UNIVERSIDADE TECNOLÓGICA FEDERAL DO PARANÁ
DEPARTAMENTO ACADÊMICO DE INFORMÁTICA
CURSO DE ESPECIALIZAÇÃO EM DESENVOLVIMENTO PARA DISPOSITIVOS MÓVEIS

RODRIGO ALESSANDRO SANTOS

APLICATIVO FRETE FÁCIL

CURITIBA
2017

APLICATIVO FRETE FÁCIL

Monografia de Especialização apresentada ao Departamento Acadêmico de Informática, da Universidade Tecnológica Federal do Paraná como requisito parcial para obtenção do título de Especialista em Desenvolvimento para Dispositivos Móveis.

Orientador: Prof. Robson R. Linhares

CURITIBA
2017

Ministério da Educação
UNIVERSIDADE TECNOLÓGICA FEDERAL DO PARANÁ
Câmpus Curitiba
Diretoria de Pesquisa e Pós-Graduação
Departamento Acadêmico de Informática
Coordenação do Curso de Especialização em Desenvolvimento para Dispositivos Móveis

TERMO DE APROVAÇÃO

“Aplicativo Frete Fácil”

por

“Rodrigo Alessandro Santos”

Este Trabalho de Conclusão de Curso foi apresentado às 17:33 do dia 18 de dezembro de 2017 na sala B201 como requisito parcial à obtenção do grau de Especialista em Desenvolvimento para Dispositivos Móveis na Universidade Tecnológica Federal do Paraná - UTFPR - Campus Curitiba. O(a) aluno(a) foi arguido pela Banca de Avaliação abaixo assinados. Após deliberação, a Banca de Avaliação considerou o trabalho aprovado.

_____ Prof. Robson Ribeiro Linhares **(Presidente/Orientador - UTFPR/Curitiba)**	_____ Profa. Maria Claudia Figueiredo Pereira Emer **(Avaliador 1 – UTFPR/Curitiba)**
_____ Prof. Adriano Francisco Ronszcka **(Avaliador 2 – Externo)**	

“A Ata de Aprovação assinada encontra-se na Coordenação do Curso.”

Dedicatória

A minha esposa Viviane, pela paciência e incentivo que tornaram possível a conclusão desta monografia.

Agradeço aos meus pais pelo apoio e incentivo a continuar meus estudos.

Agradecimentos

Agradeço aos meus pais, amigos e a minha esposa pela paciência e apoio, aos professores do curso de especialização em Desenvolvimento para Dispositivos Móveis pelos ensinamentos, e ao professor Robson por sua orientação, apoio e atenção, que possibilitou o desenvolvimento desse projeto.

Resumo

Santos, Rodrigo A., Aplicativo Frete Fácil. 2017. Trabalho de Conclusão de Curso de Especialização (Desenvolvimento para Dispositivos Móveis) - Departamento Acadêmico de Informática, Universidade Tecnológica Federal do Paraná. Curitiba, 2017.

Este trabalho visa atender o mercado de fretes de mudança, o qual ainda é carente de uma ferramenta que conecte pessoas com interesse em fazer uma mudança ao fretista. Por causa disso foi desenvolvido o aplicativo 'Frete Fácil', ele visa melhorar essa comunicação de forma rápida e simples, trazendo mais agilidade nos processos e ganhos financeiros. Para desenvolver o projeto foi realizado um levantamento de requisitos, análise dos aplicativos dos concorrentes, testes e estudo de tecnologias. O resultado final foi um aplicativo que cumpriu seu objetivo de conectar o usuário ao fretista, mas as avaliações dos usuários mostraram que a sua interface precisa ser retrabalhada antes ser disponibilizado para o mercado.

Palavras chaves: Frete. Android. Dispositivo Móvel. *Smartphone.*

Abstract

Santos, Rodrigo A., Freight Easy application - Specialization in Development for Mobile Devices, Universidade Tecnológica Federal do Paraná. Curitiba 2017.

This work aims to meet the freight market for change, which is still lacking a tool that connects people with an interest in making a change to the freight. Because of this the ‘Frete Fácil' application was developed, it aims to improve this communication quickly and simply, bringing more agility in the processes and financial gains. To develop the project, a survey of requirements, competitor application analysis, testing and technology studies was carried out. The end result was an application that fulfilled its goal of connecting the user to the freighter, but user ratings showed that its interface needs to be reworked before being made available to the market.

Keywords: Freight. Android. Mobile device. Smartphone

Lista de Figuras

Sumário

1. Introdução

Nesse capítulo será descrita a motivação da escolha de um aplicativo para frete de mudança, quais são seus principais objetivos e a metodologia utilizada no seu desenvolvimento.

1.1. Contexto

Vivemos uma época na qual as pessoas e serviços estão cada vez mais interligados pelo advento da internet e dos *smartphones*. A facilidade do uso dos aparelhos fez a sua popularidade aumentar muito nos últimos anos no Brasil. Muitas empresas ao redor do mundo estão vendo esse mercado como uma nova forma de fazer negócio, as pessoas querem estar cada vez mais conectadas às outras e estão buscando ferramentas para isso, um dos exemplos mais conhecidos é o próprio UBER que desenvolveu um aplicativo para *smartphone* com intuito de prover uma ferramenta eficaz de transporte de passageiros em carros particulares nas cidades (UBER, 2017).

Seguindo essa nova tendência de interação entre as pessoas, identificou-se que o mercado de fretes para o pequeno transportador ou autônomo ainda está carente de uma ferramenta que possa auxiliá-los no seu trabalho, aumentando seus ganhos.

Analisando este mercado, observa-se além do transportador o usuário com interesse em fazer uma mudança ou levar alguma pequena carga, mas não sabe como fazer isso. Nesse contexto é comum as pessoas procurarem perto de sua localidade ou por intermédio de conhecidos algum prestador de serviço da sua região, o problema dessa forma de busca se dá por alguns motivos: abrangência, tempo e custo.

- No caso da abrangência a pessoa fica limitada à sua rede de conhecimento ou pela busca perto de sua localidade, não tendo uma visão mais ampla das opções disponíveis na sua região;
- O tempo gasto para conseguir um frete pode ser grande, pelo fato de ter que entrar em contato com seus conhecidos ou mesmo tendo que se locomover pela cidade a procura de algum prestador;

- O custo gasto para realizar o frete pode não ser o mais barato pelo fato que a procura feita pelo usuário pode não ter sido abrangente;

Por esses motivos a utilização de um aplicativo pelo usuário facilitaria a busca por um frete, possibilitando visualizar uma gama maior de opções por um baixo custo.

Agora para os prestadores de serviço de mudança é freqüente ficar parado em uma ou mais localidades da região à espera de um frete. Essa forma de trabalho acaba não sendo a mais eficiente, pelos seguintes motivos: tempo, visibilidade das ofertas e financeiro.

- O tempo gasto pelo prestador parado à espera de um frete não está sendo remunerado, além de estar fixo em um local ele não consegue entrar em contato com os clientes para negociar frete porque não tem algum meio de comunicação que faça isso.
- Sobre a visibilidade das ofertas de mercado o prestador fica limitado à sua localidade ou rede de conhecimento, não tendo acesso a todo o mercado que está girando a sua volta.
- Olhando a parte financeira a opção de ficar parado a espera de um frete não é a mais interessante, porque o prestador nesse período poderia estar fechando e negociando novos fretes e assim aumentando a sua renda.

A principal vantagem de uma aplicação móvel é interligar esse dois mundos de forma simples, rápida e intuitiva, no qual ambos podem sair ganhando. Um dos principais ganhos tanto para o usuário e prestador é ter uma visão macro de sua região, podendo ver a sua volta todas as opções de negócio disponíveis sem precisar sair da sua casa ou do seu local de trabalho.

1.2. Objetivo

O objetivo deste trabalho é desenvolver um protótipo de aplicativo de frete para *smartphone*.

Os objetivos específicos deste trabalho são:

- Levantar requisitos por meio de entrevistas com usuários, e gerar uma versão beta do projeto para realização dos testes;

- Desenvolver o design do aplicativo;
- Desenvolver do WebService;
- Desenvolver um protótipo funcional;
- Criar questionário de avaliação do aplicativo e aplicar este questionário com voluntários.

1.3. Motivação/ Justificativa

Os aplicativos de *smartphone* como ferramenta de trabalho vem sendo utilizados em várias das áreas de negócios, o principal motivo é o fato da maioria das pessoas possuírem um *smartphone* e a internet móvel ter crescido 70% nos últimos anos, representando 62,8% dos domicílios com internet, pela primeira vez o uso do celular para acessar a internet ultrapassou o uso do computador, se tornando a principal ferramenta de acesso a internet (Correio do Povo, 2016).

O fato de o *smartphone* trazer mais recursos que um celular antigo, possibilitou a criação de aplicativos mais complexos que podem resolver problemas cotidianos antes impensáveis.

