

MEC-SETEC
INSTITUTO FEDERAL DE MINAS GERAIS – *Campus* Formiga
Curso de Ciência da Computação

Protótipo de Sistema para Monitoramento em Tempo Real de Produtos e Veículos

João Paulo de Menezes

Orientador: Prof. Dr. Bruno Ferreira

Formiga - MG

2017

João Paulo de Menezes

PROTÓTIPO DE SISTEMA PARA MONITORAMENTO EM TEMPO REAL DE PRODUTOS E VEÍCULOS

Monografia apresentada ao Curso de Ciência da Computação do Instituto Federal de Educação, Ciência e Tecnologia de Minas Gerais - Campus Formiga, como requisito parcial para obtenção do grau de Bacharel em Ciência da Computação.

Área de concentração: Computação.

Instituto Federal de Educação, Ciência e Tecnologia de Minas Gerais - Campus Formiga
Ciência da Computação

Orientador: Prof. Dr. Bruno Ferreira

Formiga - MG

2017

004

Menezes, João Paulo de.

Protótipo de Sistema para Monitoramento em Tempo Real de
Produtos e Veículos / João Paulo de Menezes. -- Formiga : IFMG, 2017.
76p. : il.

Orientador: Prof. Dr. Bruno Ferreira
Trabalho de Conclusão de Curso – Instituto Federal de Educação,
Ciência e Tecnologia de Minas Gerais – *Campus* Formiga.

1. Computação. 2. Sistema. 3. Monitoramento - Tempo Real.
4. Geolocalização. 5. Web Service. Título

CDD 004

João Paulo de Menezes

**PROTÓTIPO DE SISTEMA PARA MONITORAMENTO EM
TEMPO REAL DE PRODUTOS E VEÍCULOS**

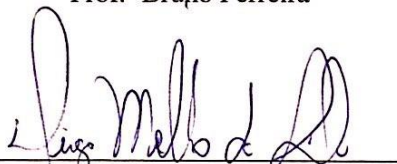
Trabalho de Conclusão de Curso apresentado ao
Instituto Federal de Minas Gerais-Campus
Formiga, como Requisito parcial para obtenção do
título de Bacharel em Ciência da Computação.

Aprovado em: 08 de junho de 20 17.

BANCA EXAMINADORA



Prof.º Bruno Ferreira



Prof.º Diego Mello da Silva



Prof.º Fernando Paim Lima

Resumo

Os sistemas de gerenciamento de entregas auxiliam um dos setores que mais crescem no país, o de compras na internet. Esse tipo de sistema auxiliam ambos os lados de uma transação, a empresa que realiza o trabalho e o destinatário que recebe o produto. Contudo, as vantagens recebidas pelo lado do destinatário podem ser melhoradas aumentando a quantidade de informações que ele recebe. Este trabalho demonstra a modelagem e o desenvolvimento de um protótipo de sistema de gerenciamento de entregas que realiza o rastreamento em tempo-real e mostra ao destinatário a posição geográfica do entregador, aumentando a precisão e quantidade das informações que são apresentados ao cliente.

Palavras-chave: Sistema. Monitoramento. Tempo Real. Pedidos. Geolocalização. Web Service

Lista de abreviaturas e siglas

API (Application Programming Interface)	Interface de Programação de Aplicações
EJB	Enterprise JavaBeans
HTML (HyperText Markup Language)	
HTTP (Hypertext Transfer Protocol)	Protocolo de Transferência de Hipertexto
IFMG	Instituto Federal de Educação, Ciência e Tecnologia de Minas Gerais
IDE (Integrated Development Environment)	Ambiente de desenvolvimento integrado
JEE	Java Enterprise Edition
JSF	JavaServer Faces
JSP	JavaServer Pages
REST (Representational State Transfer)	Transferência de Estado Representacional
SOAP (Simple Object Access Protocol)	Protocolo Simples de Acesso a Objetos
TCC	Trabalho de Conclusão de Curso
UDDI	Universal Description, Discovery and Integration
XML	eXtensible Markup Language
WSDL	Web Services Description Language

Sumário

1	INTRODUÇÃO	13
2	JUSTIFICATIVA	15
3	OBJETIVOS	19
3.1	Objetivo Geral	19
3.2	Objetivos específicos	19
4	REFERENCIAL TEÓRICO	21
4.1	Java	21
4.1.1	Java Enterprise Edition	22
4.2	JavaServer Faces (JSF)	22
4.3	PrimeFaces	23
4.4	Desenvolvimento Android	24
4.4.1	Arquitetura Android	25
4.4.2	Componentes de Aplicações	26
4.5	Web Services	28
4.5.1	SOAP e REST	29
4.5.2	WSDL	30
4.5.3	SOAP	33
5	METODOLOGIA	37
5.1	Processo de Desenvolvimento	37
5.2	Materiais Utilizados	38
6	DESENVOLVIMENTO	41
6.1	Arquitetura do Sistema	41
6.2	Diagrama de Caso de Uso	42
6.3	Banco de Dados	46
6.4	Desenvolvimento Web Service	47
6.4.1	Definição das Classes e Explicação dos Métodos	48
6.4.2	Criação WSDL e Acesso ao Serviço	51
6.5	Desenvolvimento Aplicativo Android	52
6.5.1	Apresentação da Interface do Aplicativo Android	53
6.5.2	Definição das Classes e Explicação dos Métodos	56
6.5.2.1	Comunicação Com o Web Service	56

6.5.2.2	Utilização do SQLite	59
6.5.2.3	Obtenção da Geolocalização	60
6.6	Desenvolvimento Site Responsivo	61
6.6.1	Apresentação da Interface do Site	62
6.6.2	Apresentação do Sistema Pela Visão do Administrador	63
6.7	Testes	67
7	CONSIDERAÇÕES FINAIS	71
7.1	Trabalhos Futuros	71
	REFERÊNCIAS	73

1 Introdução

Das mais diferentes áreas, cada tecnologia ganha seu espaço quando aplica uma nova funcionalidade, ou quando apresenta uma nova forma de executar seu propósito. Em meio às diversas tecnologias evolucionárias e disruptivas que surgem, as aplicações que conseguem sempre evoluir e adicionar novas funcionalidades continuam sendo utilizadas. As que não fazem isso, podem ser substituídas por alguma outra que apresenta novidades, como cita Rohers (2017) “A inovação é o motor da Administração moderna somente ela é que atrai de forma intensa a atenção do cliente, de stakeholders, a empresa que não inova está sujeita ao fracasso”.

A facilidade que os usuários têm de interagir com aplicações de todos os tipos, principalmente pela popularização de *smartphones* e outras tecnologias *smarts*, aumentam a quantidade de usuários de sistemas web. Além disso, grande parte deles acessam a internet apenas por esse tipo de aparelho, como mostra Faria et al. (2014). Na América Latina no ano de 2013 cerca de 30 milhões de usuários acessaram a exclusivamente por *smartphones* e esse número sobe para 190 milhões no restante do mundo. Outra pesquisa realizada pela (COMSCORE, 2017) em 2016, demonstra que 56.1% dos entrevistados acessaram a internet, destes 90% se conectam a internet com *smartphones*.

O aumento dos usuários implica no aumento dos setores que envolvem a web. Um dos tipos que mais cresce nos últimos anos é o de comércio eletrônico (E-bit, 2016). O setor cresceu 15,3% em 2015 e 7,4% em 2016. A qualidade dos serviços prestados pelos representantes desse ramos de trabalho impacta diretamente nesse crescimento. Ela é definida por elementos como descontos, tempo de resposta do vendedor e o prazo de entrega do produto. Este último tem crescido como mostra (EBIT, 2017). O indicador de satisfação dos clientes quanto ao prazo de entrega em 2015 teve o valor de 65%, o melhor índice desde 2013 e encerrou o ano de 2016 mantendo o mesmo valor.

Um tipo de sistema eletrônico que auxilia o aumento desses índices é o sistema de gerenciamento de entregas. (TECNOVIA, 2017). Esses sistemas gerenciam a parte de entregas de uma empresa e tem como principal objetivo aumentar a performance de seus usuários além de oferecer benefícios como redução dos custos de transporte. controle maior sobre as etapas da entrega e uma maior rapidez no processo de entrega. Outra função importante é o rastreamento em tempo real dos entregadores, que consiste em rastrear a localização do entregador através de tecnologia GPS e mostrar sua posição com alta precisão. Essa função impacta diretamente na qualidade do serviço permitindo estabelecer metas referentes ao desempenho do transporte e maior agilidade na resolução de problemas na entrega.

Porém, essa função geralmente é oferecida com precisão apenas aos administradores e gestores do sistema. O que é apresentado aos demais usuários e principalmente aos clientes é uma função bem menos precisa que demonstra apenas trechos do trajeto do entregador. Sistemas como o utilizado pelos *Correios*¹, pela *Jadlog*², pela *Total Express*³ ou pela *Transportadora Americana*⁴ seguem essa lógica e demonstram aos clientes uma função básica de rastreamento. A função aponta apenas a data e hora de quando o entregador chega nas unidades de distribuição de cada cidade.

Em empresas que trabalham com o *delivery* de comida, ramo de negócio que segundo a Associação Brasileira de Bares e Restaurantes (Abrasel), teve em 2012 faturamento de 8 bilhões de reais, o sistema de entregas muitas vezes só auxiliam o gerente a controlar os pedidos, logo os clientes têm menos funções ainda nesses casos. Sistemas como o *Delivoro*⁵, o *Consumer*⁶ e o *Vitto*⁷ tem todas suas funcionalidades voltadas para o gerenciamento e a logística do *delivery* do pedido. Já aplicações como o *iFood*⁸ e o *PedidosJá*⁹ oferecem muitas funções para os usuários destinatários, porém são funções para encontrar e realizar o pedido e não para rastrear e monitorar. Nas empresas de pequeno e médio porte, as funções mais robustas desses sistemas podem não ser muito viáveis, visto que geralmente possuem um custo monetário maior e o objetivo dessas empresas é realizar a entrega com agilidade e rapidez e não demandam muita precisão.

Nos dois casos citados, uma função que apresente ao cliente a opção de rastrear em tempo real o entregador e conhecer com mais precisão o momento em que ele chegará no local de entrega pode aumentar a satisfação do mesmo. Além disso ela poderá poupar gastos da empresa no quesito de retrabalho, visto que pode reduzir os casos onde um entregador vai ao local de entrega e o cliente não esteja presente para receber o produto.

A proposta do trabalho é desenvolver um protótipo de sistema computacional em Java com Primefaces, que auxilie uma rede de entregas, tanto no lado do remetente quanto do lado do destinatário. A principal função deste sistema será realizar o rastreamento em tempo real dos entregadores através de GPS fazendo isso com um baixo custo de aplicação. Os entregadores utilizarão uma aplicação num dispositivo Android para enviar dados para o sistema principal, que por sua vez, ficará responsável por cadastrar novos funcionários, pedidos e usuários. O destinatário poderá acessar o sistema através de um site responsivo para conferir seus pedidos e acompanhar a entrega dos mesmos.

¹ <http://www2.correios.com.br/sistemas/rastreamento/>

² <http://www.jadlog.com.br/tracking.jsp>

³ <http://tracking.totalexpress.com.br/tracking/0>

⁴ <https://www.tanet.com.br/rastrear-encomenda/>

⁵ <http://www.delivoro.com.br/>

⁶ <http://www.programaconsumer.com.br/?v=7ppi3ocf6b>

⁷ <http://sistemavitto.com.br/>

⁸ <https://restaurante.ifood.delivery/>

⁹ <https://www.pedidosja.com.br/>

2 Justificativa

Existem diversos sites e aplicações que realizam pedidos de compra e venda na web. Isso inclui as mais diversas áreas, desde a venda de alimentos até carros, mas em todos os casos, o rastreamento da produção e da entrega até o contratante é precária. O rastreamento do pedido geralmente diz apenas se o pedido está em produção, distribuição ou entrega.

A Figura 1 é um exemplo de como o rastreamento de produtos é feito atualmente no sistema utilizado pelos Correios. O usuário apenas tem acesso a data e hora de quando o entregador alcança algum ponto determinado pelo sistema nesse quando ele alcança a unidade dos Correios e na maioria dos serviços oferecidos por eles, esse pontos só são atualizados em um certo período do dia.

Figura 1 – Imagem do Sistema de Rastreamento Utilizado Pelos Correios

Informação de Rastreamento				Fechar Janela
Número Rastreador:	505660352BR			
Transporte:				
Status:	Objeto entregue ao destinatário			
Entregue em:	21/02/2017 20:23			
Localização	Data	Hora local	Descrição	
FORMIGA - Formiga/MG	21/02/2017	20:23	Objeto entregue ao destinatário	
FORMIGA - Formiga/MG	21/02/2017	10:21	Objeto saiu para entrega ao destinatário	
BELO HORIZONTE - BELO HORIZONTE/MG	20/02/2017	04:55	Objeto encaminhado para CDD FORMIGA	
AC SHOPPING DEL REY - Belo Horizonte/MG	17/02/2017	21:56	Objeto encaminhado para CTE BELO HORIZONTE	
AC SHOPPING DEL REY - Belo Horizonte/MG	17/02/2017	21:50	Objeto postado após o horário limite da agência	

Fonte: Retirada do Site dos Correios - <http://www2.correios.com.br/sistemas/rastreamento/>.

A Figura 2 é um exemplo de como o rastreamento é feito pela Transportadora Americana. O sistema é bem semelhante ao dos Correios mostrado anteriormente, exibe informações sobre a atual situação do pedido, demonstrando onde ele está dentre os centros de distribuição da empresa em algumas cidades. Além de outras informações como chaves de Conhecimento de Transporte Eletrônico(CTe) e de Nota Fiscal Eletrônico(NFe).

Figura 2 – Imagem do Sistema de Rastreamento Utilizado Pela Empresa Transportadora Americana



		Informações
Pedido:	536900558	Nota Fiscal: 006169797
Chave acesso NFe:	35161172381189000625550010061697971943802893	
Conhecimento:	TA-002-001-465883	Qtd. Volume: 2
Chave acesso CTe:	35161143244631000320570010004658831071541981	
Data de saída:	22/11/2016	Previsão de entrega: 29/11/2016
Observação Previsão Entrega:	Considerado mais 1 dia devido a coleta ter sido finalizada após o Horário Máximo de Coleta (HMC).	
Situação:	<ul style="list-style-type: none"> • Mercadoria em trânsito para entrega, com saída da TA-DIVINOPOLIS em 29/11/16 as 08:00 hrs. • Mercadoria transferida da TA-CAMPINAS para TA-DIVINOPOLIS em 24/11/16 as 10:19 hrs. • Mercadoria coletada na DELL COMPUTADORES DO BRASIL LTDA em 23/11/16 as 07:00 hrs. • Conhecimento TA-002-001-465883 emitido em 22/11/16 as 20:53 hrs. • Coleta solicitada por DELL COMPUTADORES DO BRASIL LTDA em 22/11/16 as 16:31 hrs. 	

Fonte: Retirada do Site da Transportadora Americana - <https://www.tanet.com.br/rastrear-encomenda/>

A Figura 3 demonstra como é o rastreamento feito pela Total Express, uma empresa do Grupo Abril. O sistema novamente demonstra dados para consultar uma Nota Fiscal Eletrônica mas novamente demonstra ao usuário o mesmo tipo de informação das anteriores e o mesmo ocorre com a Jadlog, outra empresa citada na seção anterior.

Figura 3 – Imagem do Sistema de Rastreamento Utilizado Pela Empresa Total Express



Data	Hora	Status
25/07/2015	10:22:42	RECEBIDO NO CENTRO DE DISTRIBUIÇÃO
25/07/2015	20:33:36	RECEBIDO NO CENTRO DE DISTRIBUIÇÃO
27/07/2015	19:58:31	TRANSFERENCIA PARA /
28/07/2015	14:23:55	RECEBIDO NO CENTRO DE DISTRIBUIÇÃO Divinópolis/MG
28/07/2015	14:23:55	SEPARADO PARA O ROTEIRO DE ENTREGA
30/07/2015	17:50:07	PROCESSO DE ENTREGA
30/07/2015	18:00:00	ENTREGA REALIZADA

[Ver Protocolo](#)

Fonte: Retirada do Site da Total Express - <http://tracking.totalexpress.com.br/tracking/>

O protótipo de sistema construído como fruto deste trabalho de conclusão de curso adicionaria uma outra informação nesse tipo de sistemas, que seria a geolocalização do entregador durante o percurso de transporte do pedido, Essa informação seria enviada

em tempo real e estaria disponível tanto para o contratante do serviço para empresa que administra a entrega.

Ele também seria capaz de gerenciar pedidos e entregas e traria vantagens para ambos os utilizadores do sistema. O contratante do serviço ganharia tempo pois ele poderia acompanhar seu pedido por todo o trajeto e saberia o estado da sua entrega tudo em tempo real, além das outras vantagens oferecidas por aplicativos de pedido, como realizar os pedidos sem a necessidade de enfrentar filas ou esperar ligações. A empresa que disponibilizar o serviço também teria vantagens, pois ao monitorar o entregador ela pode calcular melhores forma de entrega, otimizar tempo de entrega e ter menos custos com retrabalhos.

Por fim, o sistema sendo modelado e desenvolvido de forma genérica, teria um *back-end* que não é amarrado a nenhum tipo específico de negócio, onde se muda apenas a *front-end* ajustando-a para o fim desejado, assim pode-se ofertá-lo para qualquer área cujo cenário envolva os elementos citados, entregadores e destinatários.

O sistema poderia funcionar também adicionando ele a serviços já existentes, dessa forma poderia importar dados e funções dos sistemas já utilizados pela empresa e assim adicionaria funções novas para os clientes e destinatários desse tipo de serviço.

Todas essas vantagens somadas ao conhecimento adquirido durante o desenvolvimento do sistema motivam a construção do mesmo.

