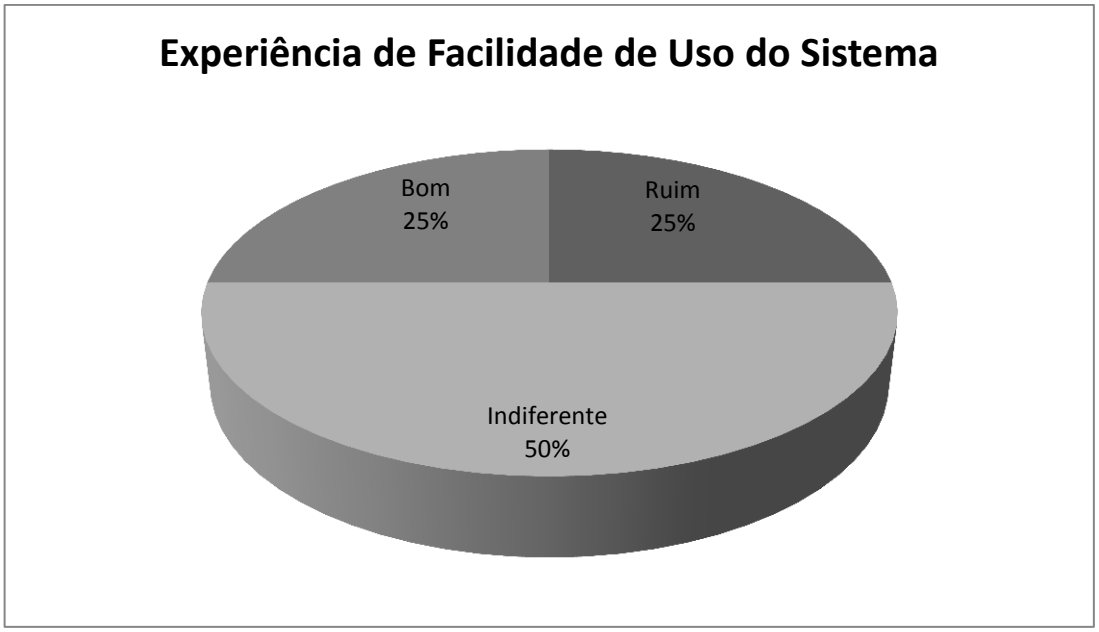

Figura 22 – Gráfico Experiência de Facilidade de Uso do Sistema . Fonte: (O AUTOR).

O gráfico da Figura 23 mostra as experiências dos usuários relativas ao bom atendimento, onde 75% afirmaram ter tido um bom atendimento e 25% afirmaram ser indiferente a esta questão no sentido de não se importarem com este aspecto.

Figura 23 - Gráfico Experiência de Atendimento. Fonte: (O AUTOR).

A Figura 24 mostra um gráfico onde é expressa a vontade dos clientes de encontrarem esse sistema em uma situação real.