O mercado atual está cada vez mais dinâmico, tornando a necessidade de se manter atualizado a todo instante, a utilização de ferramentas que conectam as pessoas ou ajudam a aumentar a produtividade está em alta, é nesse pensamento que nasceu a ideia de fazer um aplicativo de fretes de mudança ou cargas pequenas, para conectar os usuários aos prestadores de serviço da sua região.

A vantagem de o usuário utilizar o aplicativo é ter uma visão completa das possibilidades de frete na palma da sua mão, sem precisar se deslocar pela cidade. Além disso, é possível ter acesso ao histórico do prestador de serviço podendo tomar decisões não somente com base no preço, mas também por pontualidade, educação e etc. O principal diferencial do aplicativo está na opção do usuário fazer um leilão do seu frete, e a melhor oferta leva a mudança.

Para o prestador de serviço o aplicativo se torna uma ferramenta de trabalho, aumentando suas chances de conseguir um novo frete e

conseqüentemente aumentando sua produtividade e ganhos, sem ter maiores gastos para isso, além de verificar o histórico do cliente, analisar alguns requisitos do cliente como bom pagador, recomendável e pontual.

1.4. Escopo

O objetivo deste trabalho de monografia é criar um aplicativo de fretes para *smartphone* que utilizam a plataforma Android. Nesse sistema tanto o usuário quanto o prestador de serviço deverão cadastrar seus dados para ter acesso à plataforma. Esses dados ficarão armazenados em um banco de dados local e remoto. Todas as informações provenientes dos cadastros estarão disponíveis por meio do *webservice* no servidor.

Para conectar os aplicativos será utilizado um servidor, todos os dados ficarão centralizados nele, e por causa disso o servidor ficará responsável por enviar e receber as solicitações, armazenar as principais informações em um banco de dados local, gerenciar as operações realizadas pelos usuários e limitar o acesso de cada grupo.

No aplicativo o usuário poderá visualizar todos os fretes disponíveis como seus dados e avaliação, também poderá se comunicar com o prestador de serviço e avaliá-lo. Já para o prestador de serviço, serão listados todos os fretes disponíveis e um canal de comunicação e avaliação do usuário.

Não serão contemplados nesse trabalho os seguintes itens:

- Desenvolvimento do WebSite para monitoramento e gerenciamento das informações contidas no servidor;
- Um sistema de segurança sofisticado como: criptografia de dados e de comunicação, validação de senha e etc;
- Design responsivo no aplicativo;
- A cobrança de valores monetários;
- O rastreamento em tempo real do veículo.

1.5. Metodologia

O trabalho apresentado será desenvolvido em algumas etapas:

- A primeira etapa consiste em fazer um estudo das tecnologias que serão utilizadas no aplicativo, para isso serão coletadas informações em sites confiáveis como: empresa fabricante do produto e sites de fonte confiáveis;
- A segunda etapa consiste no desenvolvimento e concepção do aplicativo, como análise de requisitos, casos de uso e testes;
- A terceira etapa consiste na avaliação do protótipo, no qual ele será avaliado sobre sua usabilidade. Para avaliar o aplicativo será criado um formulário com várias perguntas sobre o aplicativo no *GoogleDocs*.

1.6. Organização do Trabalho

Este trabalho está dividido em 6 capítulos:

O capítulo 1 descreve a motivação do trabalho, assim como sua justificativa, objetivo e metodologia utilizada.

O capítulo 2 descreve as tecnologias empregadas neste trabalho, de modo a tornar mais claro o motivo pelo qual foi escolhido para esse projeto.

O capitulo 3 aborda o processo de desenvolvimento do aplicativo, como seu estudo de casos e diagramas, assim como seu resultado final.

O capitulo 4 aborda as avaliações dos usuários e suas primeiras impressões ao utilizarem o aplicativo, assim como suas sugestões de melhorias.

O capitulo 5 contém as conclusões tiradas a partir do desenvolvimento do aplicativo. Nele consta uma avaliação sobre os pontos positivos e negativos do projeto, e sobre os objetivos que foram ou não alcançados.

No capitulo 6 são enumeradas todas as referências consultadas para realização deste trabalho.

2. Estudo das tecnologias

Nesse capitulo será descrito um estudo das tecnologias empregadas no projeto.

2.1. WebService

Nessa seção será descrito o que é um *WebService* e como é dividida sua arquitetura.

2.1.1. Conceito

O *WebService* é uma solução que permite a comunicação entre aplicações diferentes de uma maneira independente do ambiente de execução e de linguagem de programação (DevMedia, 2017). É uma forma de expor dados armazenados sem que seja necessário o fornecimento do servidor ou banco de dados. São componentes que permitem às aplicações enviar e receber dados em formato *XML*, utilizando alguns protocolos como *SOAP* e *REST*, conforme pode ser visto na Figura 1.

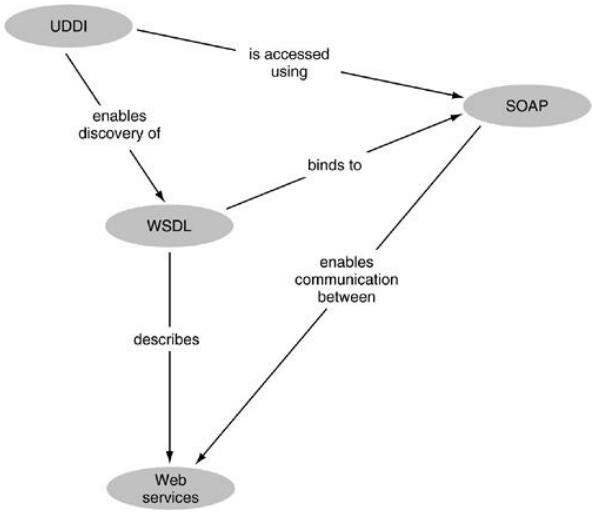


Figura 1 - Comunicação no WebService. Fonte (DevMedia, 2017)

Os *webservices* são amplamente utilizados em sistemas distribuídos pelo fato ter alta interoperabilidade, fraco acoplamento e utilizar o protocolo *http* para atuar como transporte entre o cliente e o *webservice*.

Para gerenciar a padronização de Serviços Web existem duas organizações responsáveis: *World Wide Web Consortium* (W3C) e a *Organization for the Advancement of Structured Information Standards* (OASIS), elas são grandes empresas do setor de tecnologia como a *Microsoft* (Microsoft, 2017).

2.1.2. Arquitetura

Os *webservices* são baseados numa arquitetura de três entidades: provedor de serviços, cliente do serviço e servidor de registro.

- Provedor de serviço é a entidade que cria o *webservice* e disponibiliza o serviço para alguma aplicação possa utilizar.
- Cliente do serviço é a aplicação que irá consumir os recursos do *webservice* chamando seus métodos.
- Servidor de registro é a localização central no qual o provedor de serviços pode relacionar seus *webservices* e no qual um consumidor de serviços pode pesquisá-los.

2.1.3. Protocolos

Nessa seção será descrito os principais protocolos de comunicação com *WebService,* e suas principais características.

2.1.3.1. Soap

SOAP – *Simple Object Access* (**Protocolo simples de acesso a objetos**), esse protocolo se encontra na versão 1.2 e é utilizado para troca de informações estruturadas entre sistemas e *webservices* que permitem a comunicação de mensagens entre aplicações via HTTP. (Zarelli, 2012)

O protocolo SOAP não está vinculado a qualquer tipo de linguagem de programação, sistema operacional ou transporte e é amplamente utilizado pelas grandes empresas. A mensagem SOAP é um documento XML com várias camadas como pode ser visto na Figura 2:

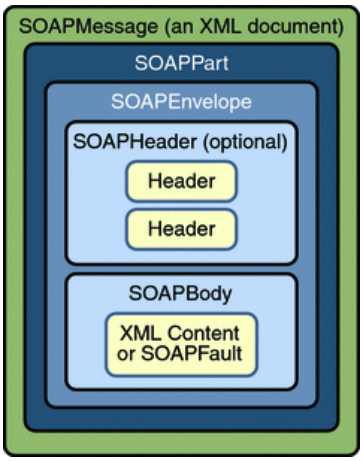


Figura 2 - Mensagem SOAP. Fonte (Zarelli, 2012)

O envelope é composto pelas seguintes partes:

- “Envelope” é um elemento raiz da mensagem SOAP, ele é responsável por definir o documento XML como uma mensagem SOAP.
- “Header” elemento responsável por informações específicas da mensagem, porém o seu uso não é obrigatório.
- “Body” contém a mensagem enviada por outra aplicação.
- “Fault” é um elemento que contém os erros que podem ocorrer.

A codificação é um conjunto de regras internas para codificar o tipo de dados.

Os tipos de dados SOAP são divididos em duas grandes categorias: tipos escalares e tipos compostos.

- Os tipos escalares contêm exatamente um valor, como um sobrenome, preço ou descrição do produto;
- Os tipos de compostos contêm vários valores, como um pedido de compra ou uma lista de cotações de ações;
- Os tipos de compostos são subdivididos em arrays e estruturas.
- O estilo de codificação para uma mensagem SOAP é definido por meio do atributo *SOAP-ENV: encodingStyle*.