3 Objetivos

3.1 Objetivo Geral

A modelagem e desenvolvimento de um Web site responsivo feito em Java com PrimeFaces que tem a principal função de receber dados geográficos em tempo real de uma aplicação móvel através de Web service. Além disso, o sistema contará com ferramentas para gerenciar os pedidos, entregadores e rastreamento em tempo real desses entregadores.

Esse protótipo de sistema tem como objetivo ser genérico a fim de ser utilizado, ou aplicado em sistemas já existentes, quando o cenário de execução envolva uma empresa com entregadores e seus destinatários.

3.2 Objetivos específicos

- Pesquisar e Estudar uma forma de Enviar dados de uma plataforma mobile para o sistema utilizando Web services. Pesquisar entre as APIs já existentes, verificando a viabilidade das mesmas. Pesquisar também formas de garantir que o envio de dados de seja feito de forma segura e que eles sejam salvos de forma confiável.
- Desenvolver um sistema Web JSF utilizando Primefaces, que deve monitorar um determinado dispositivo móvel, para isso é necessário que ele se comunique com esse dispositivo recebendo sua localização geográfica em tempo real. O sistema web também apresenta diversas questões administrativas como cadastro de produtos, clientes e relatórios gerenciais sobre as entregas.
- Desenvolver um site responsivo que possa ser utilizado num ambiente móvel e que possa executar as operações existentes no sistema como o cadastro de um novo cliente e a visualização do estado do pedido.

4 Referencial Teórico

4.1 Java

Java é uma linguagem de programação com a principal característica de ser uma linguagem orientada a objetos. Desde o seu lançamento essa linguagem tem o foco voltado para no uso em clientes web o que fica bem claro quando se entende a forma como a linguagem funciona (DEITEL; DEITEL, 2003). Com isso ela é uma linguagem portátil e que pode ser executada em diversos dispositivos diferentes. Isso pode ser observado no desenvolvimento deste trabalho, pois basicamente é utilizada em todo o projeto, seja a parte do sistema web ou a parte de dispositivos móveis. A mesma linguagem é utilizada mas interpretada de forma diferente dependendo do dispositivo onde ela se encontra.

Java foi criado pela Sun em 1991, ele foi projetado para esse conceito de ser uma linguagem utilizada em dispositivos eletrônicos embarcados pois na época não havia nenhuma outra linguagem que satisfizesse as necessidades dessa época. Segundo Sebesta (2009), os projetistas buscavam uma linguagem orientada a objetos que fosse mais simples do que as já existentes mas que ainda mantivesse a confiabilidade. Dois aspectos que contribuíram para os dispositivos para qual ela estava sendo planejada.

A portabilidade comentada acima se deve a forma como essa linguagem é compilada. Ela não é feita da forma tradicional onde se transcreve o código fonte em linguagem de máquina. Em Java primeiro o código fonte é traduzido para *bytecode*, que é a linguagem da máquina virtual Java (JVM), e então executado nessa máquina virtual (CAELUM, 2017). A JVM então traduz esses *bytecodes* para a linguagem de máquina nativa. Como dito por Deitel e Deitel (2003), a JVM é a base da plataforma Java. E completa, “uma máquina virtual (VM) é um aplicativo de software que simula um computador, mas oculta o sistema operacional e *hardware* subjacente dos programas que interagem com ela”. Logo, qualquer que seja o dispositivo que contenha uma JVM, esse dispositivo será capaz de interpretar um *bytecode* java compilado em outro lugar.

Outra vantagem que Java possui é a utilização de bibliotecas. Os programas Java são consistem em partes chamadas classes. As coleções de classes são chamadas Bibliotecas. Essas bibliotecas também são conhecidas como Java APIs, elas são formadas por diversas classes fornecidas pela própria linguagem ou por outros desenvolvedores na internet de forma freeware ou shareware.

Todas as características citadas estão presentes no Java Standart Edition (JSE), que pode se dizer ser a versão básica. Além dessa versão duas outras foram introduzidas a Enterprise Edition (JEE) voltada para processamento Web em rede distribuída e a

Micro Edition, (JME), voltada para o desenvolvimento de aplicativos para dispositivos embutidos e com recursos limitados, como smartwatches e decodificadores de TV.

4.1.1 Java Enterprise Edition

Com o crescimento da Web os criadores do Java perceberam que com essas características ele teria um desempenho nessa área. E com isso surgiu o Java Enterprise Edition. Ele fornece uma série de recursos de rede tornando mais fácil desenvolver aplicações baseadas na Web. Uma relação cliente-servidor, que é o caso de estudo deste projeto, é bem simples.

O JEE é um padrão dinâmico para a produção de aplicativos corporativos seguros, escaláveis e altamente disponíveis (BOND et al., 2003). Como Bond cita, quando um servidor JEE fornece serviços o padrão JEE vai definir quais serão os serviços fornecidos. Esses serviços serão fornecidos através de containers JEE e cada um desses contêineres vai ter um grupo definido de serviços para cada componente.

Esses contêineres vão fornecer um ambiente JSE em tempo de execução para os componentes. E esses componentes podem variar entre dois tipos básicos de componentes: os Componentes Web e os Componentes EJB. Os Componentes Web possuem 2 tipos de componentes são eles servlets e JavaServer Pages (JSP), e é responsável por manipular e apresentar os dados para o usuário. Já os componentes EJB são responsáveis por fornecer um desenvolvimento rápido e simplificado (BOND et al., 2003).

O que torna a Web realmente útil é interatividade. Ela se torna possível através das páginas Web dinâmicas. E as JSPs juntamente as Servlets implementam esse dinamismo. Elas permitem aos programadores criar conteúdo dinâmico para a Web através da reutilização de componentes pré definidos e interagir com os componentes da página usando scripts do lado do servidor (DEITEL; DEITEL, 2003).

Os componentes pré definidos do JSP são chamados JavaBeans, que são os componentes EJB. Os desenvolvedores podem criar bibliotecas identificadas encapsulando outras operações. Essas bibliotecas são identificadas por tags, que são bastante comuns para outros desenvolvedores Web, fazendo com que mesmo para aqueles não muito familiarizados com a linguagem Java possam utilizar todas as ferramentas que a linguagem oferece.

4.2 JavaServer Faces (JSF)

Utilizando somente JSP e as Servlets o desenvolvimento deste trabalho conseguiria ser desenvolvido. Porém o que foi utilizado é o JavaServer Faces (JSF). O JSF é um *framework* assim como o JSP, porém não apenas para a camada de visão, mas sim para toda de apresentação. Além a partir do JEE 6, o JSF se utiliza de Facelets para substituir o

JSP, visto que ele não suportava todas as novas características do JSF. Na documentação oficial do JEE6 (ORACLE, 2017), o JSP está sendo marcado como *Deprecated*, que é uma tag Java que os desenvolvedores utilizam para desencorajar o uso de determinada tecnologia. Logo, é mais vantajoso utilizar o JSF com Facelets do que o JSP.

O JSF simplifica o desenvolvimento através de uma abordagem centrada no componente de desenvolvimento de interfaces de usuários Java Web. Ele também garante que as aplicações são bem desenhadas com maior capacidade de manutenção, integrando o padrão de design Model-View-Controller (MVC) em sua arquitetura. Finalmente, uma vez que o JSF é um padrão do Java, desenvolvido através de Java Community Process (JCP), os provedores das ferramentas de desenvolvimento estão plenamente capacitados para fornecer facilidade de uso, visual e desenvolvimento produtivo de ambientes para JavaServer Faces (ORACLE, 2016).

JSF opera provendo uma Servlet e um modelo de componentes que inclui o gerenciamento de eventos, validação do lado do servidor, conversão de dados e renderização de componentes. Esses componentes por sua vez podem ser utilizados via taglibs que podem ser utilizadas na camada de visão (JSP ou Facelets)

4.3 PrimeFaces

Como o JSF é um *framework* baseado em componentes, ele permite que componentes de interface do usuário sejam construídos de forma personalizada. Dessa forma cada um deles terá uma boa aparência e pode ser reutilizado em qualquer outro projeto. Existem diversos *frameworks* de código abertos disponíveis, e entre eles o PrimeFaces é uma excelente escolha.

O PrimeFaces é uma biblioteca de componentes do JSF que é excelente em termos de recursos e facilidade de uso. Eles possui suporte a mais de 100 componentes de interface de usuário e foi construído com suporte a AJAX (REDDY, 2013). Ele é uma biblioteca muito leve e foi pensado para não possuir um impacto muito grande no desenvolvimento das aplicações. Além das vantagens citadas o PrimeFaces também tem componentes adicionais para desenvolvimento móvel que é um requisito para o desenvolvimento deste projeto.

Ele é a biblioteca mais utilizada para desenvolvimento de interfaces para sistema Java segundo o DevRates.com, com uma nota geral de 9,4 (PRIMEFACES, 2017). Até bibliotecas concorrentes como o IceFaces são inspiradas no Primefaces. O gráfico 4 a seguir foi gerado utilizando o Google Trends e ele mostra a popularidade entre Primefaces e seus dois maiores concorrentes o IceFaces e o RichFaces.

Entre os produtos do primefaces estão *templates* e temas. Eles são produtos pagos,

e são pacotes já prontos de layouts que possuem um visual incrível e que valorizam bastante o produto onde são utilizados. Todos os produtos possuem documentação e support que podem ser acessados através do site.

Figura 4 – Gráfico que demonstra a Popularidade através de Pesquisas com o termo PrimeFaces



Fonte: Site Google Trends

4.4 Desenvolvimento Android

Android é um sistema operacional de código aberto para dispositivos móveis, como smartphones, tablets e mais recentemente em smartwatches, possui uma interface de usuário baseada na manipulação direta, voltado principalmente para esse tipo de dispositivos, com tela sensível ao toque. Baseado no Linux, ele foi desenvolvido pela Open Handset Alliance, um conjunto de várias empresas lideradas pelo Google e divulgado em 5 de novembro de 2007 (SILVA, 2010).

A primeira versão do Kit de Desenvolvimento de Software para Android (SDK) foi lançada pelo Google em 2007. Já primeira versão comercial, o Android 1.0 uma versão alpha do sistema, foi lançada em 2008. Após a atualização do Android 1.5, chamada de Cupcake, todas as versões do android recebem um nome baseado em doces e todas

seguindo as letras do alfabeto. A última versão lançada foi o Android 7.0 Nougat, lançado em 22 de agosto de 2016, como mostra o própria site de desenvolvedores da plataforma Android (ANDROID, 2017). Atualmente, como mostra um artigo da Forbes (MATHEWS, 2017) existem mais de 1.4 bilhões de dispositivos Android pelo mundo.

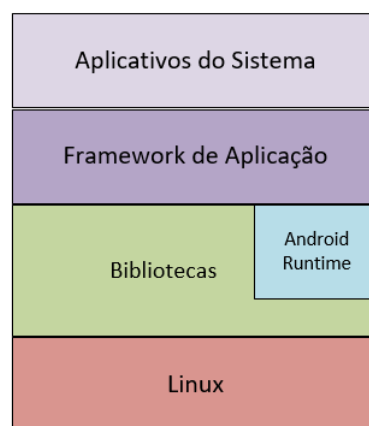
O Android SDK disponibiliza um conjunto de APIs básicas para o desenvolvimento de aplicações, utilizando a linguagem Java (SILVA, 2010). As aplicações desenvolvidas podem ser distribuídas e até vendidas através de lojas como a Google Play, Amazon App Store, Opera Mobile Store e a ApkMirror. A quantidade de aplicativos na Google Play segundo o site statista (STATISTA, 2017) em Março de 2017, era de 2.8 milhões.

A plataforma Android oferece diversos recursos como: SQLite, um sistema gerenciador de banco de dados nativo do sistema; Dalvik virtual machine (versões anteriores ao Android 4.4), uma JVM voltada para dispositivos móveis; Browser Integrado; Suporte multimídia e tecnologias de comunicação sem fio como bluetooth, 3G e WiFi (SILVA, 2010).

4.4.1 Arquitetura Android

A arquitetura da plataforma Android tem sua base no Kernel (núcleo) do Linux onde estão as funções mais básicas do sistema. Podemos dividir o resto em outras 3 camadas, a de bibliotecas, *Framework* de Aplicação e a camada dos Aplicativos do sistema (ANDROID, 2017). A Figura 5 representa a arquitetura brevemente descrita acima.

Figura 5 – Representação Da Arquitetura do Sistema Android Por Camadas



Fonte: Elaborada pelo autor.

A base da arquitetura o kernel do Linux, é uma base já consolidada e conhecida, visto que a primeira versão é de 1991 (TORVALDS, 1991) e a versão 1.0 é de 1994, permite que os fabricantes dos dispositivos que utilizam a plataforma Android, desenvolvam os

drivers de *hardware* para um kernel de confiança. Nessa base estão os drivers de *hardware* essenciais como câmera, teclado, bluetooth, WiFi entre outros (ANDROID, 2017).

A segunda camada é a de bibliotecas, é onde estão as bibliotecas C/C++ nativas que são utilizadas até mesmo por componentes e serviços básicos (ANDROID, 2017). Bibliotecas como o WebKit e o banco de dados SQLite estão presentes nessa camada. Algumas funcionalidades dessas bibliotecas podem ser acessadas graças a camada superior a essa.

A terceira seção da arquitetura é localizada na segunda camada, é denominada Android Runtime (ART). É a seção onde está localizada a máquina virtual que permite que cada aplicação Android execute em seu próprio processo. Os dispositivos com versões anteriores ao 4.4KitKat, incluindo essa versão, utilizavam a Dalvik Virtual Machine (Dalvik VM), como mostra o autor de tutorials point (TUTORIALSPPOINT, 2017), cada aplicativo contém sua própria instância da Dalvik VM . A partir da versão 5.0Lollipop, cada aplicativo contém uma instância própria do ART. Além dessa função a plataforma Android também oferece um conjunto de bibliotecas de tempo de execução maioria da funcionalidade da linguagem de programação Java, como consta em Android (ANDROID, 2017).

A camada do *Framework* de Aplicação é onde estão o conjunto completo de recursos da plataforma Android através das APIs de Java. É nessa camada que são fornecidos muitos serviços de alto nível para a camada de Aplicações na forma de classes Java (tutorialspoint, 2014). Essa camada é a que permite acessar as bibliotecas presentes na segunda camada. o site do Android (ANDROID, 2017) cita um exemplo, "... é possível acessar OpenGL ES pela Java OpenGL API da estrutura do Android para adicionar a capacidade de desenhar e manipular gráficos 2D e 3D no seu aplicativo."

E por fim a camada de Aplicativos do Sistema, a cama do topo, onde todas as aplicações irão rodar, inclusive a desenvolvida para este trabalho. As aplicações ou Aplicativos são programados em linguagem de programação Java. As ferramentas do Android SDK compilam o código em um pacote Android (APK). Os arquivos APK contêm todo o conteúdo de um aplicativo do Android e são os arquivos de instalação para cada aplicação.

4.4.2 Componentes de Aplicações

As aplicações desenvolvidas para Android são formadas por alguns componentes que são os elementos principais e fundamentais delas. Eles são os "blocos de construção" como dito pelo próprio site do Android (ANDROID, 2017). Esses componentes são fracamente acoplados através do *AndroidManifest.xml*, um arquivo XML que descreve os componentes do aplicativo e como cada componente interage com os outros (tutorialspoint,2014).

Existem alguns componentes que são os básicos para todos os aplicativos. São 4 e cada tipo tem uma finalidade distinta, eles são as Atividades, os Serviços, os Receptores de Transmissão e os Provedores de Conteúdo.

“As Atividades representam uma única tela com uma interface de usuário” (tutorialspoint,2014). Um aplicativo pode ter mais de uma Atividade, por exemplo um aplicativo pode ter uma atividade que mostra uma lista página com uma galeria de fotos, uma atividade de visualização de fotos e uma outra para envio de fotos. Nos aplicativos onde isso ocorre, uma das atividades deve ser marcada como a principal, ou seja a que será iniciada quando o aplicativo for lançado. Independente de qual seja a atividade principal, uma atividade pode iniciar outra atividade para realizar outra ação. Uma atividade é implementada como uma subclasse de *Activity*.

“Serviços são componentes executados em segundo plano para realizar operações de execução longa ou para realizar trabalho para processos remotos” (ANDROID, 2017). Citando o mesmo exemplo anterior, um aplicativo com uma Atividade para enviar fotos, pode utilizar um Serviço para enviar as fotos enquanto o usuário pode continuar acessando a galeria. Os serviços não possuem uma interface gráfica, sendo assim, um outro componente pode iniciar o serviço por ele. Um serviço é implementado como uma subclasse de *Service*.

“Os Receptores de Transmissão simplesmente respondem a mensagens de Aplicações ou do Sistema.” (tutorialspoint,2014). As mensagens podem vir tanto do sistema, como por exemplo mensagens de que o nível da bateria está baixo, ou de outros Aplicativos, no exemplo citado anteriormente a aplicação pode comunicar a outra aplicação que a foto enviada foi recebido com sucesso. Assim como os Serviços, os Receptores de Transmissão não exibem nenhuma interface para o usuário. Os receptores são implementados como subclasses de *BroadcastReceiver*.

“Provedores de conteúdo gerenciam um conjunto compartilhado de dados do aplicativo” (ANDROID, 2017). O Armazenamento de dados, num sistema de arquivos ou em um banco de dados SQLite, é possível graças a esses Provedores. Eles também permitem que outros aplicativos possam consultar e modificar esses dados. Eles são implementados como uma subclasse de *ContentProvider*.