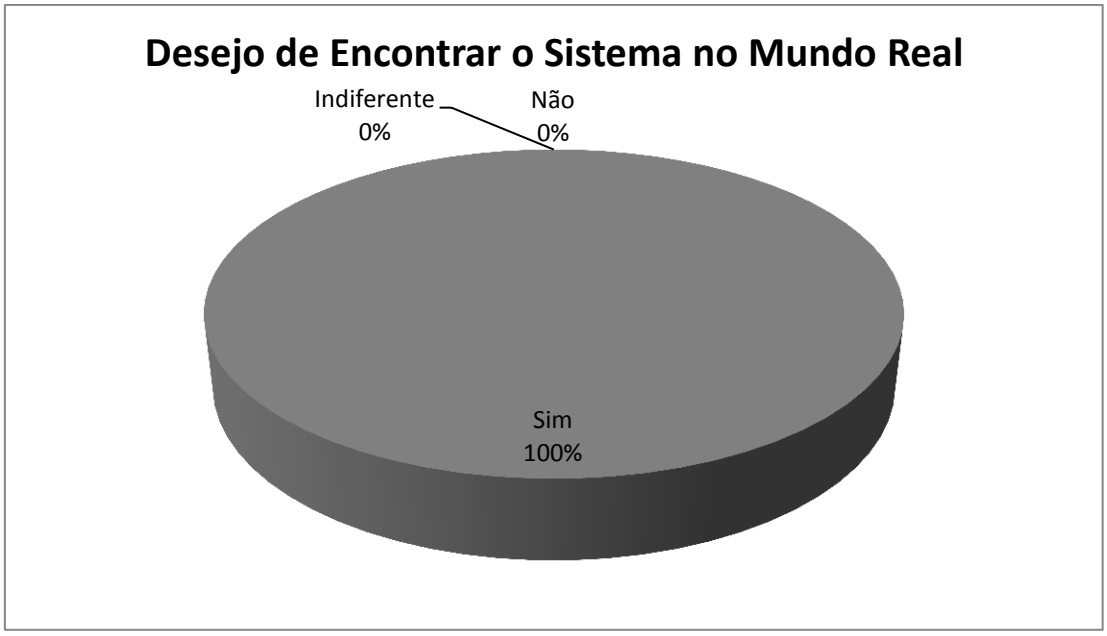

Figura 24 – Gráfico Desejo de Encontrar o Sistema no Mundo Real. Fonte: (O AUTOR).

O último gráfico construído a partir do questionário é mostrado na Figura 25. Este gráfico mostra a opinião dos clientes em relação à validação do teste, onde 75% dos participantes afirmaram que o teste proposto se aproximou do mundo real e 25% preferiram não opinar.

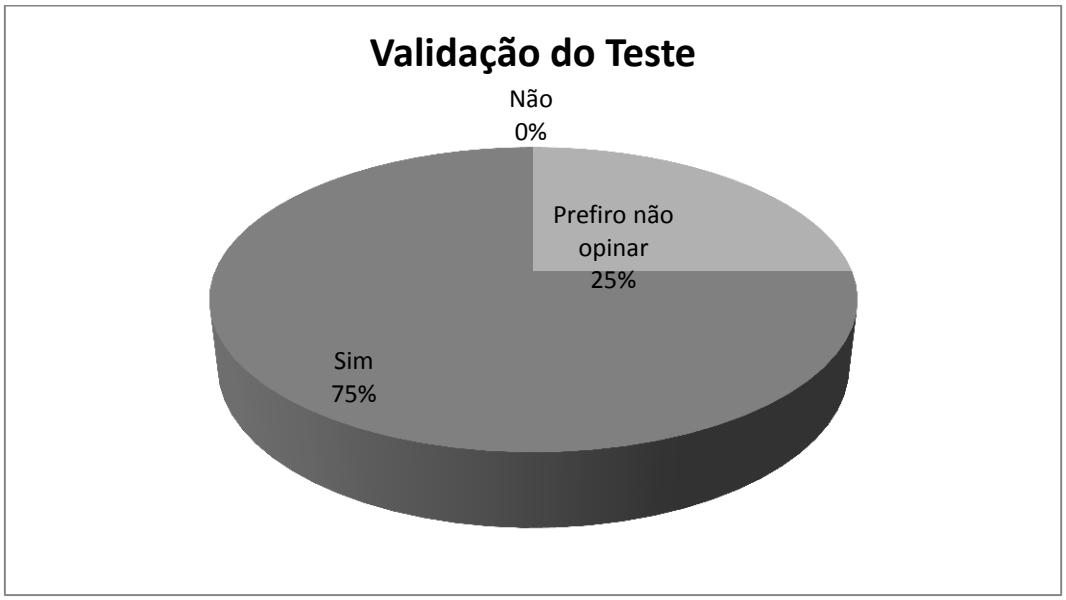

Figura 25 – Gráfico Validação do Teste. Fonte: (O AUTOR).

Após realização de teste do sistema e análise dos dados obtidos. Pode-se tirar conclusões sobre a eficiência do sistema em relação ao objetivo que se deseja alcançar com o desenvolvimento do mesmo.