As principais vantagens:

- Mapeia satisfatoriamente para o padrão de solicitação/resposta HTTP;
- Pode ser usado tanto de forma anônima como com autenticação (nome/senha);

As principais desvantagens:

- Problema de interoperabilidade entre implementações diferentes de SOAP;
- Não implementa criptografia;
- Não existe garantia de entrega;

2.1.3.2. Rest

Rest - *Representational State Transfer* (**Transferência de Estado Representacional**) é uma abstração da arquitetura da World Wide Web (*Web*), um estilo arquitetural que consiste de um conjunto coordenado de restrições arquiteturais aplicadas a componentes, conectores e elementos de dados dentro de um sistema de hipermídia distribuído (InternetDasCoisas, 2015). Esse estilo de arquitetura faz uso dos recursos oferecidos pelo HTTP como *GET, POST, PUT* e *DELETE*, como pode ser visto na Figura 3.

- GET: Recupera as informações sobre o recurso URI, esse comando não solicita qualquer alteração de estado do sistema;
- POST: Cria um novo recurso
- PUT: É utilizada quando desejamos criar ou atualizar uma informação;
- DELETE: Solicita ao servidor para excluir uma entidade específica do servidor.

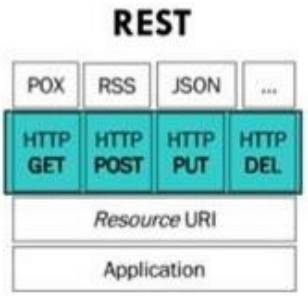


Figura 3 - Arquitetura REST. Fonte (Stackoverflow,2017)

As principais vantagens:

- Protocolos menos complexos;
- Mais flexibilidades nas comunicações;
- Arquitetura utilizada na maioria das empresas;
- Menos overhead no protocolo.

As principais desvantagens:

- Integrações com produtos fechados *WebService*.

2.1.3.3. XML

XML - *eXtensible Markup Language*, é um formato simples baseado em texto para representar informações estruturadas: documentos, dados, configuração, livros, transações, faturas e muito mais. Foi derivado de um formato padrão antigo chamado SGML (ISO 8879), para ser mais adequado para uso da Web (W3C, 2018).

Os elementos de marcação provêem um formato para descrever dados estruturados. Esses formatos servem para organizar e formatar um site em HTML ou padronizar um arquivo de dados para que outra aplicação possa interpretá-lo no caso do XML, conforme Figura 4.

```
<?xml version="1.0"?>
<menu>
   <parent title="kirupaPicks">
    <child>
        <title>kirupa.com</title>
        <link>http://www.kirupa.com</link>
    </child>
    <child>
        <title>kirupaForum</title>
        <link>http://www.kirupa.com/forum/</link>
    </child>
    <child>
        <title>kirupa's Blog</title>
        <link>http://blog.kirupa.com</link>
    </child>
   </parent>
</menu>
```

Figura 4 - Exemplo de código XML. Fonte (Kirupa, 2017)

A linguagem XML foi projetada para prover troca de informações na internet independente de sistemas operacionais e aplicativos devendo ser legível para seres humanos, porém não focando em sua aparência e sim na estrutura da informação.

O XML não possui um conjunto pré-definido de tags ou elementos podendo ser definido livremente pela aplicação. Sua estrutura é baseada em alguns termos: tags, elementos e atributos.

As principais vantagens:

- Baseado em texto simples;
- Representa uma estrutura de dados: lista;
- Auto documentado: o próprio formato descreve a estrutura e os nomes dos campos.

As principais desvantagens:

- A sintaxe do XML é redundante ou torna-se grande em relação a representações de dados semelhantes;
- Redundância pode afetar a eficiência quando se utiliza o XML para armazenamento;
- Velocidade: a grande quantidade de informação repetida prejudicando a velocidade de transferência real de informação.

2.1.3.4. WSDL

WSDL– Web Services Language (**Linguagem de descrição de serviços web**), é um XML que descreve as operações e métodos fornecidos pelo *webservice*, descreve a localização de um serviço e como o serviço pode ser acessado (DevMedia WSDL, 2018).

2.1.3.5. UDDI

UDDI – Integração, descoberta e descrição universal (*Universal Description and Discovery Information*), é um padrão desenvolvido para fornecer um diretório de busca para os negócios e seus serviços (DevMedia UDDI, 2009).O seu papel é permitir que as requisições dos clientes encontrem o fornecedor do serviço adequado, conforme pode ser visto na Figura 5.

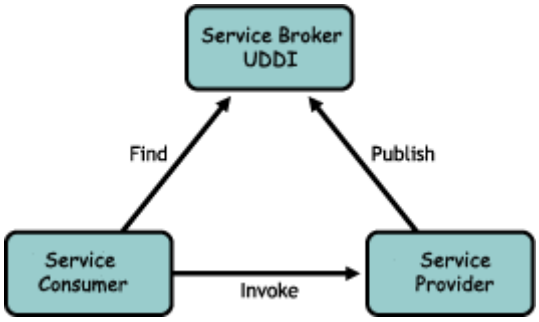


Figura 5 - Exemplo de serviço de registro. Fonte (Novell, 2017)

Esse protocolo é composto por duas funções:

- Protocolos baseados em SOAP que define como clientes UDDI se comunicam com registros;
- Um conjunto específico de registros replicados globalmente;

O serviço de registro UDDI gerencia informações de provedores, implementações e metadados. Os usuários podem requisitar dados por meio desses serviços em três partes: :

- Primeira: Descreve os dados da companhia como nome, endereço e contatos;
- Segunda: Inclui as categorias, baseada em taxonomias padrões;
- Terceira: Descreve a interface para o serviço em nível de detalhe suficiente para se escrever uma aplicação que use o *WebService*.

2.1.3.6. URI

Os webservices são identificados por meio de um URI (*Uniform Resource Identifier*), que é uma cadeia de caracteres compacta usada para identificar ou denominar um recurso na Internet como pode ser visto na Figura 6, bastante parecido com uma URL. O principal propósito desta identificação é permitir a interação com representações do recurso por meio de uma rede usando protocolos específicos (Portal Educação, 2012).

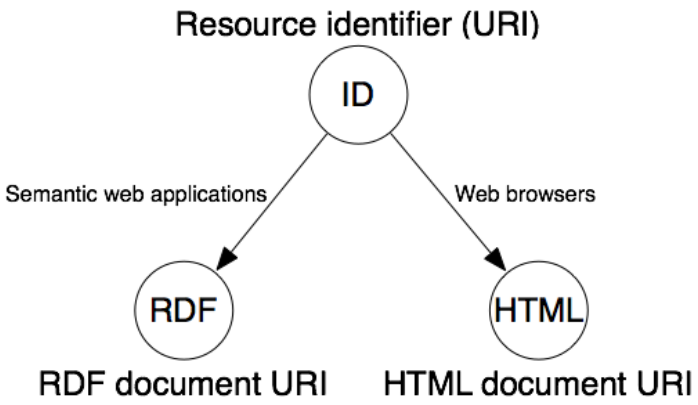


Figura 6 - Relação entre os recursos e documentos. Fonte (W3, 2008)

Toda interface possui métodos com anotações HTTP (GET, POST, DELETE e PUT) para especificar o tipo de solicitação.

2.2. Retrofit

Neste capítulo será descrito o que é o *Retrofit*, quais são seu adaptadores e como funciona sua autenticação.

2.2.1. Conceito

A *Retrofit* é uma API desenvolvida pela Square seguindo padrão REST, fornecendo um padrão simples de implementação para transmissão de dados entre aplicação e servidor, que faz uso do JSON. As tarefas executadas pelo Retrofit são realizadas de forma assíncrona.

Para utilizar a Retrofit é necessário implementar os seguintes itens: Classe modelo, interface HTTP e a classe Retrofit.Builder.

- Classe Modelo: Utilizado para mapear dos dados JSON;
- Interface HTTP: Define as possíveis operações HTTP;
- Retrofit.Builder: Utiliza a Interface a API Builder que permite definir o ponto final do URL para a operação HTTP;

2.2.2. Conversores e adaptadores

A Retrofit utiliza conversores para manipular serialização dos dados. Os principais conversores são:

- Para converter JSON: Gson, Jackson e Moshi;
- Para converter Buffers de protocolo: Protobuf e Wire;
- Para converter para XML: XML simples;

2.2.3. Adaptadores

É possível estender os recursos da Retrofit utilizando outros adaptadores como, por exemplo, *RxJava*, assim é possível fazer que o Retrofit retorne outros tipos de saída.

2.2.4. Autenticação

A Retrofit suporta chamadas de API que precisam de autenticação. A autenticação pode ser feita utilizando login e senha ou *token* de API.

Existem dois métodos para autenticar: O primeiro seria manipular o cabeçalho com ajuda de anotações e o segundo método seria utilizar o *OkHttp*.