Os componentes básicos podem usar outros componentes adicionais para sua composição. Alguns desses componentes são: Fragmentos (*Fragments*) que representam uma parte da *interface* do usuário; Views (*Views*) elementos da interface como botões, listas, etc; Intent (*Intent*) que são mensagens que unem diferentes componentes juntos.x’

4.5 Web Services

Os Web Services são serviços que são ofertados através da Web. Serviços assim como componentes, são objetos independentes que ajudam a construir uma aplicação maior. Eles possuem características únicas que os diferem dos componentes tradicionais, como a completa autonomia em relação a outros serviços. Com isso cada serviço tem seu próprio domínio e é projetado para resolver uma função de negócio específica (WEBMOBILE, 2017a).

O serviço mais bem sucedido é o Web Service. Ele se comunica com os sistemas através de protocolos internet, na maioria das vezes o HTTP. Essa comunicação é feita via dados formatados que podem ser de diferentes tipos, um dos mais utilizados é documentos na forma XML.

Um dos tipos de mensagem no formato XML é o SOAP. O Simple Object Access Protocol (SOAP) foi aceito no ano de 2000 pela W3C, ele estabeleceu uma estrutura de transmissão para realizar a comunicação entre serviços e aplicações via HTTP (WEBMOBILE, 2017a).

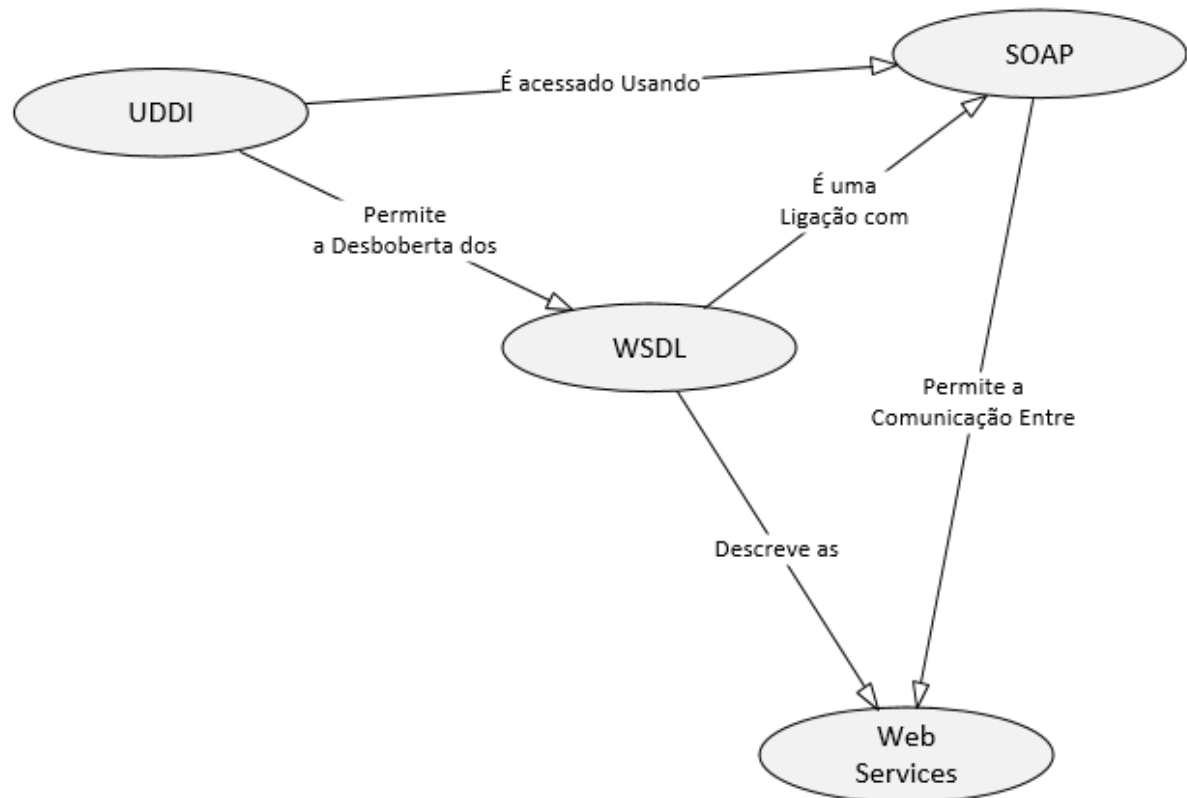
No ano seguinte o W3C publicou a especificação WSDL. Ele é uma nova implementação do XML e fornece uma linguagem para descrever a interface dos web services, que ainda mantém uma estrutura parecida com o XML. O WSDL foi completado com a especificação UDDI (*Universal Description, Discovery and Integration*) que fornece meios para a descoberta dinâmica de descrições de serviço. A Figura 6 demonstra uma representação em alto nível dos relacionamentos entre os padrões citados até o momento.

A aceitação e popularização dos Web Services resultaram no surgimento de tecnologias suplementares que se tornaram um padrão de fato. Quando desenvolve-se Web Services deve-se prezar tecnologias que forneça uma descrição de serviços, como o WSDL citado anteriormente, além de ser capaz de transportar documentos XML utilizando SOAP sobre HTTP. Normalmente um Web Service é capaz de agir como solicitante e provedor de um serviço.

No caso de Provedor de serviços, um web service oferta uma interface pública onde qualquer solicitante público pode chamar. Essa interface é disponibilizada como já dito antes através de uma descrição do serviço (WEBMOBILE, 2017a).

Como pode-se notar a descrição de serviços, como a ofertada pela WSDL é de extrema importância. “A camada de integração introduzida pela estrutura de web services estabelece um padrão, universalmente reconhecido e com interface programática suportada. O WSDL permite a comunicação entre essas camadas ao fornecer descrições padronizadas” (WEBMOBILE, 2017a).

Figura 6 – Relacionamento Entre os Padrões de Comunicação de um Web Service



Fonte: Elaborada pelo autor.

4.5.1 SOAP e REST

A comunicação entre os Web Services e as aplicações cliente ocorre devido à padronização das formas de troca de mensagens com os serviços. Dois padrões amplamente utilizados são o SOAP (Simple Object Access Protocol) e o REST (Representational State Transfer). Cada um apresenta suas vantagens e o melhor local para ser utilizado. Uma comparação entre os dois é apresentada por Henrique (2017) e transcrita abaixo:

- O SOAP é uma especificação de um protocolo estruturado em XML, logo as mensagens trocadas por esse padrão são maiores. Já o REST pode se comunicar através de diversos formatos, sendo JSON o mais usado, que comparado ao XML tem muito menos caracteres numa mensagem básica.
- A complexidade na comunicação com um Web Service SOAP é maior visto que são necessárias bibliotecas específicas para seu funcionamento. Já o REST é mais simples de ser utilizado e foi projetado para ser usado em clientes “magros”.
- O SOAP foi construído com o pensamento da comunicação entre servidores, as bibliotecas necessárias para seu funcionamento tem utilidade, elas trazem consigo uma carga adicional principalmente na área de segurança.

Com as características acima pode-se perceber que cada padrão oferece uma vantagem para uma determinada situação. Como mostra Rozlog (2017), "REST é fácil de entender e extremamente acessível porém faltam padrões, e a tecnologia é considerada apenas uma abordagem arquitetural. Em comparação, o SOAP é um padrão da indústria, com protocolos bem definidos e um conjunto de regras bem estabelecidas".

Sabendo dessas informações, o padrão escolhido para utilização neste trabalho foi o SOAP visto que ele pode atender melhor aos requisitos do trabalho. O SOAP trabalha com processamento e chamada assíncronos para garantir a segurança e confiabilidade dos dados, além de utilizar contratos formais para comunicação entre os dois lados e operações *stateful* caso o aplicativo precise de informação contextual e gerenciamento de estado com coordenação e segurança.

4.5.2 WSDL

Para melhor entender como um web service é expresso por um documento WSDL, iremos demonstrar as estruturas e os construtores que representam essa definição. O primeiro elemento é o elemento *definitions*, ele age como o container para a definição do serviço. Como pode ser observado no exemplo 1. Os elementos *interface* são elementos abstratos e os elementos *service* são concretos.

```
<definitions>
  <interface name="Time">
    ...
  </interface>
  <message name="InformacoesJogadores">
    ...
  </message>
  <service>
    ...
  </service>
  <binding name="Binding1">
    ...
  </binding>
</definitions>
```

Os elementos de interface abstrata são os elementos *interface* no WSDL. Estes construtores contêm um grupo de operações relacionadas entre si. Fazendo uma analogia a uma arquitetura utilizada no desenvolvimento de sistemas, a arquitetura baseada em componentes (SPAGNOLI, 2003), os elementos *interface* são análogos. Como descrito por

(SPAGNOLI, 2003), "... a *interface* indica como os componentes podem ser reusados e conectados a outros componentes e devem ocultar os detalhes que não são necessários para o reuso".

Compondo a interface temos os componentes *operation*, que é o equivalente a métodos de um componente. Eles vão representar uma única ação ou função desse método. O exemplo 2 apresenta um exemplo de *interface* junto com uma *operation*.

```
<definitions>
  <interface name="Time">
    <operation name="GetJogador">
      ...
    </operation>
  </interface>
</definitions>
```

"Um elemento *operation* típico consiste de um grupo de mensagens de entrada e saída correlatas. A execução de uma *operation* requer a transmissão ou intercâmbio destas mensagens entre o serviço solicitante e o provedor de serviço" (WEBMOBILE, 2017a).

Essas mensagens são representados pelos construtores *message* e são declarados não sob os elementos *operation* mas sim sobre os elementos *definitions*. As relações entre os *operation* e as mensagens serão feitos através de elementos filhos dos *operation* denominados *input* ou *output*. Como pode-se observar no exemplo 3, a mensagem de "Registro de Jogador" está declarada na definição e associada através de um elemento *input*. "Um elemento *message* pode conter um ou mais parâmetros *input* ou *output* que pertencem a uma *operation*."

```
<definitions>
  <message name="RegistroJogador">
    ...
  </message>
  <interface name="Time">
    <operation name="GetJogador">
      <input name="Msg1" message="RegistroJogador"/>
    </operation>
  </interface>
</definitions>
```

Os elementos *part* vão definir os parâmetros para uma mensagem, como já dito, eles podem ser de entrada (*input*) ou de saída (*output*). Ele irá prover as informações

necessárias para os parâmetros, como o tipo do dado associado e o identidade que ele assume. O exemplo 4 demonstra parâmetros criados para completar a mensagem do exemplo anterior.

```
<definitions>
  <message name="RegistroJogador">
    <part name="nome" type="xs:string">
      Tim Duncan
    </part>
    <part name="posicao" type="xs:string">
      Pivô
    </part>
    <part name="numero" type="xs:integer">
      21
    </part>
  </message>
</definitions>
```

Por fim os elementos concretos os *services*. Esses são os elementos que irão estabelecer as conexões entre protocolos. Eles serão compostos por um ou mais *endpoint*, que representam os locais onde o Web Service pode ser acessado. Os *endpoints* consistem de informação de localização e protocolos para acesso ao Web Service e são armazenados numa coleção de elementos *endpoint*, como dados de endereço físico ou lógico. Além dos endpoints os services podem conter elementos *bindings*, que vão associar informações de formato de protocolo e mensagem. Eles são os responsáveis por definir os requisitos de chamada para as *operations*. Os *bindings* são compostos por *operations* que tem as função semelhante às da interface mas com o contexto a que ele está aplicado. O exemplo 5 demonstra o uso desses elementos concretos.

```
<definitions>
  <service name="Service1">
    <endpoint name="EndPoint1" binding="Binding1">
      ...concrete implementation details...
    </endpoint>

    <binding name="Binding1">
      <operation>
        <input name="Msg1" message="jogador"/>
      </operation>
    </binding>
  </service>
</definitions>
```



```
    </binding>
  </service>
</definitions>
```

4.5.3 SOAP

O Simple Object Access Protocol (SOAP) é um protocolo utilizado por Web Services baseados em XML e um dos mais conhecidos formatos de mensagens para este mesmo fim. Como já dito, teve seu início por volta dos anos 2000 ele inicialmente foi planejado como uma tecnologia para transpor a brecha entre plataformas baseadas em comunicação de chamada de procedimento remoto (RPC).

A especificação SOAP estabelece um formato padrão de mensagem que consiste em um documento XML capaz de hospedar dados RPC e centrados em documentos (WEBMOBILE, 2017b) , principalmente com um documento que descreva uma descrição dos serviços como é o caso do WSDL.

O SOAP estabelece um método de comunicação baseado em um modelo de processamento semelhante ao funcionamento da arquitetura de um Web Service. A diferença vem quando o SOAP introduz termos e conceitos relacionados ao modo como as mensagens são manipuladas. Ele se utiliza de um mecanismo chamado nó SOAP.

Um nó SOAP representa o processamento lógico responsável pela transmissão, recebimento e realização de uma série de tarefas sobre mensagens SOAP. Uma implementação de um nó SOAP é tipicamente específica à plataforma e é normalmente rotulada como um SOAP server (servidor) ou SOAP listener (ouvinte). Existem também variações especializadas, tais como SOAP routers (roteadores)(WEBMOBILE, 2017b).

Os nós podem existir como 3 tipos diferentes, os emissores iniciais, intermediários e receptores finais. Um mesmo nó pode assumir comportamento dos 3 tipos no decorrer do seu tráfego e do estado da atividade atual. Um nó intermediário é exatamente o nó que está entre as extremidade dos emissores iniciais e os receptores finais.

Já o recipiente de uma mensagem SOAP é chamado de envelope SOAP. A estrutura lembra um pouco a demonstrada anteriormente pois é no formato XML também. O elemento principal, a raiz é o *Envelope*, que delimita o documento da mensagem. Ele é composto por uma seção *Body* e uma área *Header* que é opcional. O exemplo 6 demonstra o elemento raiz *Envelope* e as duas seções que ele carrega.

```
<env:Envelope xmlns:env=http://www.w3.org/2003/05/soap-envelope>
  <env:Header>

  </env:Header>
```

```

    <env:Body>

    </env:Body>
</env:Envelope>

```

O construtor *Header* é o que representa o cabeçalho SOAP e ele é mais comumente utilizado em implementações de extensões SOAP, na Identificação de alvos SOAP intermediários e no Fornecimento de meta-informação adicional sobre a mensagem. As informações contidas no *Header* podem ser adicionadas, removidas ou processadas mesmo enquanto a mensagem SOAP está trafegando até o seu destino, isso pode ser feito através de SOAP intermediários.

Já o construtor *Body* que é obrigatório, age como um recipiente para a carga útil que são os dados que são entregues pela mensagem (WEBMOBILE, 2005). Ele também pode hospedar informações de exceção dentro de componentes de falha utilizando o construtor *Fault*. Eles podem coexistir com informações úteis dentro de uma mesma mensagem, mas normalmente são enviados separadamente em mensagens específicas de erro. O construtor *Fault* possui uma série de elementos de sistema que identificam características de exceção. O exemplo 7 demonstra o caso incomum onde um envelope carrega tanto dados úteis quanto dados de falha.

E esses são elementos básicos da estrutura interna e a sintaxe de uma mensagem SOAP. Um último atributo pode ser mencionado que é o que representa que o nó SOAP possui um papel. “Para indicar que um nó SOAP possui um papel, o atributo `env:role` deve ser usado. Esse atributo permite que uma mensagem SOAP identifique blocos de cabeçalho destinados a tipos específicos de receptores SOAP.” (WEBMOBILE, 2005).

```

<env:Envelope xmlns:env=http://www.w3.org/2003/05/soap-envelope>
  <env:Header>

  </env:Header>
  <env:Body>
    <x:Jogador xmlns:x=http://www.ligabasquete.ws/>
      <x:nome>

      </x:nome>
    </x:Jogador>

    <env:Fault>
      <env:Code>

      <env:Value>

```

```
        env:VersionMismatch
      </env:Value>
    </env:Code>
  <env:Reason>
    <env:Text xml:lang=pt-BR>
      Versões não compatíveis
    </env:Text>
  </env:Reason>
</env:Fault>
</env:Body>
</env:Envelope>
```


5 Metodologia

5.1 Processo de Desenvolvimento

O processo utilizado foi o de Desenvolvimento orientado ao Reuso, como mostra Sommerville (2007). Esse processo consiste no reuso de outros software e códigos já existentes que conseguem suprir os requisitos do sistema a ser construído. O principal objetivo é a redução dos custos e riscos durante o desenvolvimento, principalmente a redução do tempo de produção. Ele tem basicamente 4 estágios de desenvolvimento. E são eles: Análise de componentes, Modificação de requisitos, Projeto do sistema com reuso e Desenvolvimento e Integração.

Na Análise de componentes, é feita uma busca por componentes que podem suprir ou ajudar a implementar a especificação do projeto. Para esse projeto foi feita uma pesquisa a fim de buscar bibliotecas e APIs que fossem úteis para a implementação.

Para a aplicação móvel desenvolvida foram encontradas a API do Google Maps, para ajuda da coleta e amostragem da geolocalização, e a biblioteca Ksoap2, responsável pela comunicação da aplicação com o Web Service.

Na Modificação de requisitos, os requisitos são analisados a fim de verificar se serão modificados visto os componentes encontrados no estágio anterior. Nesse projeto nenhuma modificação precisou ser feita nos requisitos devido ao tipo de componentes que foram buscados para utilização. Como o que foi reutilizado foram bibliotecas, elas podem ser perfeitamente adicionadas ao sistema sem modificar os requisitos.

No Projeto do sistema, o *framework*, a abstração que irá unir os códigos, é projetado ou algum outro existente é utilizado. Nesse projeto os projetos feitos serão demonstrados nas próximas seções.

E por fim o Desenvolvimento e a integração, onde o software é desenvolvido ou é utilizado algum já existente e os componentes são integrados a eles. Nesse projeto o sistema quase como um todo será desenvolvido utilizando a linguagem de programação Java. O Servidor Web com a versão Enterprise Edition (JEE), utilizando JSF e Primefaces. A aplicação Android também com o Java na versão Standard Edition (SE), obviamente, cada um no seu respectivo ambiente de desenvolvimento.