5. CONCLUSÕES E TRABALHOS FUTUROS

Percebendo que o bom atendimento ao cliente é um dos pilares para qualquer empreendimento, uma vez que o cliente é o principal gerador de lucro para as empresas, foi enxergada a possibilidade de melhorar este pilar em empreendimentos dos tipos bares e restaurantes.

Nas pesquisas realizadas para fundamentar este trabalho, tecnologias como o *Android* e *Web Service* mostraram-se capazes de permitir que o sistema fosse desenvolvido. Porém, a comunicação entre essas duas tecnologias gerou certas dificuldades.

Após superar as dificuldades encontradas, foi possível criar uma versão de apresentação do sistema capaz de atender aos propósitos do trabalho. Como mostra o teste realizado, onde foi possível constatar os níveis de satisfação dos usuários, classificados em, “Ruim, Indiferente e Bom”. Além disso, o teste possibilitou visualizações de trabalhos futuros.

Para trabalhos futuros, destaca-se o melhoramento na interface dos aplicativos, funcionalidades extras ao sistema capazes de fazerem conexões com redes sociais permitindo, por exemplo, os usuários compartilharem momentos no estabelecimento, criação de aplicativo que permita que o usuário faça seus pedidos de qualquer lugar e não somente no estabelecimento.

6. REFERÊNCIAS BIBLIOGRÁFICAS

ALBINADER, Jorge Abilio; LINS, Rafael Dueire. **Web Services em Java.** Rio de Janeiro: Brasport, 2006.

ANDROID. **KitKat 4.4:** Inteligente, simples e totalmente seu. Disponível em: <https://www.android.com/intl/pt-BR_br/versions/kit-kat-4-4/>. Acesso em: 5 nov. 2015.

BARBOSA, João Otávio Chrisóstomo. **Entendendo o Scrum e conduzindo projetos com sucesso.** Disponível em: <http://www.devmedia.com.br/entendendo-o-scrum-e-conduzindo-projetos-com-sucesso/33724>. Acesso em: 21 dez. 2015.

BENTES, Otavio. **Atendimento ao Cliente.** Curitiba: IESDE Brasil S.A., 2012.

BERGHER, Ricardo. **Android x iOS: qual o melhor sistema operacional para tablete?.** Disponível em: <http://www.zoom.com.br/tablet-ipad/deumzoom/android-x-ios-qual-o-melhor-sistema-operacional-para-os-tablets>. Acesso em: 22 dez. 2015.

BEZERRA, Eduardo. **Princípios De Análise E Projeto de Sistemas Com Uml.** 3. ed. São Paulo: Elsevier, 2015.

BLOG.MAKESYS. **Definindo seu software: o que são requisitos funcionais e não funcionais?.** Disponível em: <http://blog.makesys.com.br/definindo-seu-software-o-que-sao-requisitos-funcionais-e-nao-funcionais>. Acesso em: 20 set. 2015.

CAETANO, Rodrigo. **Metodologias de desenvolvimento: qual a mais adequada?.** Disponível em: <http://computerworld.com.br/gestao/2009/08/05/metodologias-de-desenvolvimento-qual-a-mais-adequada>. Acesso em: 20 dez. 2015.

CRUZ, Fábio. **Scrum e Agile em Projetos Guia Completo:** Conquiste sua certificação e aprenda a usar métodos ágeis no seu dia a dia. Rio de Janeiro: Brasport, 2015.

CRUZ, Sérgio Manuel Serra. **Serviços Web – Uma breve introdução (Parte I).** Disponível em: <http://www.nce.ufrj.br/conceito/artigos/2005/01p1-1.htm>. Acesso em: 10 nov. 2015.

DANTAS, Daniel Chaves Toscano. **Simple Object Acces Protocol (SOAP).** Disponível em: < http://www.gta.ufrj.br/grad/07_2/daniel/>. Acesso em: 8 dez. 2015.