- Autenticação com anotações

Para utilizar as anotações é necessário adicionar um campo "*Authorization*" no cabeçalho da solicitação com o valor da credencial. É possível utilizar a classe de *Credenciais OkHttps* para autenticação básica e para utilizar o *Token* como autenticação basta utilizar o método *getUserDetails* com o seu *token*.

- Autenticação com interceptores *OkHTTP*

Caso tenha várias chamadas que exigem que você autentique, você pode usar um interceptor para isso. Um interceptor é usado para modificar cada solicitação antes de ser executado e altera o cabeçalho da solicitação. A vantagem é que você não precisa adicionar *@Header("Authorization")* a cada definição de método de API.

2.3. SQLite

SQLite é uma biblioteca de código aberto que implementa um mecanismo de banco de dados SQL transacional autônomo, sem servidor e não possui um processo de servidor separado.

O SQLite é um banco de dados SQL completo com várias tabelas, índices, disparadores e visualizações está contido em um único arquivo de disco, essa biblioteca lê e grava diretamente do arquivo de banco de dados em disco.

SQLite é recomendado para aplicações que utilizam pequeno volume de dados como: aplicativo básicos para dispositivos móveis, pequenos sites. Por causa de seu desempenho não é recomendável utilizá-lo em aplicações de grande porte ou que recebem muitas requisições.

As principais características do SQLite são:

- Não necessita de instalação;
- Ferramentas de estatística e de análise;
- Software livre e multiplataforma;

2.4. Netbeans

O NetBeans IDE é um ambiente de desenvolvimento integrado e gratuito para desenvolvedores de várias linguagens como: JAVA, C, PHP. O IDE é executado em várias plataformas como Windows, Solaris, Linux e MacOS. Abaixo segue uma Figura 7 ilustrando a IDE.

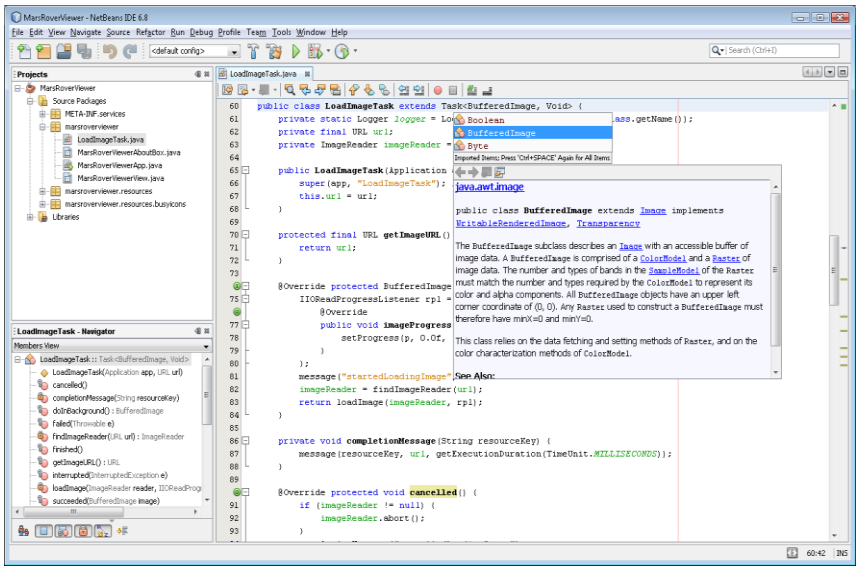


Figura 7 - Exemplo da IDE do Netbeans. Fonte (Netbeans, 2017)

O principal objetivo era criar um IDE com funcionalidades existente nas IDEs de mercado porém totalmente desenvolvida em Java. Os seus principais recursos:

- Suporte a Database;
- Suporte ao Java Enterprise Edition;
- Visualizador de classes;
- Editor de código e outros;

2.5. Android

Android é um sistema operacional baseado no núcleo Linux e atualmente é desenvolvido pela empresa de tecnologia Google, com uma interface baseada na manipulação direta. (Abril, 2012)

2.5.1. Arquitetura

A arquitetura do Android é baseada no Kernel Linux, ela é dividida nas seguintes camadas:

- Kernel do Linux;
- Camada de abstração de hardware (HAL);
- Android Runtime;

- Bibliotecas C/C++ nativas;
- Estrutura da Java API
- Aplicativos do sistema

2.5.2. Activity

A Activity é uma classe que serve de ponto de entrada para interação de um aplicativo com o usuário, fornecendo uma janela na qual o aplicativo desenha sua interface.

2.5.3. Ciclo de vida Activity

A Activity responde diretamente as invocações do framework do Android, por isso é fundamental entender o seu ciclo de vida, porque ela pode ser interrompida de forma inesperada.

Métodos:

- onCreate(): É um método obrigatório e é invocado apenas uma vez;
- onStart(): É chamada após o evento OnCreate() e também quando uma Activity volta do background e volta a ter foco. É um local indicado para se certificar de que todos os recursos requeridos continuam disponíveis;
- onResume(): Sempre invocada quando uma Activity recebe foco novamente;
- onPause(): Salva o estado da aplicação. É um local ideal para parar animações, salvar dados e liberar recursos do sistema;
- onStop(): Quando a Activity está sendo encerrada e não fica mais visível ao usuário. Local utilizado para liberar todos os recursos que não são utilizados pelo usuário;
- onDestroy(): Finaliza a aplicação, pode ser invocada também pelo sistema operacional caso deseje desalocar recursos;
- onRestart(): Quando uma Activity volta depois de ficar temporariamente e invoca o método onStart;

A Figura 8 ilustra de forma geral o funcionamento de uma atividade, vejamos todas as etapas a seguir.

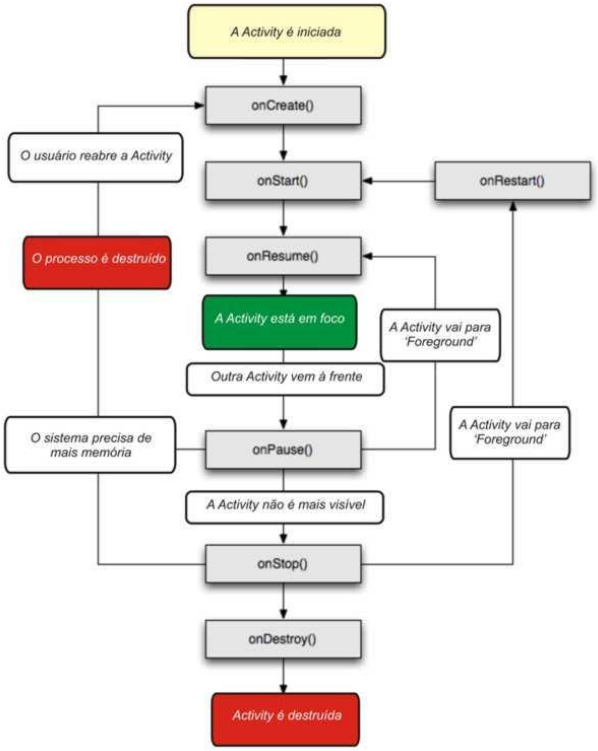


Figura 8 - Ciclo de vida de uma atividade. Fonte (TheClub, 2012)

2.5.4. Pilha Activities

Toda vez que uma aplicação é iniciada, o sistema operacional Android cria uma pilha de Activities no modelo LIFO (último que entra, primeiro que sai), na qual a primeira Activity já é colocada na pilha.

Quando uma nova Activity é iniciada, ela é colocada no topo da pilha como ativa e a anterior vai para segunda posição da fila. Quando a Activity ativa é finalizada, ela é retirada da pilha e a segunda se torna a principal. Esse processo pode ser visto mais detalhadamente na Figura 9.

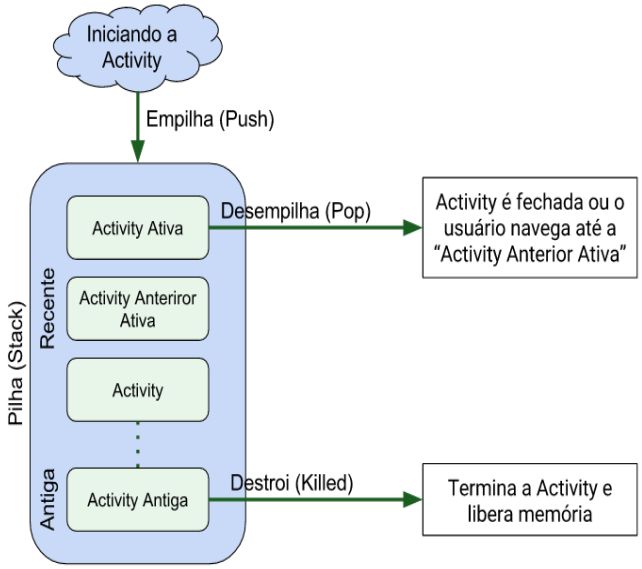


Figura 9 - Pilha de Activity. Fonte (AndroidPro, 2016)