5.2 Materiais Utilizados

O sistema foi implementado utilizando diferentes ferramentas de desenvolvimento e diferentes dispositivos de *hardware*. Todos os softwares de desenvolvimento utilizado são gratuitos. A maior parte do sistema foi implementado em um computador portátil, toda a codificação foi realizada nesse dispositivo. Em adição a ele, um *smatphone* foi utilizado para realizar os testes com a aplicação móvel desenvolvida, principalmente pelos testes envolverem as funções de geolocalização desse dispositivo. As especificações técnicas dos dispositivos usados são as seguintes:

- Computador.
- Sistema Operacional : 64bits - Windows 10 Home Single Language
- Processador: Intel Core i7-4500U CPU @ 1.80GHz base x64
- Memória RAM: 8,00 GB
- Smartphone.
- Sistema Operacional : Android 5.0.2
- Processador: Snapdragon 400 1.2 GHz Quad Core
- Memória RAM: 1,00 GB

O Sistema teve a maior parte desenvolvida com a IDE Eclipse¹ com o pacote para trabalhar com o Java EE. Esse pacote é voltado para o desenvolvimento de ferramentas Web principalmente com Java EE, JPA, JSF e outros. O Pacote inclui ferramentas para versionamento através do Git, ferramentas de desenvolvimento JavaScript, editores e ferramentas para trabalho com XML além de ferramentas básicas de desenvolvimento.

A aplicação android foi desenvolvida com a IDE Android Studio², chamada de IDE oficial do Android. Ela contém todas as ferramentas necessárias para desenvolvimento de aplicações para android em qualquer de suas versões. Ela ainda possui um emulador com os mais diversos dispositivos para teste das aplicações que estão sendo desenvolvidas ou dá a opção para o usuário testar num dispositivo real.

O *template* utilizado como base para criação do site foi o AdminLTE³. É um *template* gratuito que se utiliza de algumas ferramentas do Bootstrap⁴. É um pacote com diversos arquivos css e javascript além de interfaces prontas para o auxílio da criação de

¹ <http://www.eclipse.org/downloads/packages/eclipse-ide-java-ee-developers/keplersr2>

² <https://developer.android.com/studio/index.html?hl=pt-br>

³ <https://adminlte.io/>

⁴ <http://getbootstrap.com/>

sites principalmente ferramentas para desenvolvimento responsivo. O site foi construído utilizando Primefaces⁵ que oferece diversos componentes para construção de um site padronizado e com facilidade de utilização. O motivo para a escolha de um *template* que não seja do Primefaces é que eles são distribuídos apenas através de licenças pagas, como todo o sistema está se baseando em ferramentas gratuitas, o AdminLTE foi escolhido para ser utilizado em seu lugar.

⁵ <https://www.primefaces.org/>

6 Desenvolvimento

O principal objetivo deste trabalho é a modelagem e o desenvolvimento do protótipo de sistema descrito nas seções anteriores. O desenvolvimento seguiu a modelagem já demonstrada e pode ser entendida com mais facilidade observando os diagramas de caso de uso demonstrados. O sistema gerado foi chamado de Archon tanto o site quanto a aplicação carregam o nome do sistema. Então será bem comum encontrar esse nome nas próximas seções.

Como já dito, o sistema é dividido em 3 partes básicas o Web Service, o Site Responsivo e a Aplicação Móvel. Cada parte será analisada separadamente relevando os pontos mais importantes e demonstrando partes do sistema.

Antes de iniciar as análises, será explicado o ambiente de desenvolvimento utilizado, os *hardwares* utilizados e as Ferramentas ainda não citadas.

6.1 Arquitetura do Sistema

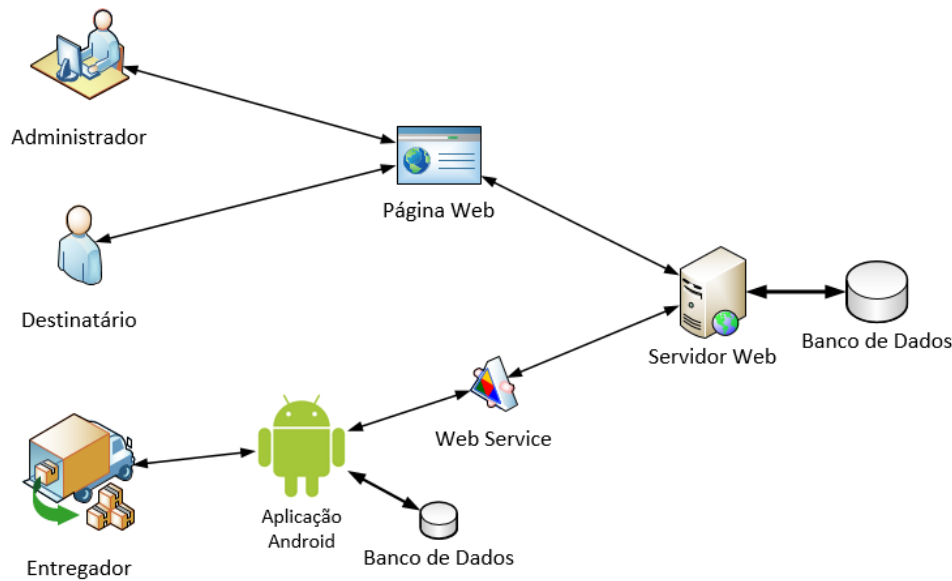
O protótipo de sistema deste projeto foi planejado a fim de obedecer todas os objetivos propostos. Simplificando para melhor entendimento, o sistema tem 5 partes básicas: um site responsivo, uma aplicação Android, o banco de dados, um Web Service e o *back-end* Web. Todas as 5 partes são essenciais para o funcionamento do sistema como um todo. Todas essas partes estão listadas na figura 7 que demonstra também como elas se conectam

O site responsivo é uma página Web e é a interface entre o Servidor e os usuários. Seja ele um gerente, administrador da empresa que irá utilizar o sistema ou o cliente, destinatário que acompanhará seus pedidos. Mas cada tipo de usuário tem suas próprias funções, os quais acessam áreas diferentes.

A aplicação Android, será utilizada pelos entregadores através de um dispositivo móvel. Ela vai ofertar as funções básicas utilizadas pela entregador através de uma conexão com o Web Service. Essa conexão entre dispositivo móvel e Web Service é feita através das conexões de rede sem fio do aparelho. A principal função desse dispositivo é a coleta da Geolocalização do entregador através de ferramentas presentes neles, e o envio ao sistema, fazendo assim a função de rastreamento. Cada dispositivo é identificado por um código diferente, fazendo com que cada um seja único para o sistema. Nesse caso utilizamos o endereço MAC dos dispositivos.

O Web Service faz a ponte de ligação entre os dispositivos móveis utilizados pelo

Figura 7 – Arquitetura do Sistema Archon



Fonte: Elaborada pelo autor.

entregadores e o Servidor Web. Ele padroniza a comunicação com o *back-end*, fazendo que qualquer dispositivo com sistema Android possa ser utilizado, aumentando assim a interoperabilidade.

O sistema gerenciador de banco de dados utilizado é o MySQL, nas próximas seções será apresentado a modelagem do banco de dados utilizado Além desse banco de dados cada dispositivo móvel carrega consigo uma base de dados, o SQLite, o qual guarda as informações dos pedidos de cada entregador

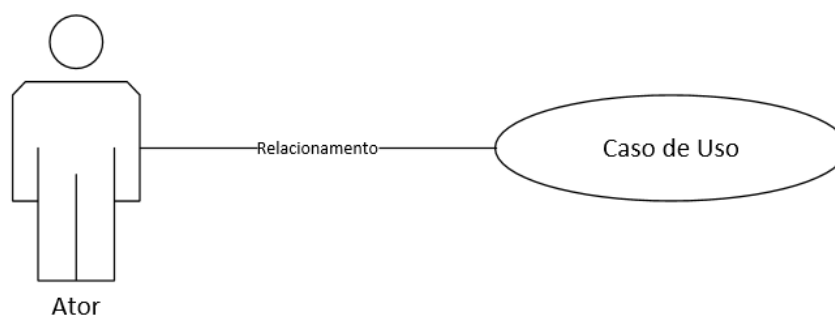
O Servidor Web ou *back-end* é a peça central que une todas as outras partes. Ele é o responsável pela maioria das funções do sistema. Ele possui todas as funções de acesso ao banco de dados, as funções do sistema, a modelagem das classes e como elas se comunicam com o usuário.

6.2 Diagrama de Caso de Uso

Os diagramas de casos de uso descrevem o que o sistema faz do ponto de vista de cada tipo de usuário. Com isso pode-se entender com mais facilidade quais são as funcionalidades do sistema e como cada usuário se conecta com ele e com outros usuários. Esses diagramas são compostos por basicamente 3 representações: Atores, casos de uso e o relacionamento entre estes dois elementos. A Figura 8 demonstra como cada um deles é representado visualmente.

Os Atores são representados por esses bonecos e uma legenda descreve o nome do

Figura 8 – Exemplo de Representação de um Diagrama de Caso de Uso



Fonte: Elaborada pelo autor.

ator que geralmente descreve uma função dentro de determinada empresa. Um ator pode ser um usuário humano ou um outro sistema computacional.

Os Casos de uso é representado por um círculo ou elipse que também possui um rótulo com o nome desse caso de uso. Ele vai definir uma função do sistema. Portanto a maioria deles vai definir uma ação de usuário que será rotulada com um verbo definindo qual a função.

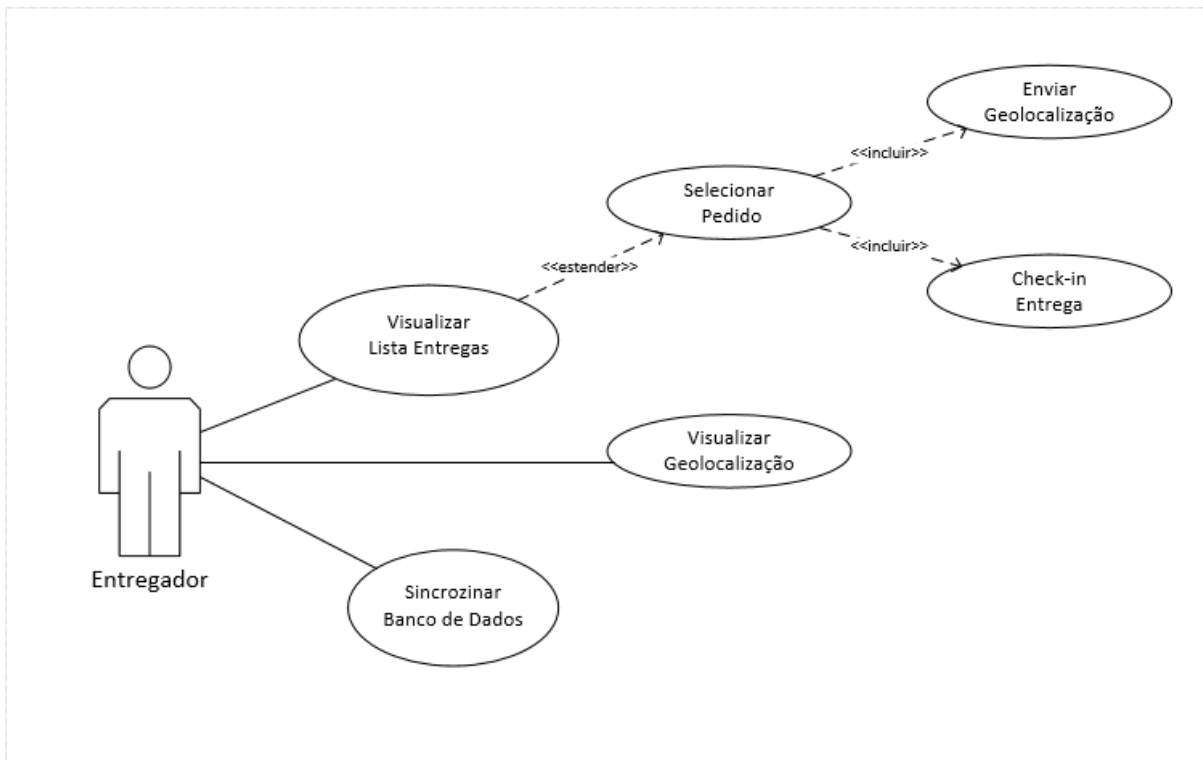
Por fim os relacionamentos representam as associações entre atores e casos de uso; existem generalizações entre atores e generalizações do tipo *extends* e *includes* que são empregadas entre os casos de uso. As associações entre atores e casos de uso vão definir uma funcionalidade do sistema do ponto de vista do usuário. Generalização entre atores hipotéticos chamados de 1 e 2, com uma seta indo de 1 para 2 dizem que os casos de uso de 2 também são casos de uso de 1, mas 1 tem seus próprios casos de uso. As generalizações *include*, de um caso de uso 1 para um 2, indica que 2 é essencial para o comportamento de 1. As generalizações *extend* de um caso de uso 2 para um caso de uso 1 indica que 2 pode ser acrescentado para descrever o comportamento de 1.

Durante o decorrer deste trabalho algumas perfis de usuários serão utilizadas para descrever atores e outros objetos importantes. Apenas para deixar mais claro, uma breve explicação sobre eles são: Entregador, quem utiliza o aplicativo Android e quem realiza a entrega dos Pedidos; Pedido, não é um ator mais é o nome dado ao objeto que está sendo entregue independente do que ele seja. Por fim Destinatário é quem espera receber o produto ou quem faz o rastreamento do mesmo.

Os diagramas a seguir foram projetados durante o desenvolvimento do protótipo de sistema feito no decorrer deste trabalho. Eles possuem 3 tipos de atores: Entregadores, Administradores e Destinatários. Para facilitar a compreensão cada ator será explicado

separadamente.

Figura 9 – Diagrama de Caso de Uso - Entregador



Fonte: Elaborada pelo autor.

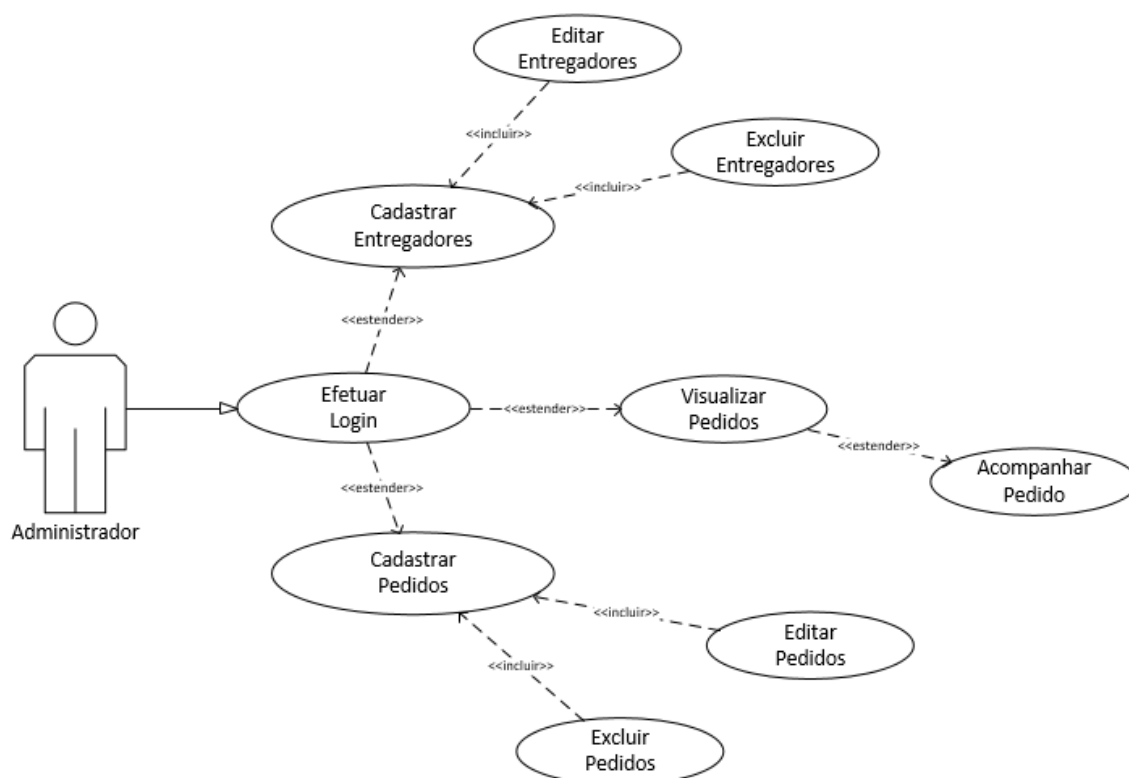
A Figura 9 apresenta o Entregador que ilustra os usuários do sistema que exercem essa mesma função. Eles utilizam o sistema através da aplicação móvel e tem sua principal função de enviar a sua geolocalização para o sistema a fim de realizar o rastreamento do pedido. O conceito desse sistema é ser genérico para ser utilizado em conjunto com qualquer aplicação que envolva esses 3 tipos de atores.

O Administrador representa os usuários que gerenciam uma empresa. A Figura 10 mostra o caso de uso do gerente que cuida dos pedidos em uma empresa. Eles utilizam o sistema através de um site responsivo, onde conseguem realizar todas essas funções. Como por exemplo, a de ver a exata posição dos seus entregadores e se o pedido já está em mãos do cliente.

O Destinatário representa os usuários do sistema que estão recebendo pedidos e dar a eles a opção de rastrear o pedido é o ponto principal deste trabalho, conforme a Figura 11. Eles ainda não possuem muitas funções no sistema, mas o objetivo final do sistema é aumentar essa comunicação entre os Entregadores e Destinatários.