DÉCIO, Otávio. **Entendendo XML.** Disponível em: <http://www.portaldaprogramacao.com/artigos2.asp?n=204>. Acesso em: 1 dez 2015.

DEITEL,Paul; etal. **Android para programadores:** Uma abordagem baseada em aplicativos. Porto Alegre: Bookman, 2013.

DESENVOLVIMENTOAGIL. **SCRUM.** Disponível em: <http://www.desenvolvimentoagil.com.br/scrum/>. Acesso em: 21 dez. 2015.

DEVELOPER.ANDROID. **Dashboards.** Disponível em: <http://developer.android.com/intl/pt-br/about/dashboards/index.html>. Acesso em: 5 nov. 2015.

DEVELOPER.ANDROID. **Android Studio Overview.** Disponível em: <https://developer.android.com/intl/pt-br/tools/studio/index.html>. Acesso em: 5 nov. 2015.

DEVELOPER.ANDROID. **Android Studio.** Disponível em: <http://developer.android.com/intl/pt-br/sdk/index.html>. Acesso em: 2 jan. 2016.

DEVELOPER.ANDROID. **Honeycomb.** Disponível em: <http://developer.android.com/intl/pt-br/about/versions/android-3.0-highlights.html>. Acesso em: 28 dez. 2015.

GSMARENA. **Nielsen: Androi overtakes iOS in desirability, sales in the US.** Disponível em: <http://www.gsmarena.com/nielsen_android_overtakes_ios_in_desirability_sales_in_the_us-news-2565.php>. Acesso em: 1 dez. 2015.

HAMANN, Renan. **iOS, Android e Windows Phone: números dos gigantes comparados [infográfico].** Disponível em: <http://www.tecmundo.com.br/sistema-operacional/60596-ios-android-windows-phone-numeros-gigantes-comparados-infografico.htm>. Acesso em: 5 nov. 2015.

HIRAMA, Kechi. **Engenharia de Software:** Qualidade e Produtividade com Tecnologia. Rio de Janeiro: Elsevier, 2012.

JAVAFREE.UOL. **Web Services. Construindo, disponibilizando e acessando Web Services via J2SE e J2ME.** Disponível em: <http://javafree.uol.com.br/artigo/871485/Web-Services-Construindo-disponibilizando-e-acessando-Web-Services-via-J2SE-e-J2ME.html>. Acesso em: 1 out. 2015.

MACÊDO, Diego. **Web Services.** Disponível em: <http://www.diegomacedo.com.br/web-services/>. Acesso em: 28 out. 2015.

MASTRANFAIRS. **Por que automatizar um restaurante?.** Disponível em: <http://www.mastranfairs.com/feiraautocom/NoticiasTexto.aspx?idNoticia=43#.VqrX2FIbKl1>. Acesso em: 3 nov. 2015.

MEYER, Maximiliano. **A história do Android.** Disponível em: <https://www.oficinadanet.com.br/post/13939-a-historia-do-android>. Acesso em: 10 nov. 2015.

MSDN.MICROSOFT. **Web Services.** Disponível em: <https://msdn.microsoft.com/pt-br/library/cc564893.aspx>. Acesso em: 25 set. 2015.

NIELSEN. **Android Leads in U.S. Smartphone Market Share and Data Usage.** Disponível em: <http://www.nielsen.com/us/en/insights/news/2011/android-leads-u-s-in-smartphone-market-share-and-data-usage.html>. Acesso em 25 nov. 2015.

OFICINADANET. **O que é Web Service?.** Disponível em: <https://www.oficinadanet.com.br/artigo/447/o_que_e_web_service>. Acesso em: 8 nov. 2015.

OLIVEIRA, Maycon Fábio. **WSDL: Simplifique a integração de dados via Web Service.** Disponível em: <http://www.devmedia.com.br/wsdl-simplifique-a-integracao-de-dados-via-web-service/30066>. Acesso em 1 dez. 2015.

PEREIRA, Ana Paula. **O que é XML?.** Disponível em: <http://www.tecmundo.com.br/programacao/1762-o-que-e-xml-.htm>. Acesso em: 1 dez 2015.