2.5.5. Android Manifest

O arquivo AndroidManifest.xml como ilustrado na Figura 10 é responsável por informar ao sistema operacional Android todas as informações necessárias para o aplicativo funcionar antes que seja executado. É obrigatório que esse arquivo se encontre no diretório raiz do projeto.

```
1<?xml version="1.0" encoding="utf-8"?>
2<manifest xmlns:android="http://schemas.android.com/apk/res/android"
3      package="com.androidapp.basicElements"
4      android:versionCode="1"
5      android:versionName="1.0">
6    <application android:icon="@drawable/icon" android:label="@string/app_name">
7        <activity android:name=".BasicElements"
8                  android:label="@string/app_name">
9            <intent-filter>
10               <action android:name="android.intent.action.MAIN" />
11               <category android:name="android.intent.category.LAUNCHER" />
12           </intent-filter>
13       </activity>
14
15   </application>
16   <uses-sdk android:minSdkVersion="2" />
17
18</manifest>
```

Figura 10 - Exemplo do arquivo AndroidManifest. Fonte (127bytes, 2010)

Além disso, o AndroidManifest tem outras funções como:

- Adicionar o nome do pacote ao aplicativo e usar como identificador exclusivo;

- Configurar outros componentes do aplicativo como *Activities*, *Services*, *Content Prodivers* e etc;
- Declarar as permissões que o aplicativo deve ter para funcionar corretamente no dispositivo do usuário;
- Declarar o nível mínimo da Android API que o aplicativo exige para funcionar.

Estrutura do arquivo do manifesto

- Manifest: Deve conter o elemento <application> e especifica pacote e versão do aplicativo;
- Activity: Especifica uma Activity. Todas as Activities devem ser adicionadas no arquivo, porque se alguma não for declarada o sistema operacional não irá encontrá-la e nunca será executada;
- Service: Declara um serviço como um dos componentes do aplicativo. Eles são usados para implementar operações em segundo plano de fundo ou uma API de comunicação rica que pode ser chamada por outros aplicativos;
- Receiver: Especifica o BroadcastReceiver como um dos componentes do aplicativo. Existem duas maneiras de declarar o BroadcastReceiver: um é declarado no arquivo AndroidManifest com este elemento. O outro é criar o BroadcastReceiver dinamicamente no código e registrá-lo com o método Context.registerReceiver();
- Provider: Declara um componente provedor de conteúdo, Caso contrário, o sistema não tem conhecimento deles e não os executa;
- Uses-Permission: Serve para configurar as permissões que o aplicativo deve receber para funcionar. As permissões são concedidas quando o aplicativo é instalado ou enquanto está sendo executado.

2.6. Android Studio

O Android Studio é o ambiente de desenvolvimento integrado (IDE) oficial para o desenvolvimento de aplicativos Android como ilustrado na Figura 11 e é baseado no IntelliJ IDEA (Android Studio, 2017).

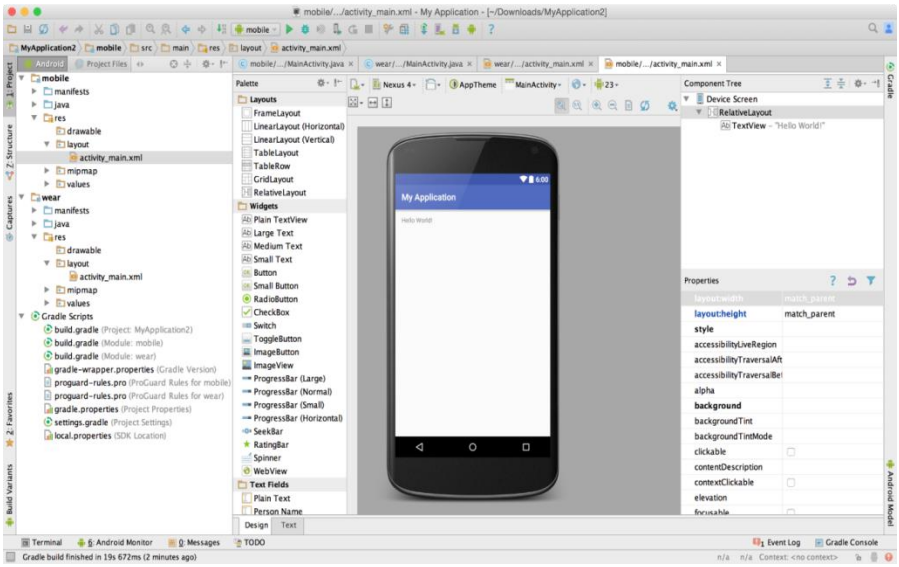


Figura 11 – IDE do Android Studio. Fonte (Android Criar Projeto, 2017)

Características:

- Suporte para compilações baseadas em Gradle;
- Refatoração específica para Android e reparações rápidas;
- Suporte nativo para a *Google Cloud Platform*, permitindo a integração com *Google Cloud Messaging* e *App Engine*.

2.7. Design de interação

Design de Interação é uma área que estuda a interação entre um sistema e o usuário, o objetivo do design de interação consiste em redirecionar essa preocupação, trazendo a usabilidade para dentro do processo de design. Essencialmente, isso significa desenvolver produtos interativos que sejam fáceis, agradáveis de utilizar e eficazes (Univast, 2017). O Design de Interação é composto por outras áreas como ilustrado na Figura 12.

Figura 12 – IDE do Android Studio. Fonte (Android Criar Projeto, 2017)

Algumas das principais metas do Design de Interação são:

- Usabilidade: Fazer que o usuário chegue ao seu objetivo de maneira fácil;
- Legibilidade: Compreensão dos itens que estão na tela, isso envolve texto ou qualquer outro símbolo que tenha significado;
- Estética: Se a aparência do sistema está de acordo com a proposta do projeto;

Segue alguns benefícios da aplicação de Design de interação:

- Adequar as respostas do sistema às entradas do usuário;
- Balancear interação e funcionalidade;
- Prevenir erros do usuário;

2.7.1. Design centrado no usuário (DCU)

É uma metodologia que visa colocar o usuário no foco, no qual suas necessidades, desejos e expectativas estejam em primeiro plano. Para isso os designers devem criar as interfaces e realizar testes com usuários reais.

O design centrado no usuário possui quatro etapas básicas:

- Identificar e definir requisitos: Levantar necessidades e entender os pontos de conflitos dos usuários por meio de pesquisas, observações e entrevistas;
- Criar soluções alternativas: Levantar hipóteses e ideias que podem vir a solucionar os problemas encontrados durante a identificação dos requisitos;
- Construir protótipos testáveis: tirar ideia do papel e criar protótipos testáveis;
- Avaliar com usuário: Testar o protótipo com usuários reais para descobrir o que se pode melhorar e validar as alternativas propostas, evitar desenvolvimento de funcionalidades inúteis e o excesso de informação;

2.7.2. UX - Experiência do usuário

Experiência do Usuário (*User Experience*) se dá na utilização do produto. Isso envolve o sentimento da pessoa enquanto está utilizando o sistema, essa percepção do usuário pode ser um fator de sucesso ou fracasso do produto. O principal objetivo é tornar as interfaces úteis para ajudar os usuários a cumprir suas metas.

Para coletar as informações dos usuários existem várias técnicas, seguem algumas delas:

- Personas: Persona é uma identidade fictícia que reflete um grupo de usuários.
- Prototipação em papel: Consiste em criar um esboço em papel, desenhos de interface para depois então usá-los em testes;
- Entrevistas: A entrevista é uma das técnicas tradicionais mais simples de utilizar e que produz bons resultados na fase inicial de obtenção de dados. Convém que o entrevistador dê espaço ao entrevistado para esclarecer as suas necessidades;
- Surveys: Construir uma pesquisa online, utilizando o GoogleDocs por exemplo, para obter o feedbacks de possíveis usuários.

2.7.3. Design Sprint

O *Design Sprint* é um *framework* de cinco fases que ajuda a responder questões críticas do negócio por meio de prototipagem rápida e testes de usuários como poder ser visto na Figura 13. Os *Sprints* permitem que sua equipe atinja suas metas em curto período de tempo. O processo incentiva o pensamento centrado no usuário, alinhando a visão da equipe, levando ao lançamento do produto, de forma mais rápida. (SaibaMais,2017)

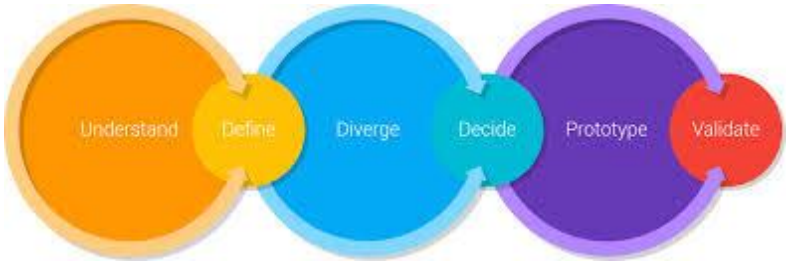


Figura 13 – Os passos do Design Sprint. Fonte (iMaster, 2015)

Desenvolvido pelo *Google Ventures*, o *Design Sprint* é um processo dividido nas seguintes etapas.