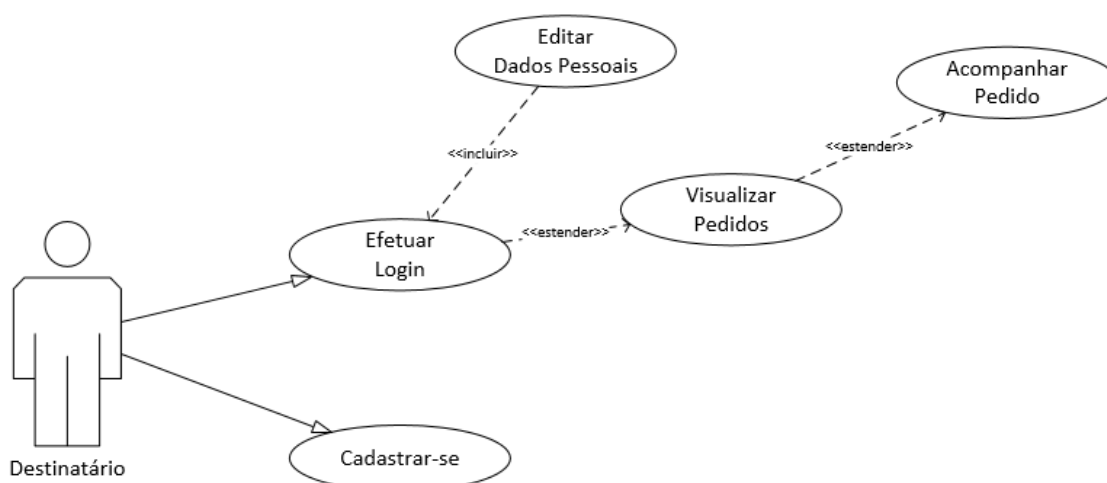
Todos os Atores do sistema ainda tem poucas funções implementadas. Por ser apenas um protótipo, o sistema apresentado aqui só tem as funções básicas de cada tipo de usuário. Como já dito, uma maior comunicação entre Entregadores e Destinatários

Figura 10 – Diagrama de Caso de Uso - Administrador



Fonte: Elaborada pelo autor.

Figura 11 – Diagrama de Caso de Uso - Destinatário

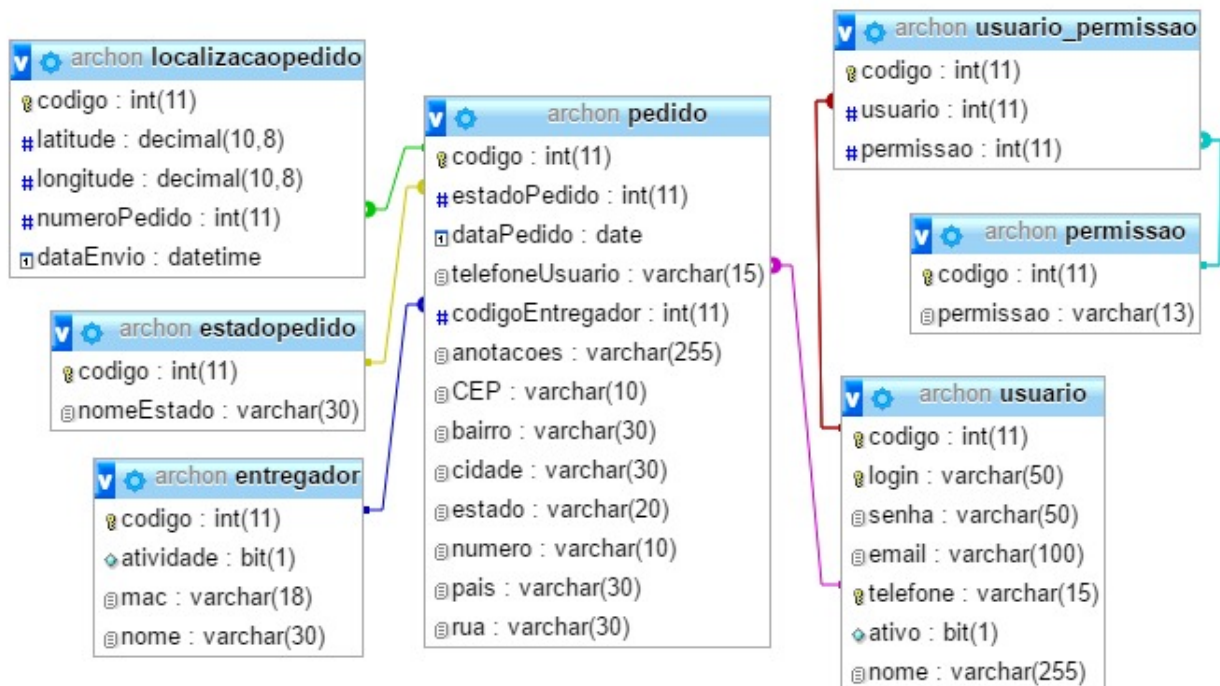


Fonte: Elaborada pelo autor.

é um dos principais focos do trabalho, mas isso não exclui a parte administrativa e de logística do sistema. Como proposta para trabalhos futuros em adição a este ficariam a adição de mais funções para os usuários.

6.3 Banco de Dados

Figura 12 – Modelo Lógico do Banco de Dados do Sistema



Fonte: Elaborada pelo autor.

A Figura 12 representa o banco de dados do sistema graficamente, gerada automaticamente pela ferramenta phpMyAdmin. Não é um banco de dados complexo, mas modelado a atender as necessidades básicas do sistema.

Pode-se perceber que a tabela mais importante é tabela “pedido”, devido a quantidade de atributos que ela possui e a quantidade de relacionamentos com as outras tabelas. Essa tabela representa os dados do Pedido, ela tem muitos atributos mas metade deles é designado para identificar o endereço da entrega. O Pedido tem um Usuario relacionado a ele, que designará seu Destinatário e um Entregador que designará quem seu entregador. Os Pedidos possuem também um EstadoPedido, que define o atual estado do pedido.

A tabela “localizacaopedido” representa os dados geográficos que definem a posição do Pedido. Está é a tabela com maior número de registros, visto que cada Pedido é ras-

treado por um bom período de tempo, ou seja, para cada Pedido uma grande quantidade de localizações é inserida.

A tabela “usuario” representa os Usuários do sistema. Esses usuários possuem uma permissão. Cada tipo de permissão define quais são as funções que cada usuário pode executar. No início apenas 2 permissões são habilitadas, a de Destinatários e a de Administradores. O endereço do sistema inicialmente era vinculado ao usuário, mas foi mudado para o pedido, visto que um usuário pode utilizar o serviço para enviar uma entrega para outra pessoa.

Por fim a tabela “entregador” representa os Entregadores. Os entregadores tem como um de seus atributos o mac que é um código que identifica o entregador pelo dispositivo móvel que ele utiliza.

6.4 Desenvolvimento Web Service

Durante o desenvolvimento do Web Service, algumas metas precisavam ser mantidas a fim de que o sistema funcionasse corretamente. Essa lista de requisitos foi gerada no planejamento e no desenvolvimento do projeto. E a lista de objetivos específicos do Web Service é a seguinte:

- 1) O Web Service precisa receber os dados de geolocalização do Aplicativo Android e saber indicar de qual Pedido é o dado recebido.
- 2) Ele precisa prover métodos de sincronização dos bancos de dados, visto que são diferentes e não estão conectados.
- 3) Ele deve estar disponível em tempo integral
- 4) Ele deve receber o código que representa o Entregador e verificar se o mesmo está vinculado ao pedido que está na mensagem visando a segurança dos dados do sistema.

Para cumprir o 1º objetivo, um método foi idealizado onde ele recebe as informações do aplicativo Android, e salva as informações em um banco de dados. Um dos parâmetros do método deve ser o identificador do Pedido, para que este possa ser registrado de forma correta.

Para o 2º objetivo, outro método foi idealizado onde o Entregador através do aplicativo faria uma requisição para o Web Service pedindo a ele que essa sincronia seja realizada.

O 3º objetivo é alcançado se o Web Service ficar ligado ao servidor de aplicação do sistema, assim tanto o sistema quanto o servidor estaria disponíveis.

E para o 4º e último, como o aplicativo Android não requer autenticação por parte do Usuário, a autenticação é feita pela Web Service. Logo, em todos os métodos da Web Service um identificador do Entregador deve ser testado a fim de manter a segurança do sistema.

6.4.1 Definição das Classes e Explicação dos Métodos

A criação do Web Service foi feita com o auxílio de um pacote da extensão do Java próprio para a criação desse tipo de tecnologia. O nome do pacote é bem sugestivo ao seu uso, `jws` (java web services) e contém várias tags `annotation` de que ao serem utilizadas configuram as classes como sendo Serviços e métodos.

A Figura 13 demonstra o código da interface que define o Web Service utilizado no Archon. Nele pode-se observar a utilização das anotações `@WebService` e `@WebMethod`. Além disso é pode-se notar a definição dos métodos ofertados por essa interface, são 4 no total onde um deles é apenas um método de testes.

A primeira anotação a `@WebService`, marca a classe como uma implementação de um Web Service ou uma interface como a definição de uma interface de Web Service, que é o caso apresentado. Ele possui ainda dois atributos; `name`, que define o nome do Web Service e `targetNamespace`, que define o endpoint que será marcado no Web Service.

A outra anotação a `@WebMethod`, configura o método onde ela foi utilizada para ser uma operation do Web Service associado. O método deve ser público e seus tipos de parâmetros e o tipo de retorno devem seguir as normas definidas no JAX-RPC 1.1.

As próximas imagens demonstram a classe que implementa a interface demonstrada acima. É essa classe que vai representar o Web Service, já que é nela que ficam a implementação dos métodos utilizados e é a partir dessa classe que o arquivo WSDL será gerado. A classe que implementa a interface acima recebe o nome de `ArchonWebServiceImpl`.

O primeiro método a ser demonstrado é o “hello”. Este método é utilizado apenas para teste do funcionamento do Web Service, logo seu funcionamento é bem simples. Ele recebe uma Mensagem do tipo `String` e retorna uma outra mensagem do mesmo tipo.

O próximo método é o “solicitaPedidos”, este é o método utilizado pelos Entregadores para sincronizar os bancos de dados do dispositivo móvel com o banco de dados do sistema ele é demonstrado na Figura 14. Como já dito, o dispositivo móvel utiliza um banco de dados menor apenas para auxiliar o sistema principal, logo os dois são separados. É um método que é utilizado repetidas vezes e tem como parâmetro de entrada o código do endereço MAC do dispositivo do Entregador. O endereço MAC é um código físico presente no *hardware* do dispositivo. No caso esse código é obtido através da aplicação

Figura 13 – Definição da Interface do Web Service do Sistema

```
1 package joao.archon.util;
2
3 import javax.jws.WebMethod;
4 import javax.jws.WebService;
5
6 @WebService(name = "ArchonWebService", targetNamespace =
7     "http://archon_webservice/")
8 public interface ArchonWebService {
9
10     @WebMethod(operationName = "hello")
11     public String hello(String name);
12
13     @WebMethod(operationName = "solicitaPedidos")
14     public int solicitaPedidos(String codigoEntregador);
15
16     @WebMethod(operationName = "atualizaEstadoPedido")
17     public int atualizaEstadoPedido(String codigoMACEntregador, Integer codigoPedido);
18
19     @WebMethod(operationName = "atualizaCoordenadas")
20     public String atualizaCoordenadas(String codigoMACEntregador, Integer codigoPedido,
21         String latitude, String Longitude);
22
23 }
```

Fonte: Elaborada pelo autor.

móvel e durante o cadastro do Entregador e esse código é utilizado para identificação do mesmo, por questões de segurança.

Acompanhando o código, na linha 53 da figura 14, podemos ver que uma lista com todos os pedidos do entregador é criada. Essa lista é obtida através do método listarPedidosEntregador da classe EntregadorVisao. Essa função possui 2 entradas, o código do Entregador e o Estado dos Pedidos que ser quer obter a lista. Pode-se observar que o estado setado como parâmetro é o valor “4”, que representa o estado “Não Sincronizado”, ou seja, os pedidos que ainda não foram sincronizados pelo dispositivo do Entregador. Logo, a função listarPedidosEntregador, nesse caso vai retornar todos os métodos que ainda não estão sincronizados com o dispositivo do Entregador.

A variável codigoPedido é o retorno da função. Inicialmente ela é setada com o valor “-1” um valor negativo para um código no banco de dados e que claramente não representa nenhum registro dele. Caso a função encontre Pedidos que ainda não foram sincronizados, ela retornará o código de um desses pedidos, caso contrário, retornará o valor “-1” que é tratado como erro pela aplicação.

Quando um pedido não sincronizado for encontrado, antes de terminar esse método e retornar o valor do pedido encontrado, a função atualizaEstadoPedido é executada, como pode ser observado na linha 62 da figura 14. Essa função tem 2 entradas, o código do Pedido e o novo Estado do Pedido. Pode-se observar que o estado setado como parâmetro é o valor “4”, que representa o estado “Aguardando Inicio Entrega”, ou seja os pedidos que

Figura 14 – Trecho de Código que Demonstra o método SolicitaPedidos

```
46 @Override
47 @WebMethod
48 public int solicitaPedidos(String codigoMACEntregador) {
49     EntregadorVisao eV = new EntregadorVisao();
50     PedidoVisao pV = new PedidoVisao();
51
52     int codigoPedido = -1;
53     List<Pedido> listaPedidosExistentes = pV.listarPedidosEntregador(
54         eV.buscarMAC(codigoMACEntregador).getCodigo(), 4);
55
56     if ((listaPedidosExistentes.isEmpty()) || (listaPedidosExistentes == null)) {
57         return codigoPedido;
58     } else {
59         Pedido pedido = listaPedidosExistentes.get(0);
60
61         codigoPedido = pedido.getCodigo();
62         pV.atualizaEstadoPedido(pedido.getCodigo(), 0);
63
64         return codigoPedido;
65     }
66 }
67
```

Fonte: Elaborada pelo autor.

estão aguardando para serem entregues. Como o pedido foi sincronizado pelo Entregador seu novo estado é aguardar que o entregador inicie a entrega do mesmo.

O método “atualizaEstadoPedido”, demonstrado na Figura 15, é utilizado pela aplicação móvel automaticamente quando o Entregador realiza alguma ação que irá atualizar o Estado de algum Pedido. Ele é chamado por exemplo, pela funções da aplicação onde o Entregador Inicia ou Finaliza a Entrega. Ele possui 2 parâmetros de entrada o endereço MAC do Entregador e o código do Pedido a ser atualizado.

Observando o código na linha 76 da figura 16, nota-se uma estrutura condicional. Essa estrutura verifica se o código do Pedido bate com o Entregador cadastrado como responsável por esse pedido. Essa verificação é feita por questões de segurança para garantir que o pedido possa ser atualizado somente pelo entregador responsável por ele. Essa estrutura utiliza o código que representa o registro do Entregador no banco de dados para comparações. Esse código é obtido através do endereço MAC recebido como parâmetro.

O retorno dessa função é um valor booleano descrito através de um valor inteiro. Retornará o valor “1”, como sendo verdadeiro, caso o estado do pedido seja atualizado com sucesso, e retornará o valor “0”, como sendo falso, caso essa operação não seja concluída.

Por fim, o método “atualizaCoordenadas”, demonstrado na figura 16, é o método principal do Web Service, e que realiza uma das principais funções desse sistema que é o rastreamento da geolocalização do Entregador. Este é método mais utilizado do Web Service, e é executado mais de uma vez por minuto para cada pedido que está sendo ras-

Figura 15 – Trecho de Código que Demonstra o método AtualizaEstadoPedido

```
69 @Override
70 @WebMethod
71 public int atualizaEstadoPedido(String codigoMACEntregador, Integer codigoPedido) {
72     PedidoVisao pV = new PedidoVisao();
73     EntregadorVisao eV = new EntregadorVisao();
74
75     pV.buscaPedidoCodigo(codigoPedido);
76     if ((pV.getPedido() != null)
77         && (pV.getPedido().getCodigoEntregador() ==
78             eV.buscarMAC(codigoMACEntregador).getCodigo())) {
79         pV.atualizaEstadoPedido(codigoPedido);
80         return 1;
81     } else
82         return 0;
83 }
84
85 }
86 }
```

Fonte: Elaborada pelo autor.

treado. Essa função recebe os dados de Localização por parâmetros, que são 4 o endereço MAC do Entregador, o código do Pedido que está sendo rastreado, a latitude e a longitude que representam os dados geográficos da posição do Entregador.

Como pode-se observar na linha 31 da Figura 16 existe uma estrutura condicional, ela serve para conferir se esse Entregador está realmente fazendo a entrega desse Pedido, apenas para segurança dos dados. Como explicado nas seções anteriores, essa segurança é feita conferindo o endereço MAC do entregador com o cadastrado no sistema. Já nas linhas 35,36,37 e 38 da mesma figura, tem-se a criação de um objeto de LocalizacaoPedido, que guarda as informações desse tipo das localizações, logo após isso as informações são salvas. o acesso ao objeto de LocalizacaoPedido é feito pela classe de visão que permite o acesso as operações de modificação dessa classe.

6.4.2 Criação WSDL e Acesso ao Serviço

E essas são as classes que definem o Web Service. Como dito, o eclipse com o pacote JEE foi utilizado no desenvolvimento do projeto. Como as marcações de @WebService e @WebMethod foram usadas, a IDE nos auxilia na geração dos componentes restantes para o funcionamento do serviço. O eclipse nos oferece funções que detectam as classes que definem os Web Services e a partir disso ele auxilia o usuário gerando os arquivos restantes e embutindo o serviço no servidor de aplicação que você utiliza para hospedar o sistema.