PRESSMAN, Roger S.. **Engenharia de Software:** Uma abordagem Profissional. 7. ed. Porto Alegre: AMGH, 2011.

RAMOS, Ricardo Argenton. **Treinamento prático em UML.** São paulo: Digerati Books, 2006.

RECKZIEGEL, Mauricio. **Entendendo os WebServices.** Disponível em: <http://imasters.com.br/artigo/4245/web-services/entendendo-os-webservices/>. Acesso em: 9 set. 2015.

REIS, Daniel Fonseca. **Conceitos básicos sobre metodologias Ágeis para Desenvolvimento de Software (Metodologias Clássicas x Extreme Programming).** Disponível em: <http://www.devmedia.com.br/conceitos-basicos-sobre-metodologias-ageis-para-desenvolvimento-de-software-metodologias-classicas-x-extreme-programming/10596>. Acesso em: 20 dez. 2015.

ROMMEL, Marcus. **Simple Object Acces Protocol:** Entendendo o Simple Object Acces Protocol (SOAP). Disponível em: < http://wiki.pge.ce.gov.br/images/0/0b/SOAP.pdf>. Acesso em: 8 dez. 2015.

SAMPAIO, Cleuton. **SOA e Web Services em Java.** Rio de Janeiro: Brasport, 2006.

SANTANNA, Mauro. **SOAP e WebService.** Disponível em: <http://www.linhadecodigo.com.br/artigo/38/soap-e-webservices.aspx>. Acesso em: 5 dez 2015.

SANTINO, Renato. **Android já teve 10 versões diferentes; relembre a evolução do sistema.** Disponível em: <http://olhardigital.uol.com.br/noticia/android-ja-teve-10-versoes-diferentes-relembre-a-evolucao-do-sistema/35801>. Acesso em: 5 nov. 2015.

SANTOS, Allan Carneiro. **Web Services em aplicações Android e iOS.** Disponível em: <http://www.devmedia.com.br/web-services-em-aplicacoes-android-e-ios/28901>. Acesso em: 8 dez. 2015.

SOAWEBSERVICES. **Como funcionam os Web Services.** Disponível em: <http://www.soawebservices.com.br/como-funciona.aspx>. Acesso em: 20 set. 2015.

SOMMERVILLE, Ian. **Engenharia de Software.** 6. ed. São Paulo: Pearson Prentice Hall, 2005.

STEFFEN, Juliana Berossa. **O que são essas tais de metodologias Ágeis?.** Disponível em: <https://www.ibm.com/developerworks/community/blogs/rationalbrasil/entry/mas_o_que_s_c3_a3o_essas_tais_de_metodologias__c3_a1geis?lang=en>. Acesso em 20 dez. 2015.

W3SCHOOLS. **XML WSDL.** Disponível em: <http://www.w3schools.com/xml/xml_wsdl.asp>. Acesso em: 10 dez. 2015.

TERMO DE AUTORIZAÇÃO PARA PUBLICAÇÃO DIGITAL NA BIBLIOTECA "JOSÉ ALBANO DE MACEDO"

Identificação do Tipo de Documento

( ) Tese

( ) Dissertação

(X) Monografia

( ) Artigo

Eu, **Melksedek Amorim Santos Góis**, autorizo com base na Lei Federal nº 9.610 de 19 de Fevereiro de 1998 e na Lei nº 10.973 de 02 de dezembro de 2004, a biblioteca da Universidade Federal do Piauí a divulgar, gratuitamente, sem ressarcimento de direitos autorais, o texto integral da publicação **Automatizando Atendimento em Bares e Restaurantes com o uso de Dispositivos Móveis** de minha autoria, em formato PDF, para fins de leitura e/ou impressão, pela internet a título de divulgação da produção científica gerada pela Universidade.

Picos-PI 19 de Fevereiro de 2016.

Melksedek Amorim Santos Góis
Assinatura

Melksedek Amorim Santos Góis
Assinatura