- Entender e definir: Buscar o melhor entendimento do problema proposto. Nessa fase é importante as pessoas compartilharem seus conhecimentos sobre aquele proposto;
- Divergir: Cada participante desenha sua ideia. O objetivo é explorar o máximo de idéias possíveis, não tendo limite e nem restrições;
- Decidir: É a fase de escolher uma ideia entre as mais votadas;
- Prototipar: Nessa fase o objetivo é criar um protótipo de médio-alta fidelidade porque será testado com usuários reais e um protótipo de baixa qualidade não traria o retorno desejado;
- Validade: O protótipo é levado para teste aos potenciais usuários do produto, nesse momento eles irão interagir e dar um *feedback* real sobre o produto. No final do dia alguns itens serão aprimorados e o que não deu certo será descartado;

2.7.4. Protótipos

O protótipo é uma versão preliminar de um sistema, que possibilita ilustrar os conceitos de funcionalidades e design, melhorando o entendimento do projeto.

Para criar um aplicativo é muito importante avaliar os recursos e interação, por isso podemos destacar os três principais:

- Baixa fidelidade: Visa definir de modo simples como seria a interação do usuário com a aplicação não tendo nenhuma preocupação com o design. Geralmente são feito a mão utilizando lápis e borracha, mas hoje já existem alguns softwares que auxiliam nesta atividade. Esse processo é utilizado na fase de definição do projeto e levantamento de requisitos;
- Média fidelidade: São protótipos muito utilizados na fase que envolve a arquitetura de informação. Nesse protótipo é possível navegar entre as telas, criar um layout básico, simulação simples de uso e etc.;
- Alta fidelidade: Possui uma representação gráfica bem próxima do que será o aplicativo final, na maioria dos casos é possível simular o fluxo das funcionalidades como fosse usuário final, mas sem gravar na base de dados.

2.7.5. UI – Interface de usuário

A Interface de usuário (UI – *User Interface*) é o meio pelo qual o usuário interage com uma determinada aplicação ou dispositivo. Essas interações são feitas por meios de botões, menus, imagens e outros, e levam o usuário a uma interação agradável e positiva.

Cada sistema operacional tem o seu próprio conjunto de recomendações, com objetivo que aplicativos utilizem melhor as funcionalidades do sistema operacional.

Principais práticas de um projeto de UI:

- Agrupar informações semelhantes;
- Deixar claro o que deve ser clicado e selecionado;
- Mostrar informações ao invés de escondê-las;
- Ser objetivo e direto;

2.8. Aplicativos do mercado

Nessa seção serão listados alguns dos principais aplicativos de mercado voltado ao mercado fretes. O objetivo é apresentar seus principais pontos positivos e negativos (PlayStore, 2017).

2.8.1. Quero Frete

É um aplicativo focado no carreteiro autônomo, para encontrar cargas perto da sua localidade como ilustrado na Figura 14.

Os principais pontos fortes:

- Rastreamento da carga;
- Possibilidade de negociar o frete via telefone ou por mensagem;
- O aplicativo grátis para os caminhoneiros;
- Layout intuitivo;

Os principais pontos fracos:

- Não tem muita atualização no aplicativo;
- Pouca variedade de transportadora;
- Problema no gerenciamento do cadastro. Ex: Durante o login o sistema informa que o CPF está cadastrado e recusa a conexão, sem nenhuma opção de contornar o problema;

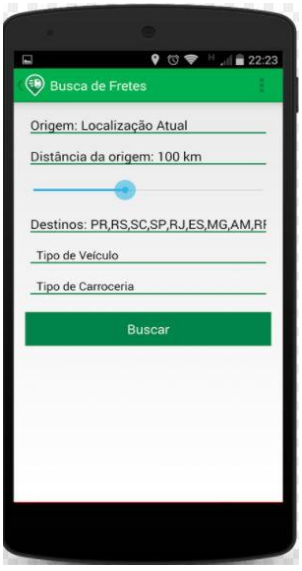


Figura 14 – Tela do aplicativo ‘Quero Frete’. Fonte (PlayStore, 2017)

2.8.2. FROTANET

É um aplicativo voltado ao caminhoneiro autônomo, ele permite a transportadora enviar ofertas de frete diretamente para os caminhoneiros, de forma fácil e instantânea, sem intermediação de terceiros como ilustrado na Figura 15.

Os principais pontos fortes:

- Indicadores de resultado, o caminhoneiro terá relatórios completos de sua viagem;
- Acompanhamento de viagens registra diversas ocorrências durante uma viagem como: início e fim do carregamento, ponto de abastecimentos, restaurantes e outros;

Os principais pontos fracos:

- Não tem fretes para pequenas mudanças;

Figura 15 – Tela do aplicativo ‘FROTANET. Fonte (PlayStore, 2017)

2.8.3. iMoving

Faz orçamentos de mudanças para seus usuários, compara preços de mudanças, tem *reviews* de transportadoras e dicas de segurança como ilustrado na Figura 16.

Os principais pontos fortes:

- Rastreamento da carga;
- Contabiliza o tempo de chegada da carga;

Os principais pontos fracos:

- Problema de login, mesmo com o cadastro o aplicativo não reconhece os dados de entrada;

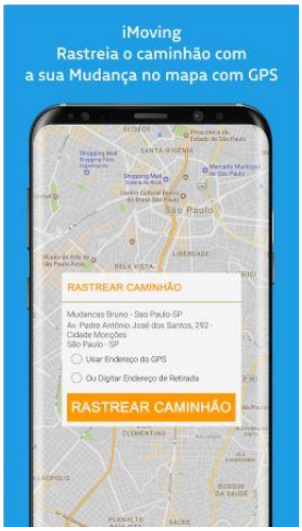


Figura 16 – Tela do aplicativo ‘iMoving’. Fonte (PlayStore, 2017)

2.8.4. Sontra

O aplicativo é focado em fazer fretes diretamente com transportadoras e embarcações no Brasil, no qual o fretista cadastra os seus dados e através de sua localização é listados os fretes disponíveis na região como ilustrado na Figura 17.

Os principais pontos fortes:

- A interface do aplicativo é intuitiva;

- É possível conversar com a transportadora via aplicativo;

Os principais pontos fracos:

- Não tem como simular o valor do frete;
- Não há muitas opções de fretes disponíveis;
- Não atende o mercado de mudança.

Figura 17 – Tela informativa do aplicativo ‘Sontra. Fonte (PlayStore, 2017)

3. Desenvolvimento do Aplicativo

Esse capítulo descreve os requisitos do sistema, projeto UML, projeto de interface e arquitetura do sistema.

3.1. Descrição do Problema

O aplicativo deverá cadastrar os dados dos usuários e fretistas, listar as mudanças disponíveis e possibilitar a comunicação entre eles.

3.2. Requisitos Funcionais

- O sistema deve permitir que o usuário cadastre seus dados e da mudança;
- O sistema deve permitir que o fretista cadastre seus dados e do seu veículo;
- O sistema poderá enviar e receber mensagens entre usuário e o fretista;
- O sistema deve listar todos os fretistas disponíveis;
- O sistema deve listar todas as mudanças disponíveis;
- O sistema deve registrar o histórico das conversas entre o usuário e fretista e sua avaliação;
- O sistema deve gerenciar o sistema de lance;
- O WebService registrará todas as mensagens enviadas no banco de dados.

3.3. Requisitos Não Funcionais

- Os dados cadastrais, mensagens e avaliações deverão ser registradas no banco de dados Postgre no Webservice;
- As conversar entre o usuário e o fretista deverão ser cadastrado no SQLite no aplicativo;
- O sistema deve retornar a reposta em até 5 segundos;
- O aplicativo deve ser rápido não levando mais de 5 segundos para executar uma ação;
- O sistema deverá ser autenticar no Webservice.
- O aplicativo não deve ocupar mais que 30MB.

3.4. Arquitetura do sistema

A arquitetura do sistema “Frete Fácil” consiste em um *Web Service* e aplicativo como ilustrado na Figura 18. O *Web Service* é responsável por gerenciar as solicitações e armazenar os dados dos usuários. O aplicativo é a ferramenta do usuário e do fretista para utilizar o sistema, por causa disso ele tem duas maneiras de trabalhar. O primeiro modo é do usuário, no qual ele poderá cadastrar os dados de sua mudança e escolher um frete na lista de opções disponíveis. No segundo modo, o fretista poderá cadastrar seus dados e do seu veiculo e escolher na lista de fretes disponíveis qual tem interesse em trabalhar. A comunicação do aplicativo com Web Service será feita por meio da internet.