Ele também permite as configurações sobre como o Web Services vai se comportar quando estiver agindo como cliente e como estiver agindo como servidor. Além de monitorar o tráfego das mensagens recebidas e enviadas.

Figura 16 – Trecho de Código que Demonstra o método AtualizaCoordenadas

```

22 @Override
23 @WebMethod
24 public String atualizaCoordenadas(String codigoMACEntregador, Integer codigoPedido,
25     String latitude, String longitude) {
26
27     PedidoVisao pV = new PedidoVisao();
28     EntregadorVisao eV = new EntregadorVisao();
29
30     pV.buscaPedidoCodigo(codigoPedido);
31     if ((pV.getPedido() != null)
32         && (pV.getPedido().getCodigoEntregador() ==
33             eV.buscarMAC(codigoMACEntregador).getCodigo())) {
34
35         LocalizacaoPedidoVisao lpV = new LocalizacaoPedidoVisao();
36         lpV.getPedido().setLatitude(Float.parseFloat(latitude));
37         lpV.getPedido().setLongitude(Float.parseFloat(longitude));
38         lpV.getPedido().setNumeroPedido(codigoPedido);
39         lpV.salvar();
40
41         return "salvo";
42     }
43     return "não_salvo";
44 }
45

```

Fonte: Elaborada pelo autor.

O eclipse consegue realizar essas operações graças ao Axis. O Axis (Apache eXtensible Interaction System) é um *Framework* suportado pela Apache, que é uma implementação do protocolo SOAP. Ele oferece 2 formas de interpretar as classes Java como Web Service. a Primeira é utilizando arquivos JWS (Java Web Service) nativos do Axis e a outra que foi utilizada neste trabalho que é utilizando um arquivo WSDD (Web Service Deployment Descriptor) ele vai descrever como os vários componentes irão trabalhar para o funcionamento do Web Service.

Após gerar os arquivos necessários para a implantação basta apenas definir o servidor de aplicação onde o seu sistema está funcionando. Os arquivos WSDL e WSDD podem ser modificados para configuração do serviço. No caso deste trabalho foi trocado o endereço de localização para o endereço onde o serviço irá ser disponibilizado. Como o sistema não foi hospedado em na Internet, e os testes foram feito em uma rede local, o endereço usado foi um endereço local. Utilizando o endereço configurado já consegue-se ter acesso aos Serviços e adicionando a extensão “?wsdl” consegue-se acesso ao WSDL que conseqüentemente te mostra toda a descrição do serviço.

6.5 Desenvolvimento Aplicativo Android

Durante o desenvolvimento do aplicativo Android, algumas metas precisavam ser mantidas a fim de que o sistema funcionasse corretamente. Essa lista de requisitos foi

gerada no planejamento e no desenvolvimento do projeto. E a lista de objetivos específicos do aplicativo Android é a seguinte:

- 1) Listar para o Entregador os Pedidos que ele está responsável por fazer a entrega.
- 2) O aplicativo não irá realizar pedidos de autenticação para o Usuário, com o intuito de deixar a usabilidade mais simples. Logo, outra maneira deve ser implementada para suprir isso.
- 3) O Entregador pode escolher quais Pedidos ele quer realizar a Entrega
- 4) O Entregador pode escolher o momento em que ele pode iniciar e finalizar a Entrega.

E as maneiras encontradas para cumprir esses objetivos podem ser encontradas na demonstração do aplicativo logo abaixo.

6.5.1 Apresentação da Interface do Aplicativo Android

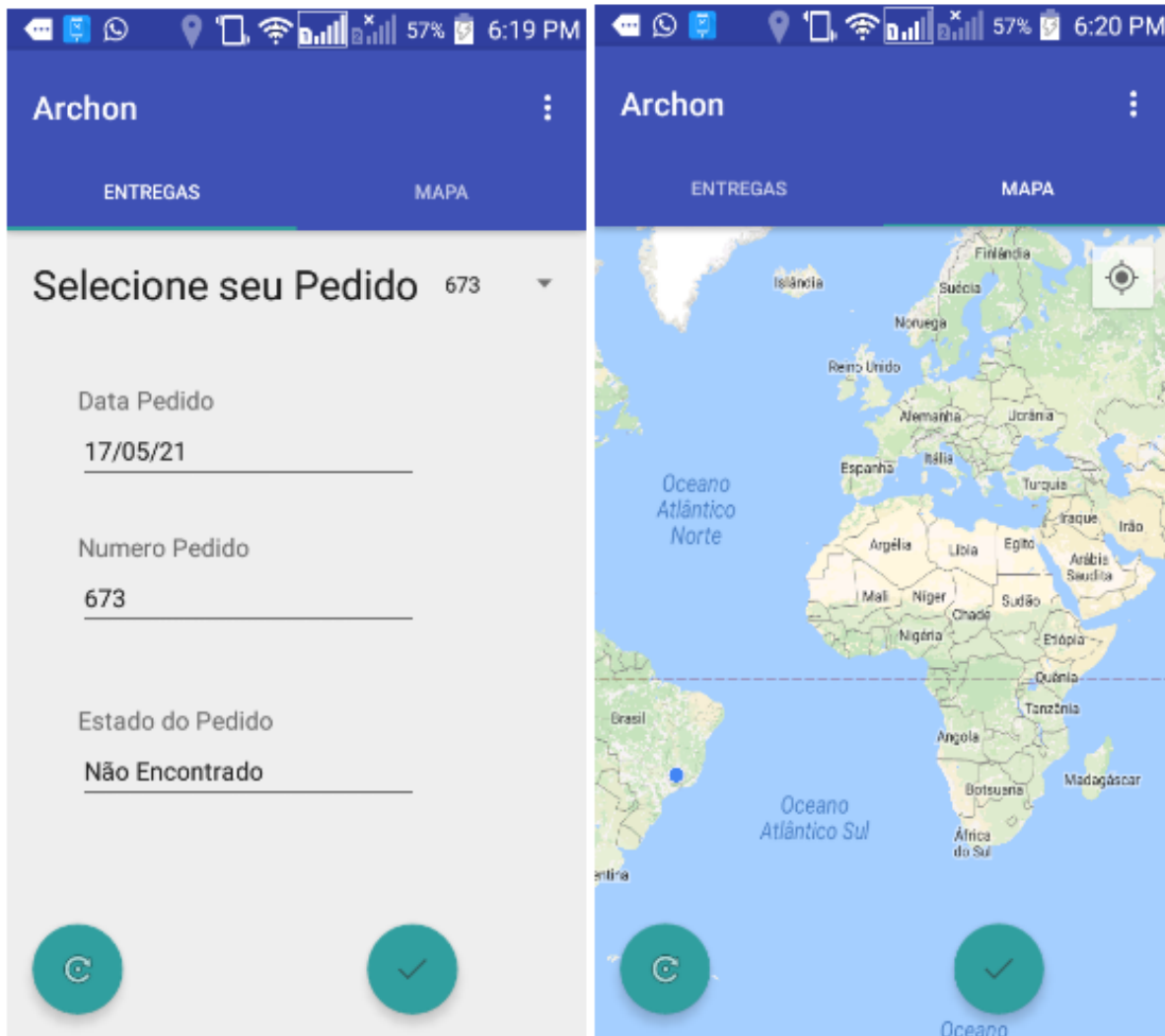
O aplicativo Android Archon foi desenvolvido a fim de cumprir com todos objetivos propostos para ele além de ter uma interface de fácil acesso e compreensão para os usuários. O usuário não precisa se autenticar com o sistema Archon ao utilizar o aplicativo, mas precisa ter acesso a internet e manter o componente GPS do dispositivo ligado. Todas as opções do sistema podem ser acessadas de forma rápida.

A Figura 17 apresenta a interface da tela principal do aplicativo, dois *Fragments*, o da esquerda denomina-se “Entrega” o da direita “Mapa”. Como pode-se observar a interface possui 2 abas superiores que alternam entre essas 2 *Fragments*. Esses abas são áreas selecionáveis que alternam o contexto central da interface. O *Fragment* “Entrega” é o responsável por mostrar ao usuário as informações dos Pedidos que ele foi encarregado. Já o *Fragment* “Mapa” demonstra ao usuário a sua geolocalização, assim ele pode se orientar e determinar sua posição. O botão no canto superior direito desta *Fragment*, é um botão que ao ser acionado demonstra no mapa a posição atual do Entregador.

No canto superior da Figura 18, imagem a esquerda o texto “Selecione o seu Pedido” é acompanhado ao seu lado direito por um Spinner. Um Spinner é um componente de seleção, que nesse caso vai permitir ao usuário selecionar um Pedido. Na figura 18, a esquerda um enfoque sobre o componente Spinner e a direita uma demonstração da *Fragment* ao selecionar o Spinner onde uma lista dos Pedidos do Entregador que estão sincronizados ao dispositivo aparecem.

Independente da interface que está sendo acessada consegue-se observar alguns botões na parte inferior da tela. Esses botões são componentes *FloatingActionButton*, e este tipo de componente aparece não importa o *Fragment* que esteja selecionado. São 3 botões, um no canto inferior esquerdo e os outros 2 no canto direito. A Figura 19 demonstra os 3 tipos de botões existentes.

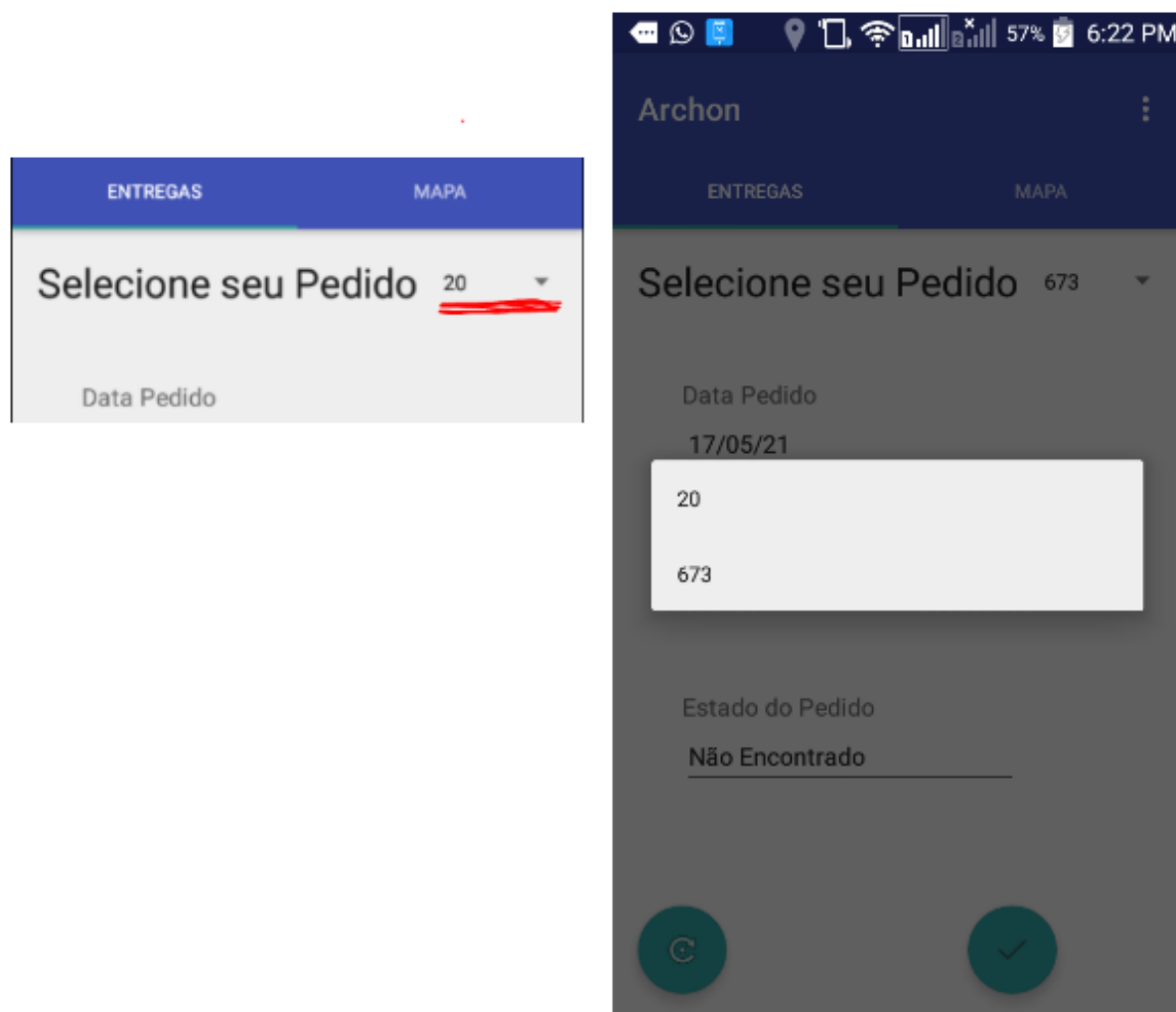
Figura 17 – Interface do Aplicativo Android Archon



Fonte: Elaborada pelo autor.

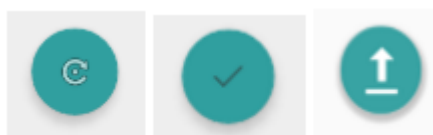
O primeiro botão, à esquerda na Figura 19 é o botão utilizado pelo usuário para a sincronização do banco de dados. O terceiro, à direita na Figura 19, é o botão de iniciar o rastreamento da entrega, ao selecionar um Pedido utilizando o Spinner citado acima, ao clicar nesse botão o rastreamento do Pedido é iniciado. O segundo, ao meio na Figura 19, é o botão de finalizar o rastreamento da entrega, quando o entregador chegar ao seu destino esse botão tem a funcionalidade de interromper o rastreamento do Pedido. Esses dois últimos botões citados só aparecem em determinadas situações, quando cada um deles puder ser utilizado e isso é relativo ao estado do Pedido Selecionado.

Figura 18 – Interface do Seletor de Pedidos do Aplicativo Android Archon



Fonte: Elaborada pelo autor.

Figura 19 – Botões de funcionalidade do Aplicativo Android Archon



Fonte: Elaborada pelo autor.

6.5.2 Definição das Classes e Explicação dos Métodos

Na seção anterior pôde-se conhecer a interface do sistema, nesta seção serão abordadas as implementações dos métodos e sobre algumas funcionalidades do sistema. Essa abordagem será feita com apresentação e explicação do código fonte da função a ser demonstrada. As explicações a seguir são das partes consideradas mais importantes para o funcionamento do Aplicativo e as partes que atendem aos objetivos próprios do Aplicativo.

6.5.2.1 Comunicação Com o Web Service

A comunicação entre o aplicativo Android e o Web Service é feita através de vários métodos, classes e threads. A principal e mais importante classe é a SoapRequests. A comunicação direta com o Web Service é feita por essa classe. Ela é composta por métodos que acessam ao Serviço fazendo requisições e enviando informações.

Essa comunicação só é possível graças a biblioteca ksoap2 (KSOAP, 2017). Originalmente ele foi desenvolvido para trabalhar com J2ME, mas atualizada por Manfred Moser (manfred@simpligility.com) para trabalhar com Android. Ela oferece uma biblioteca de cliente SOAP, que é composta por classes que representam objetos SOAP, Envelopes SOAP e que realizam a comunicação com o Web Service.

Figura 20 – Trecho de Código da Definição da Classe SoapRequests

```
public class SoapRequests implements Serializable {  
  
    private static final boolean DEBUG_SOAP_REQUEST_RESPONSE = true;  
    private static final String NAMESPACE = "https://http://util.archon.joao";  
    private static final String MAIN_REQUEST_URL =  
        "http://192.168.137.1:8080/archon/services/ArchonWebServiceImpl?wsdl";  
    private static final String SOAP_ACTION = "http://util.archon.joao/hello";  
}
```

Fonte: Elaborada pelo autor.

A Figura 20 demonstra a definição da classe SoapRequests e seus atributos. Esses atributos são necessários para a conexão com o Web Service, são utilizados pelos objetos da biblioteca ksoap2 para realizar essa conexão. Já a figura 21 demonstra um dos métodos presentes na classe, para demonstrar como é a realizada a comunicação.

Para desenvolver um método que se comunica com o Web Service precisa se conhecer os serviços que ela oferece. A função demonstrada na figura é para se comunicar com o método “hello” do Web Service, como pode-se notar na linha 114 o nome do método é aplicado a uma variável. A funcionalidade desse método no serviço é apenas para testes ele recebe uma mensagem e retorna uma mensagem. A mensagem a ser enviada está sendo recebida como parâmetro dessa função e a mensagem recebida do serviço está sendo retornado por essa função.

Figura 21 – Trecho de Código de uma Função que se Comunica com o Web Service

```
112 public String getMessage(String mensagem) {
113     String data = "Inicializada";
114     String methodname = "hello";
115
116     SoapObject request = new SoapObject(NAMESPACE, methodname);
117     request.addProperty("name", mensagem);
118
119     SoapSerializationEnvelope envelope = getSoapSerializationEnvelope(request);
120
121     HttpTransportSE ht = getHttpTransportSE();
122     try {
123         ht.call(SOAP_ACTION, envelope);
124         testHttpResponse(ht);
125         SoapPrimitive respostaWebService = (SoapPrimitive) envelope.getResponse();
126         return respostaWebService.toString();
    }
```

Fonte: Elaborada pelo autor.

O objeto `SoapObject` definido na linha 116 da figura 21, é um objeto obtido pela biblioteca `ksoap2`. Ele é um objeto simples que armazena o Namespace do Web Service e o nome do método que se pretende acessar. Adicionalmente a esses valores ele também armazena os parâmetros do método que ele pretende acessar, como demonstrado na linha 117. Esses parâmetros são definidos pelo Web Service, e esse é outro ponto que mostra que deve se conhecer o serviço para qual se está desenvolvendo métodos de consumo.