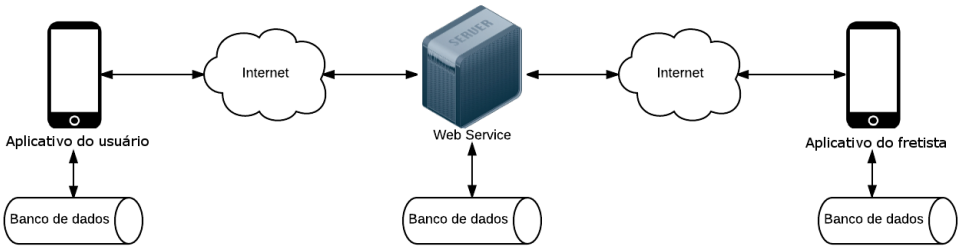


Figura 18 – Arquitetura do sistema Frete Fácil

3.5. Diagrama de entidade e relacionamento

Nesse capítulo será descrito os diagramas de entidade e relacionamento do *WebService* e *Smartphone*.

3.5.1. Banco de dados Web Service

O diagrama da Figura 19 ilustra as relações entre as tabelas do banco de dados *Web Service* e suas associações.

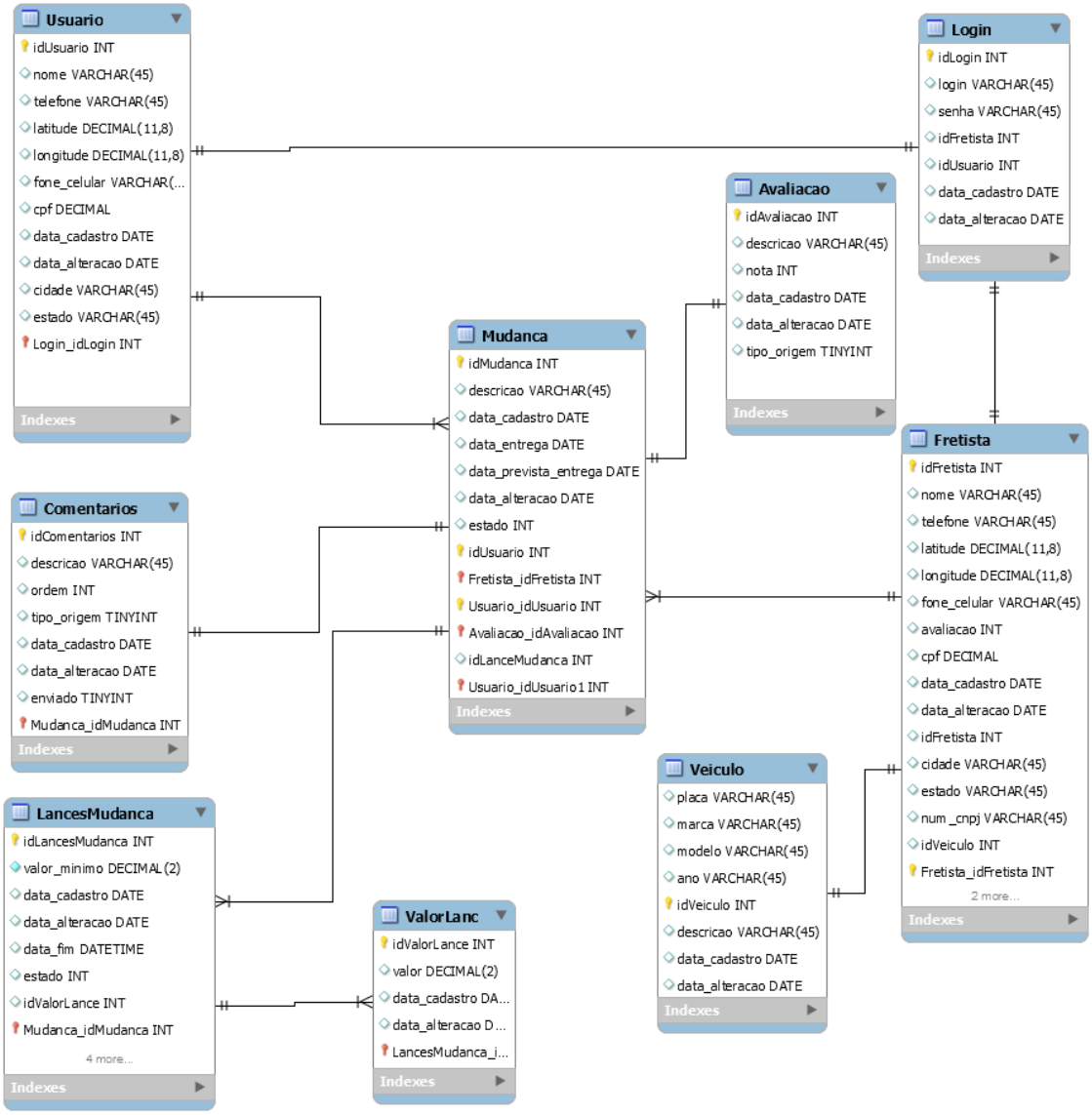


Figura 19 – Modelagem do banco de dados Web Service

3.5.2. Banco de dados Smartphone

O diagrama da Figura 20 ilustra as relações entre as tabelas do banco de dados do *smartphone* e suas associações

Figura 20 - Modelagem do banco de dados SmartPhone

3.6. Caso de uso

Nesta seção serão listadas as principais relações entre o usuário e o fretista com o sistema.

3.6.1. Cadastrar usuário

O diagrama da Figura 21 tem como objetivo mostrar o cadastro do usuário no sistema Frete Fácil.

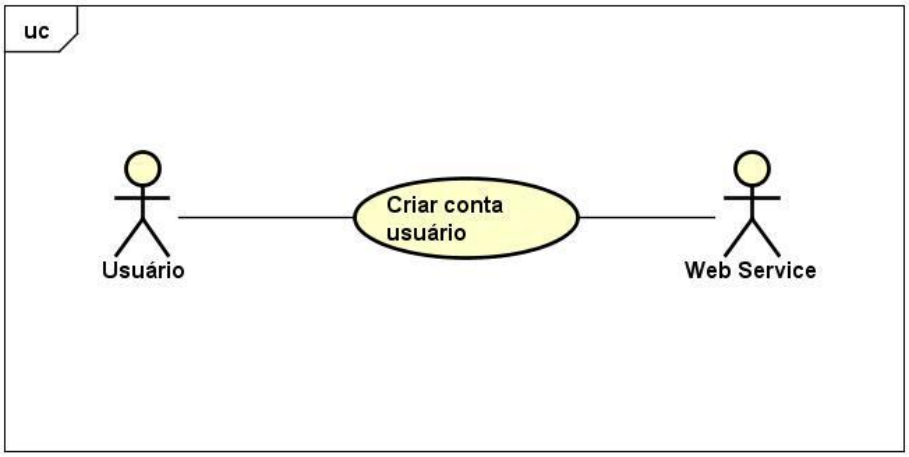


Figura 21 – Cadastro do usuário e fretista

Nome do Caso de Uso:

Cadastrar usuário.

Descrição:

Cadastrar os dados do usuário no servidor.

Atores:

- Usuário;
- WebService;

Requisitos:

O sistema deve permitir que o usuário cadastre seus dados e da mudança.

Fluxo principal:

1. Usuário inclui os dados
2. Sistema valida dados
2.1. Valida os dados cadastrados (A1)
3. Sistema armazena os dados
4. Informa se houve falha ou sucesso
5. Fim do caso de uso

Fluxo alternativo:

A1: em “Sistema valida dados”, se o Usuário fornecer dados errados ou incompletos
A1.1: Retorne uma mensagem de erro
A1.2: Retorne o fluxo básico para o item “Usuário inclui os dados”

3.6.2. Cadastrar mudança

O diagrama da Figura 22 tem como objetivo mostrar o cadastro da mudança no sistema Frete Fácil.

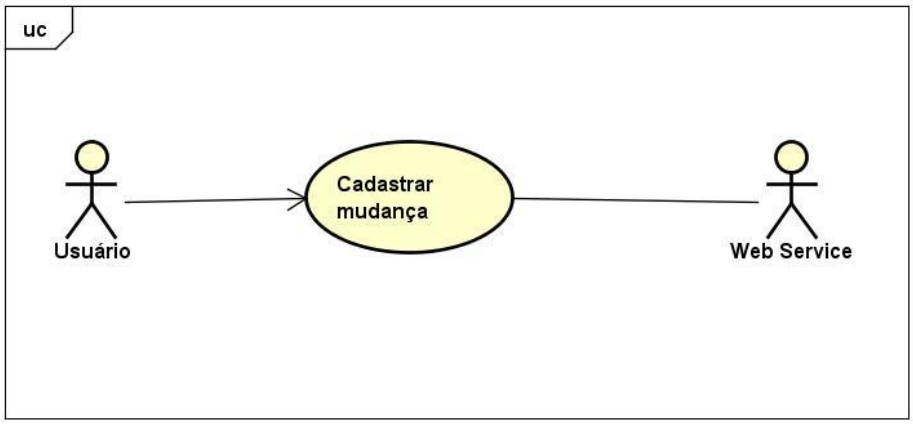


Figura 22 – Cadastro da mudança pelo usuário

Nome do Caso de Uso:

Cadastrar mudança.

Descrição:

Cadastrar os dados da mudança no servidor.