Na linha 119 da figura 21 o envelope que será utilizado para comunicação é construído, ele se utiliza do objeto criado anteriormente. Como já citado, o SOAP se utiliza de protocolos da internet para se comunicar, e nesse caso não é diferente. O objeto inicializado na linha 121, realiza a comunicação com serviço através do protocolo HTTP. Ele também é obtido pela biblioteca `ksoap2` e internamente realiza a comunicação utilizando o HTTP. A função `call` executada logo em seguida é a que realiza a comunicação.

Até esse momento a comunicação já foi realizada e caso o serviço tenha retornado alguma informação, ela está armazenada no envelope que foi passado como parâmetro para a função que realizou a comunicação. O objeto que está recebendo o conteúdo do envelope é bem simples e contém apenas o Namespace do serviço o nome da variável de retorno e o valor dela. Ao utilizar o método `toString()` apenas se está imprimindo o valor da variável do envelope.

Esse é o funcionamento base para qualquer outro método de consumo dos serviços do Web Service. O que tem de diferente entre os outros métodos é exatamente o objetivo do método, a comunicação é feita da mesma forma. Com essa classe o consumo dos serviços já pode ser feito.

Contudo os métodos citados até agora soltam exceções `SocketTimeoutException` e `IOException` por isso não podem ser executados em uma Activity principal, além disso não

faz sentido executar esses métodos na atividade principal, visto que alguns deles executam por longos períodos de tempo o que fariam o aplicativo ficar parado enquanto os métodos ficariam executando.

A saída é utilizar Threads para que o processamento seja feito em segundo plano e caso ocorra algum erro ele não ocorra diretamente na atividade principal. No Android existem diferentes formas de se implementar Threads, as usadas neste trabalho foram as Threads em modo puro através de Runnable e de uma classe própria do android chamada AsyncTask. A função que executa o rastreamento dos pedidos, é executada por longos períodos e pode ser executada até em paralelo, caso o Entregador esteja entregando mais de um pedido. Por esse motivo ela foi implementada com Runnable. Já as funções de sincronizar os bancos de dados e de atualizar os estados do pedido como são executadas uma quantidade menor de vezes e não precisam ficar tanto tempo executando, foram implementadas com AsyncTask. O AsyncTask é uma Thread mas com tempo de vida já programado por algumas tarefas pré estabelecidas, executando cerca de 3 funções e depois se encerrando. Ele também tem uma interface gráfica de carregamento que pode auxiliar os usuários nesse caso.

Como não existe autenticação no sistema por parte do usuário do aplicativo, a maneira encontrada para prezar a segurança das mensagens trocadas foi a criação de um código específico para cada Entregador. Esse código seria cadastrado no sistema junto a outras informações do Entregador e é ele que validará as mensagens enviadas ao serviço pela aplicação. O código utilizado até o momento tem sido o endereço MAC do dispositivo de comunicação Wifi do smartphone. A figura 22, representa um trecho de código onde uma função consome um serviço da Web Service através de uma Thread e utiliza-se deste código para realizar a validação.

Figura 22 – Trecho de Código da Chamada da Thread que se Comunica com o Web Service

```
237 WifiManager manager = (WifiManager) getSystemService(Context.WIFI_SERVICE);
238 WifiInfo info = manager.getConnectionInfo();
239 ThreadAtualizaEstadoPedido tAp =
240     new ThreadAtualizaEstadoPedido(getBaseContext(),
241                                     info.getMacAddress(),
242                                     itemListaPedidos , view );
243 tAp.execute();
```

Fonte: Elaborada pelo autor.

A classe ThreadAtualizaEstadoPedido, utiliza-se do AsyncTask para realizar a comunicação com o Web Service. um dos seus parâmetros, demonstrado na linha 241 da Figura 22 é o endereço MAC do WiFi do dispositivo Android. Ele é obtido pela classe WifiManager que é básica do Android. Quando o Web Service receber o envelope, ele

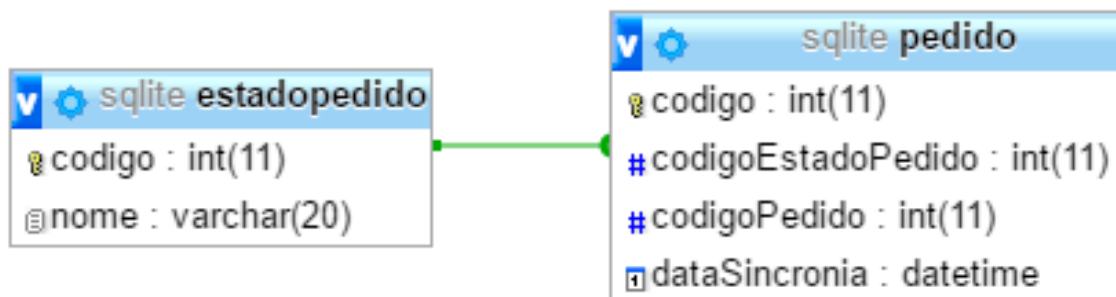
verificará se o endereço transportado nele é de algum Entregador cadastrado e caso sim executará alguma função.

6.5.2.2 Utilização do SQLite

Para a utilização do SQLite uma classe foi criada. O DbHelper extend uma classe básica do Android que é o SQLiteOpenHelper. Dessa forma o acesso ao banco de dados não é feito diretamente e assim pode-se instanciar várias classes DbHelpers para a manipulação dos dados.

O Banco de dados utilizado por enquanto, representado pela Figura 23 é bem simples e só guarda informações muito necessárias para o funcionamento do sistema, são eles os registros de Pedidos e seus Estados. Na tabela Pedidos se encontram os campos de codigoPedido e data da Sincronia, além do código que representa o estado do Pedido. A tabela EstadoPedido guarda os tipos de pedidos existentes.

Figura 23 – Representação do Banco de Dados utilizado no SQLite



Fonte: Elaborada pelo autor.

A Figura 24 demonstra a definição da classe DbHelper e seus atributos. O principal atributo é o representado pela linha 14, que se refere ao nome do banco de dados utilizado.

Figura 24 – Trecho de Código da Função Responsável por Adicionar Pedidos no SQLite

```

12 public class DbHelper extends SQLiteOpenHelper {
13
14     public static final String DATABASE_NAME = "ArchonDb.db";
15     public static final String DATABASE_ID_NAME = "codigo";
16
17     public DbHelper(Context context) { super(context, DATABASE_NAME, null, 1); }
20

```

Fonte: Elaborada pelo autor.

Já a função para inserção de Pedidos no banco de dados está demonstrada na figura 25. Para realizar a inserção de registro no banco de dados, se utiliza própria do Android chamada SQLiteDatabase, demonstrada na linha 121 da Figura 25. É um objeto instanciado através do DBHelper, que utiliza o ContentValues que nada mais é que uma lista de valores, para que possa manipular os dados no banco de dados.

Figura 25 – Trecho de Código da Função Responsável por inserir valores no banco de dados SQLite do Aplicativo

```
119 public boolean inserirPedido(Integer codigoPedido) {
120     SQLiteDatabase db = this.getWritableDatabase();
121     ContentValues contentValues = new ContentValues();
122
123     contentValues.put("codigo", codigoPedido);
124     contentValues.put("estado", 0);
125     db.insert("Pedidos", null, contentValues);
126     return true;
127
128 }
```

Fonte: Elaborada pelo autor.

6.5.2.3 Obtenção da Geolocalização

A posição geográfica do dispositivo é obtida com o auxílio do dispositivo de localização do smartphone e da classe GoogleApiClient do pacote básico do Android. Uma combinação dos dois ajudam a obter a localização com maior precisão. A localização é descrita através de um objeto Location, que possui diferentes atributos que descrevem a geolocalização. A Figura 26 demonstra como o objeto mLastLocation do tipo Location sendo inicializada, esse objeto salva a última localização.

Figura 26 – Trecho de Código da Função Responsável por Iniciar o valor da Posição Geográfica do Aplicativo

```
81 @Override
82 public void onConnected(@Nullable Bundle bundle) {
83     mLastLocation = LocationServices.FusedLocationApi.getLastLocation(mGoogleApiClient);
84     createLocationRequest();
85 }
```

Fonte: Elaborada pelo autor.

Essa função onConnected, é executada automaticamente assim que o dispositivo se conecta com o Google Play Services, e assim já altera o valor do objeto mLastLocation. A função da linha 84 inicia o LocationRequest que são objetos utilizados para melhorar a

qualidade do serviço do FusedLocationProviderApi, que é a API que o google play services utiliza para lhe prover as informações. A Figura 27 demonstra como ela é inicializada.

Figura 27 – Trecho de Código da Definição dos valores do LocationRequest

```
91     protected void createLocationRequest() {
92         mLocationRequest = new LocationRequest();
93         mLocationRequest.setInterval(10000);
94         mLocationRequest.setFastestInterval(5000);
95         mLocationRequest.setPriority(LocationRequest.PRIORITY_HIGH_ACCURACY);
96     }
```

Fonte: Elaborada pelo autor.

Mas a função mais importante é a demonstrada na Figura 28, que é outra função executada automaticamente pelo aplicativo. Essa função é executada sempre que a localização é alterada. Ao fazer isso automaticamente ela troca o valor do objeto mLastLocation que será o objeto que irá prover todas as informações de geolocalização para o sistema.

Figura 28 – Trecho de Código da Função Responsável por Atualizar o valor da Posição Geográfica do Aplicativo

```
130     @Override
131     public void onLocationChanged(Location location) {
132
133         mLastLocation.setLatitude(location.getLatitude());
134         mLastLocation.setLongitude(location.getLongitude());
135     }
```

Fonte: Elaborada pelo autor.

6.6 Desenvolvimento Site Responsivo

O site Responsivo é a interface gráfica que interliga os Destinatários e os Administradores ao sistema. Todas as funções que esses tipos de usuário possuem podem ser encontrados nele e elas variam de usuário para usuário dependendo da sua permissão. Contudo o site é apenas uma interface. Como citado anteriormente, esse sistema pode ser utilizado em qualquer área que possa se interessar. Para isso apenas essa interface, o site, seria alterado, para que ele possa atender aos requisitos do novo local onde ele está. Logo, o que será apresentado nesta seção é o sistema como um todo através do site.

Durante o desenvolvimento do Site, algumas metas precisavam ser mantidas a fim de que o sistema funcionasse corretamente. Essa lista de requisitos foi gerada no planejamento e no desenvolvimento do projeto. E a lista de objetivos específicos para o Site é a seguinte:

- 1) O site deve ser a interface para os Destinatários e Administradores, logo ele deve diferenciar os dois tipos de usuário e lhe apresentar as funções que eles têm acesso.
- 2) O site deve ser Responsivo, o que quer dizer que ele e seus componentes devem alterar seus layouts com base no tamanho de tela onde eles está sendo exibido.
- 3) Para o Administrador o site deve apresentar as funções de gerenciamento de Entregadores e Pedidos.
- 4) Para o Destinatário o site deve apresentar uma lista com os Pedidos relacionados a aquele usuário, assim como a opção de acompanhar a entrega de um pedido a sua escolha.
- 5) A página de Acompanhamento do Pedido, deve oferecer as informações básicas do pedido, as informações de endereço dele assim como um mapa que demonstra em tempo real a posição do Pedido.

E as maneiras encontradas para cumprir esses objetivos podem ser encontradas na demonstração da interface do site responsivo na próxima seção.

6.6.1 Apresentação da Interface do Site

O Site do Sistema Archon foi desenvolvido a fim de cumprir com todos objetivos propostos para ele além de ter uma interface de simples compreensão que apresente os conceitos do sistema para o usuário, além de uma outra interface, com mais opções e seja mais ajustada para o meio administrativo.

Para acessar essa segunda área do site o usuário precisa se autenticar com login e senhas criadas no momento do cadastro. A Figura 29 demonstra, à esquerda, o formulário de login para a autenticação no sistema.

Já o formulário à direita na figura 29, demonstra o formulário de cadastro para novos usuários do sistema. Ele solicita algumas informações necessárias para o cadastro do usuário. Todos os usuários cadastrados desta forma são automaticamente do tipo Destinatário. Os usuários do tipo Administrador só podem ser cadastrados pelos Gerentes do Sistema Archon.

A parte de autenticação do usuário foi desenvolvida com o *Framework* de aplicação Spring. Durante a configuração do sistema você informa a ele as informações que diferenciam os tipos de usuário e as informações de autenticação e o spring realiza os procedimentos além de filtrar os conteúdos que cada tipo de usuário pode acessar. O spring também barra o acesso ao sistema por visitantes, que ainda não possuem login.

Após a autenticação o usuário ganha acesso a área restrita do sistema que varia de acordo com o tipo de permissão do usuário. A visão do destinatário é semelhante a

Figura 29 – Interface de Login e Cadastro do Sistema Archon

The image displays two side-by-side screenshots of the Archon system's user interface. The left screenshot shows the login page, titled 'Archon' and 'SISTEMA PARA MONITORAMENTO EM TEMPO REAL DE PRODUTOS E VEÍCULOS'. It prompts the user to 'Digite suas Credenciais Para Entrar no Sistema' and provides input fields for 'Login' and 'Password', along with a 'Logar' button and a 'Registre-se' link. The right screenshot shows the registration page, titled 'Archon' and 'Registre um novo Usuario'. It includes input fields for 'Nome', 'Login', 'Email', 'Telefone', 'Senha', and 'Confirme a Senha', a 'Registre-se' button, and a link for 'Já possui uma conta'.

Fonte: Elaborada pelo autor.

do Administrador com exceção de que ele tem acesso apenas a 2 funções. A de listar os Pedidos e Acompanhar Pedidos. Logo, apenas a Visão do administrador será analisada.

6.6.2 Apresentação do Sistema Pela Visão do Administrador

Assim que o Administrador se autentica no sistema ele acessa uma página web demonstrada na Figura 30. Esse *template* é o AdminLTE (AdminLTE, 2017), é um modelo de código aberto para sistemas de administração e painéis de controle. É um modelo HTML responsivo baseado na estrutura CSS do Bootstrap (Bootstrap, 2017). É um projeto totalmente modular que permite que ele seja personalizado e construído. Apesar disso como alguns componentes do Primefaces definem classes CSS que são as mesmas utilizadas pelo AdminLTE, logo alguns componentes não sabem ao certo, qual especificidade do CSS ele deve seguir.

Por essa figura consegue-se entender como é formado a interface base do sistema. Um menu na lateral esquerda mostra as opções que ele pode acessar. Uma barra superior com algumas outras funções. E a direita o espaço para o conteúdo da página, que apresenta

uma mensagem de boas vindas.

Figura 30 – Tela *HomePage* do Sistema Archon



Fonte: Elaborada pelo autor.

O menu na lateral esquerda é um menu do tipo multilevel. O identificador na parte superior de cada opção indica a que tipo de alvo as funções se aplicam. A opção selecionada demonstra as opções para acessar a área de Pedidos.

A página Lista de Pedidos, demonstrada pela figura 31, exibe ao usuário todos os pedidos que fazem referência a aquele usuário. Caso ele seja o administrador do sistema, todos os pedidos serão mostrados para que ele possa ter uma visão do sistema como um todo. A tabela tem as informações básicas de cada Pedido e uma coluna com a marcação “Acompanhar Pedido”. Ao clicar no ícone desta coluna o usuário é direcionado para a página de acompanhamento do pedido selecionado

Na parte inferior da tabela um botão com a Descrição “Cadastrar Pedidos”. Este botão ao ser acionado ativa uma painel que abre uma interface para criação de novos Pedidos, como demonstra a Figura 32.

Nesta tela vemos alguns campos para preenchimento dos dados do novo pedido. alguns desses campos já possuem alguns valores sugeridos, ou autocompletam os valores inseridos. O campo do Nome do Entregador, só mostram os nomes de Entregadores cadastrados no sistema. Já o campo de Telefone do Destinatário, autocompleta o valor caso o telefone do já esteja presente no sistema

Ao clicar na opção de Acompanhamento de Pedidos na tabela demonstrada na Figura 31, o usuário é redirecionado para a tela de Acompanhamento do Pedido. Essa

Figura 31 – Tela de Listagem de Pedidos do Sistema Archon

Numero do Pedido	Telefone do Destinatario	Codigo Entregador	Estado Pedido	Acompanhar Pedido
20	37991979406	7	Pedido Entregue	
22	37991979406	7	Aguardando Inicio da Entrega	
45	99999	7	Aguardando Inicio da Entrega	
47	99999	7	Aguardando Inicio da Entrega	
665	37991979406	7	Aguardando Inicio da Entrega	
673	37991979406	7	A Ser Entregue para o Entregador	

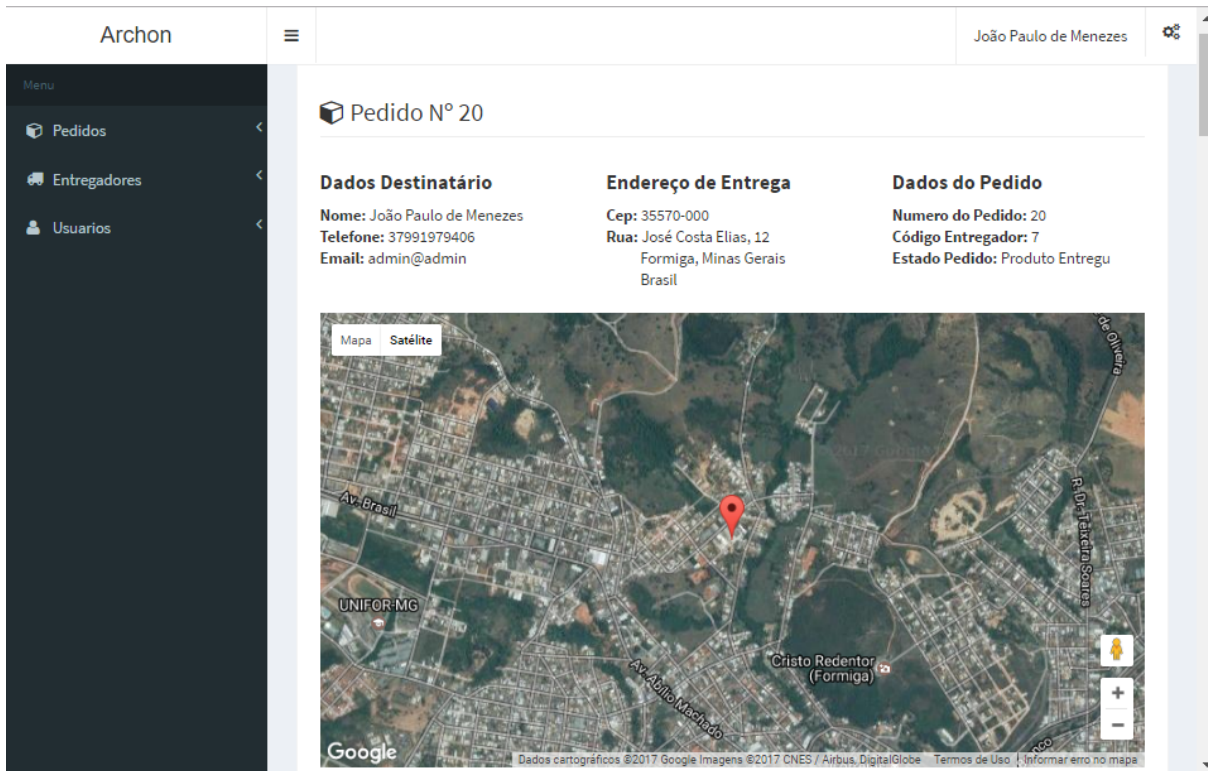
Fonte: Elaborada pelo autor.

Figura 32 – Tela de Cadastro de Pedidos do Sistema Archon

Fonte: Elaborada pelo autor.

página dá informações mais completas sobre o pedido além de mostrar através de um mapa a posição geográfica do Entregador que está transportando o seu pedido. A Figura 33 demonstra a página de acompanhamento do pedido após o usuário ter clicado em acompanhar o pedido de código 20. O mapa e acompanhamento do pedido, é atualizado via Ajax em intervalos curtos de tempo sempre pegando a última localização do pedido.

Figura 33 – Tela de Acompanhamento de Pedidos do Sistema Archon



Fonte: Elaborada pelo autor.

Esse mapa é utilizado com o auxílio de um componente do Primefaces chamado GMap. Sua utilização é feita com o uso da tag `<p:gmap>` no arquivo xhtml. Essa tag utiliza da API do Google Maps Javascript para mostrar o mapa e suas opções na tela. Ela tem 4 atributos principais: `center`, que define o centro do mapa quando ele for carregado; `zoom`, que define a proximidade da visão do mapa quando ele for carregado; `type`, que define o tipo de mapa são 4 tipos básicos disponíveis; e `style` que é utilizado para definir as dimensões do mapa na página Web.

A figura 34, demonstra o trecho de código que expressa o mapa. Pode-se observar na linha 74 a definição da tag `<p:gmap>`. São passados para o componente os valores da localização do pedido, obtidos pela view `UtilitiesBean` além de alguns parâmetros que não se alteram.

A linha 73 demonstra a utilização do componente responsável pela atualização do formulário via Ajax. O componente executa o método demonstrado no atributo `lis-`

Figura 34 – Trecho de código Responsável pela exibição do Mapa na Tela de Acompanhamento de Pedidos

```
72 @ <h:form id="map">
73     <p:poll interval="120" listener="#{pollUtilities.atualiza}" update="map" />
74     <p:gmap center="#{utilitiesBean.ultimaLocalizacao.latitude},
75             #{utilitiesBean.ultimaLocalizacao.longitude}"
76            zoom="15" type="HYBRID" style="width:100%;height:400px" model="#{utilitiesBean.modelo}" />
77 </h:form>
78
```

Fonte: Elaborada pelo autor.

tener, que nesse caso é a atualização das informações de localização do pedido, a cada 120 segundos, valor demonstrado no atributo interval. O formulário a ser atualizado é o demonstrado no atributo update.

Para que essa API do Google Maps funcione corretamente uma tag <script> deve ser inserida na seção Head da página web. A tag está demonstrada logo abaixo. Pode-se perceber que ela é formada pelos atributos src e type. O atributo src apresenta um endereço para uma página web que possui alguns parâmetros, um desses parâmetros é a chave da API, denominada por key. Essa chave pode ser obtida no site da API do Google Maps.

```
<script src="http://maps.google.com/maps/api/js?sensor=false&
        key=InformeSuaKey"
        type="text/javascript" />

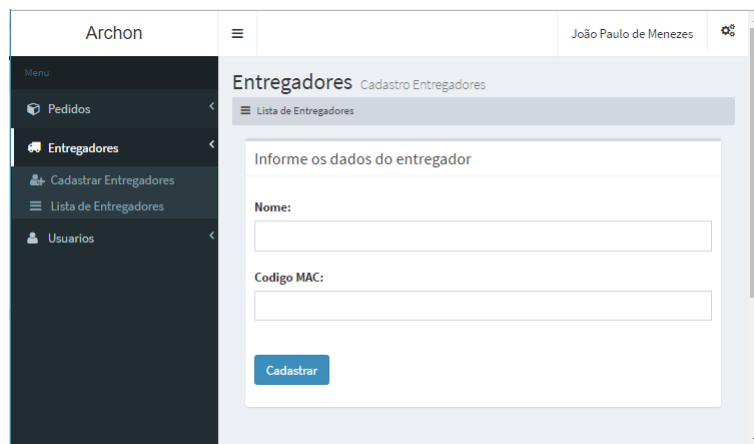
}
```

A opção do Menu lateral esquerdo sobre os entregadores, oferece 2 opções, a de Cadastrar e a de Listar os Entregadores. A Tela de cadastro dos Entregadores é bem simples e exige apenas o nome do Entregador e o endereço MAC obtido pelo Aplicativo Android que o Entregador irá utilizar. Pode-se ver a tela de cadastro de entregadores pela Figura 35.

6.7 Testes

O sistema não foi hospedado na internet, logo os testes realizados foram todos feitos numa rede local. O ambiente de testes utilizado é composto de uma rede wi-fi e de dois dispositivos necessarios para acessar o sistema. A rede utilizada foi uma Rede Wifi com um raio de aproximadamente 60 metros, devido as interferencias no local de testes. O sistema estava em um servidor Apache Tomcat local e foram usados dois dispositivos diferentes para realizar os envios dos dados geográficos. O primeiro foi um *smartphone*, o

Figura 35 – Tela de Cadastro de Entregadores do Sistema Archon



Fonte: Elaborada pelo autor.

mesmo citado na seção de materiais e métodos, e o outro o Emulador do Android Studio demonstrado na figura 36.

Para os testes o aplicativo foram instalados no emulador e no *smartphone* e então uma serie de casos de teste foram executados. O sistema teve todas as suas funções mapeadas pelo caso de teste que era composto da seguinte sequência de ações.

1. Realizar Login no sistema como um Administrador.
2. Cadastrar um Entregador. Essa função demanda executar o aplicativo para obter o código do Entregador gerado por ele.
3. Cadastrar um Pedido no site utilizando o Entregador cadastrado anteriormente.
4. Sincronizar os Pedidos do Entregador através do aplicativo.
5. Iniciar e entrega através do aplicativo.
6. Monitorar a entrega como um administrador através do site.
7. Cadastrar um novo usuário destinatário com o numero de telefone utilizado no cadastro de pedido.
8. Monitorar a entrega como o destinatário através do site.
9. Concluir a entrega do Pedido através do aplicativo e exibir o estado do Pedido.
10. The third etc ...

Não há muitos dados para serem obtidos desses testes a não ser a posição geográfica dos entregadores mas o objetivo dos testes era a verificação de cada função implementada

Figura 36 – Emulador de Dispositivo Android



Fonte: Retirada do Android Studio.

no sistema. O sistema ainda não possui nenhum tipo de gerador de relatório e pela falta de muitos dados nenhum gráficos podem ser gerados com os dados obtidos. Contudo o caso de teste gerado aborda todas as funções implementadas e pode se constatar que todas funcionaram nesse ambiente de testes.

7 Considerações Finais

O sistema Archon, produto deste trabalho de conclusão de curso foi modelado e desenvolvido, visando cumprir todos os objetivos previstos e no fim do cronograma previsto ele satisfaz os objetivos.

Um aplicativo Android foi modelado e desenvolvido, ele gerencia os pedidos do entregador e envia os dados de geolocalização do aparelho para o Web Service. O serviço está disponível para receber os dados enviados pela aplicação. Além disso ele oferece outros serviços para o aplicativo, fornecendo dados importantes para ela.

O site responsivo foi desenvolvido com o *template* AdminLTE e com os componentes Primefaces, contudo, utilizar esses dois causa conflito de CSS e causa certa dificuldade para a organização dos elementos. Muitos dos componentes do Primefaces nem podem ser utilizados por causa desse conflito. Como o conceito dessas 2 tecnologias é facilitar o desenvolvimento e estabelecer um padrão visual, e a utilização das 2 juntas resultam em conflito, conclui-se que se for observar apenas o conceito delas não faz muito sentido utilizar as 2 em conjunto. O sistema Web desenvolvido oferece, através do site desenvolvido, todas as funções esperadas para os Administradores e Destinatários.

O *back-end* do sistema Web desenvolvido oferece, através do site, todas as funções esperadas para os Administradores e Destinatários. O aplicativo oferece as funções do Entregador. Então pode-se concluir que este trabalho de conclusão de curso concluiu seus resultados esperados.

Mesmo com todas essas implementações esse sistema ainda se encontra na fase de protótipo, contudo todas as funções estabelecidas foram cumpridas sendo assim o trabalho de construção do protótipo está concluído e assim pode-se concluir que este trabalho também obteve sucesso.

7.1 Trabalhos Futuros

O sistema apresentado ainda é um protótipo, sendo assim, a modelagem e implementação de outras classes e métodos é o ponto chave para que ele se torne um produto final. Outro ponto para melhoria do sistema é a otimização de funções, principalmente do Aplicativo Android, onde otimizações como a de consumo de bateria podem agregar mais valor ao sistema como um todo.

No sistema em geral a principal adição a ser feita é a hospedagem do sistema na internet. Até então, os testes realizados foram em redes locais e por consequência não foram

realizados a longas distâncias.

Adicionar mais tabelas no banco de dados do aplicativo. Isso abriria mais espaços para a criação de mais funções para a aplicação, e assim incrementar mais a utilização do aplicativo para o Entregadores, tornando ele menos dependente de outras aplicações e dispensando o uso de relatórios impressos com as informações dos pedidos, ou utilizando o próprio aplicativo para realizar a impressão.

Adicionar a função de gerenciar as preferências do aplicativo. Assim o Entregador pode alterar algumas as preferências de algumas funções, como por exemplo, o intervalo de tempo que os dados de localização dos pedidos é enviado.

Pesquisar métodos de adicionar mais opções no mapa que o entregador possui. Opções como, exibir o trajeto da entrega; encontrar o menor caminho para a entrega do pedido; otimizar o trajeto da entrega caso mais de um pedido esteja sendo entregue numa mesma viagem.

Estudar uma forma de minimizar o envio de dados, o sistema atualmente recebe muitos dados geográficos num curto período de tempo, quando o número de usuários estiver grande essa grande quantidade de dados poderá fazer com que o sistema perca desempenho. Logo, procurar uma forma de reduzir esse envio de dados feitos pelos entregadores sem perder os dados importantes será quase que crucial quando o sistema crescer.

Uma forma de gerar relatórios gerenciais. Além das funções dadas aos usuários do tipo destinatário esse sistema também tem que oferecer vantagens para os administradores das empresas. Logo, adicionar funções para gerar relatórios iriam ajudar a parte gerencial da empresa. Exemplos de relatórios que poderiam ser gerados: quantidade de tempo gasto pelos entregadores para realizar entregas, tempo gasto para iniciar a entrega logo após o pedido ser feito, quantidade de entregas feitas por dia pelos entregadores.

Uma última adição seria criar métodos desse sistema importar os dados de outros sistemas já existentes, como os pedidos e clientes já cadastrados. Assim este sistema poderia ser utilizado em conjunto com sistemas já existentes.

Referências

ANDROID. *Desenvolvedores Android*. 2017. Disponível em: <<https://developer.android.com/index.html>>. Acesso em: 12 de maio 2017. Citado 2 vezes nas páginas 26 e 27.

BOND, M. et al. *Aprenda J2EE: com ejb, jsp, servlets, jndi, jdbc e xml*. [S.l.]: São Paulo: Pearson Education, 962p, 2003. Citado na página 22.

CAELUM. *Apostilas Caelum. O que é Java - Java e Orientação a Objetos*. 2017. Disponível em: <<https://www.caelum.com.br/apostila-java-orientacao-objetos/o-que-e-java>>. Acesso em: 12 de maio 2017. Citado na página 21.

COMSCORE, I. . *IMS Mobile in LatAm Study*. 2017. Disponível em: <<http://insights.imsincorporate.com/mobile2016/pt/#bottom2>>. Acesso em: 17 de junho 2017. Citado na página 13.

DEITEL, H. M.; DEITEL, P. J. *Java, como programar*. 4ª edição. São Paulo, 2003. Citado 2 vezes nas páginas 21 e 22.

EBIT. *WebShoppers - O mais completo relatório sobre o e-commerce*. 2017. Disponível em: <<http://www.ebit.com.br/webshoppers>>. Acesso em: 12 de maio 2017. Citado na página 13.

FARIA, L. H. L. et al. A aplicabilidade do modelo estendido ao consumo da teoria unificada da aceitação e uso de tecnologia (utaut2) no brasil: Uma avaliação do modelo a partir de usuários de internet em smartphone. *Revista de Administração da Universidade Federal de Santa Maria*, Universidade Federal de Santa Maria, v. 7, n. 2, p. 332–348, 2014. Citado na página 13.

HENRIQUE, F. *Construindo uma API RESTful em Java*. 2017. Disponível em: <<http://www.devmedia.com.br/construindo-uma-api-restful-em-java/29904>>. Acesso em: 17 de junho 2017. Citado na página 29.

KSOAP. *ksoap2-android - lightweight, efficient SOAP on Android*. 2017. Disponível em: <<http://simpligility.github.io/ksoap2-android/index.html>>. Acesso em: 12 de maio 2017. Citado na página 56.

MATHEWS, L. *This Is How Google Keeps 1.4B Android Devices Safe*. 2017. Disponível em: <<https://www.forbes.com/sites/leemathews/2017/02/20/this-is-how-google-keeps-1-6-billion-android-devices-safe/#5f5c93243137>>. Acesso em: 12 de maio 2017. Citado na página 25.

ORACLE. *Introduction to Javasever Faces - What is JSF?* 2017. Disponível em: <<http://www.oracle.com/technetwork/topics/index-090910.html>>. Acesso em: 12 de maio 2017. Citado na página 23.

PRIMEFACES. *Why PrimeFaces*. 2017. Disponível em: <<https://www.primefaces.org/whyprimefaces/>>. Acesso em: 12 de maio 2017. Citado na página 23.

- REDDY, K. S. P. *PrimeFaces Beginner's Guide*. [S.l.]: Packt Publishing Ltd, 2013. Citado na página 23.
- ROHERS, P. *Inovação gera vantagem competitiva*. 2017. Disponível em: <<http://www.administradores.com.br/artigos/negocios/inovacao-gera-vantagem-competitiva/85494/>>. Acesso em: 17 de junho 2017. Citado na página 13.
- ROZLOG, M. *REST e SOAP: Usar um dos dois ou ambos?* 2017. Disponível em: <<https://www.infoq.com/br/articles/rest-soap-when-to-use-each>>. Acesso em: 17 de junho 2017. Citado na página 30.
- SEBESTA, R. W. *Conceitos de linguagens de programação*. [S.l.]: Bookman Editora, 2009. Citado na página 21.
- SILVA, L. A. d. *Apostila de Android: Programando passo a passo*. [S.l.]: Ed, 2010. Citado 2 vezes nas páginas 24 e 25.
- SOMMERVILLE, I. *Engenharia de software-8ª edição*. Ed Person Education, 2007. Citado na página 37.
- SPAGNOLI, L. Um estudo sobre o desenvolvimento baseado em componentes. *PucRS - Brasil*, 2003. Citado 2 vezes nas páginas 30 e 31.
- TECNOVIA. *Como medir (e melhorar) a performance das transportadoras*. 2017. Disponível em: <<http://www.tecnovia.com.br/2017/05/05/monitorar-prazos-e-medir-a-performance-das-transportadoras-p1/>>. Acesso em: 17 de junho 2017. Citado na página 13.
- TUTORIALSPPOINT. *Android - Application Development*. 2017. Disponível em: <https://www.tutorialspoint.com/android/android_tutorial.pdf>. Acesso em: 12 de maio 2017. Citado na página 26.
- WEBMOBILE. *Artigo Webmobile 1 - Editorial*. 2017. Disponível em: <<http://www.devmedia.com.br/artigo-webmobile-1-editorial/2888>>. Acesso em: 12 de maio 2017. Citado 2 vezes nas páginas 28 e 31.
- WEBMOBILE. *Introdução às tecnologias Web Services: Soa, soap, wsdl e uddi - parte 2*. 2017. Disponível em: <<http://www.devmedia.com.br/introducao-as-tecnologias-web-services-soa-soap-wsdl-e-uddi-parte-2/2925>>. Acesso em: 12 de maio 2017. Citado na página 33.