Atores:

- Usuário;
- WebService;

Requisitos:

O sistema deve permitir que o usuário cadastre os dados da mudança no WebService.

Fluxo principal:

1. Usuário efetuar login
1.1. Sistema valida dados (A1)
2. Usuário inclui os dados da mudança
3. Sistema valida dados
3.1. Valida os dados cadastrados (A2)
4. Sistema armazena os dados
5. Informa se houve falha ou sucesso

6. Fim do caso de uso

Fluxo alternativo:

A1: em “Sistema valida dados”, se os dados de login tiverem errados ou incompletos
A1.1: Retorne uma mensagem de erro

A1.2: Retorne o fluxo básico para o item “Usuário efetuar login”

A2: em “Sistema valida dados”, se os dados da mudança tiverem errados ou incompletos
A2.1: Retorne uma mensagem de erro

A2.2: Retorne o fluxo básico para o item “Usuário inclui os dados”

3.6.3. Solicitar fretista

O diagrama da Figura 23 tem como objetivo mostrar a escolha de um fretista.

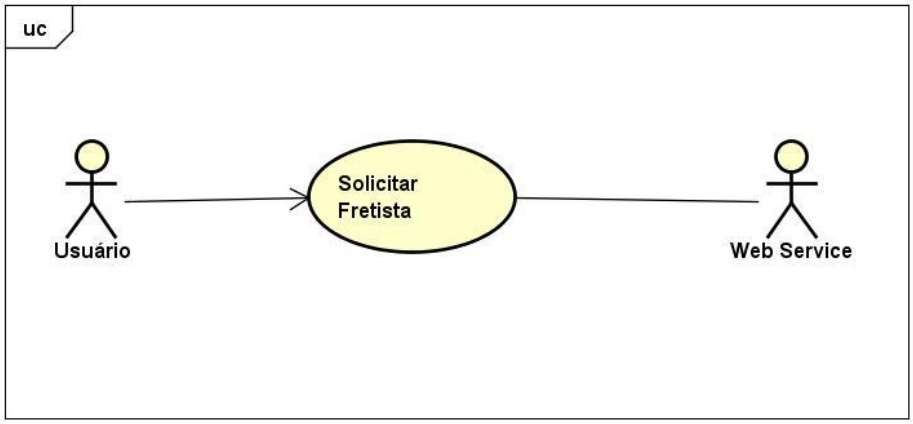


Figura 23 – Solicitar fretista

Nome do Caso de Uso:

Solicitar fretista.

Descrição:

Solicitar um fretista no sistema.

Atores:

- Usuário;

- WebService;

Requisitos:

O sistema deve permitir que o usuário consulte os fretistas disponíveis no sistema e selecione um para realizar sua mudança.

Fluxo principal:

1. Usuário efetuar login
1.1. Sistema valida dados (A1)
2. Usuário solicita uma lista de fretista
3. Sistema analisa pedido
3.1. Verifica se há fretista disponíveis no sistema (A2)
4. Sistema retorna uma lista de fretista
5. Usuário seleciona fretista
6. Sistema valida dados
6.1. Valida os dados enviados (A3)
7. Informa se houve falha ou sucesso
8. Fim do caso de uso

Fluxo alternativo:

A1: em “Sistema valida dados”, se os dados de login tiverem errados ou incompletos
A1.1: Retorne uma mensagem de erro

A1.2: Retorne o fluxo básico para o item “Usuário efetuar login”

A2: em “Verifica se há fretista disponíveis no sistema”, senão existir nenhum fretista cadastrado no sistema
A2.1: Retorne uma mensagem de erro
A2.2: Retorne o fluxo básico para o item “Usuário solicita uma lista de fretista”
A3: em “Valida os dados enviados”, se os dados do fretista estiverem errados ou incompletos
A3.1: Retorne uma mensagem de erro
A3.2: Retorne o fluxo básico para o item “Usuário solicita uma lista de fretista”

3.7. Diagramas de seqüência

Nesta seção serão listados alguns diagramas de seqüência referentes ao sistema.

3.7.1. Cadastrar usuário

A Figura 24 descreve o diagrama de seqüência ‘Cadastrar usuário’, que ilustra o evento de cadastrar os dados do usuário no WebService.

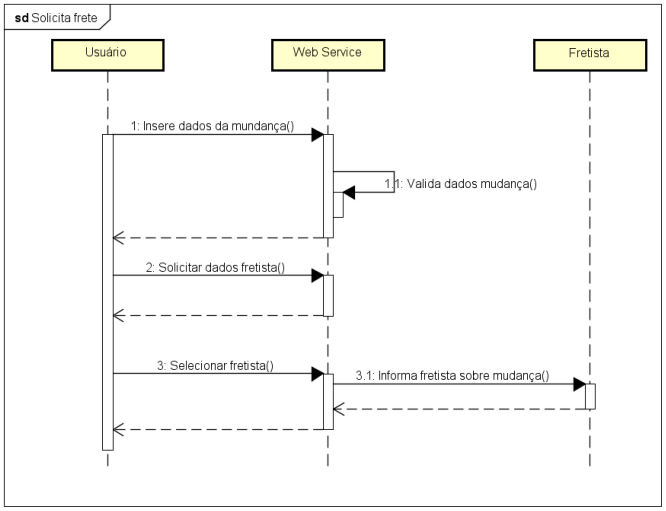


Figura 24 – Diagrama de seqüência ‘Cadastrar usuário’

3.7.2. Cadastrar mudança

A Figura 25 descreve o diagrama de seqüência ‘Cadastrar mudança’, que ilustra o evento de cadastrar os dados da mudança no Web Service.

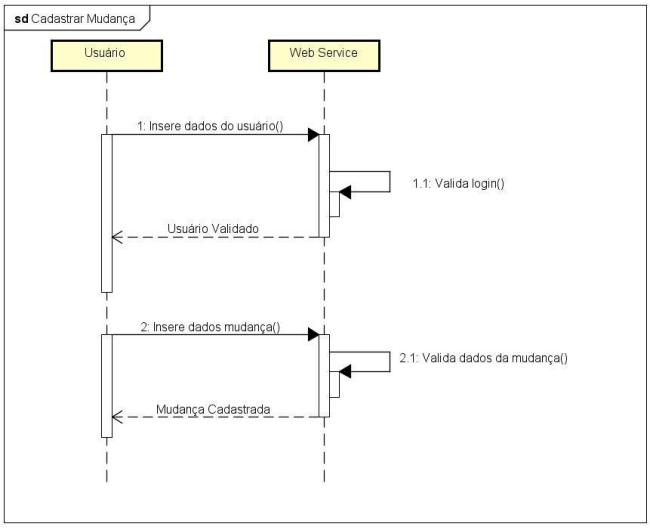


Figura 25 – Diagrama de seqüência ‘Cadastrar mudança’

3.7.3. Selecionar fretista

A Figura 26 descreve o diagrama de seqüência ‘Selecionar fretista’, que ilustra o evento de selecionar um fretista para realizar a mudança.

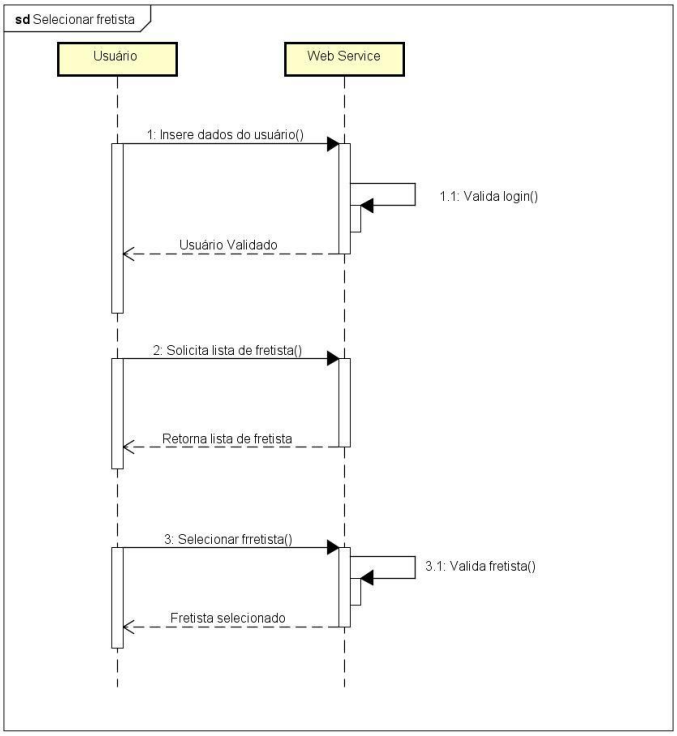


Figura 26 - Diagrama de seqüência ‘Selecionar fretista’

3.8. Diagrama de atividades

Nesta seção serão listados alguns diagramas de atividade referentes ao sistema.

3.8.1. Cadastrar fretistas no favoritos

A Figura 27 descreve o diagrama de atividade ‘Cadastrar fretistas no favoritos’, que ilustra o usuário solicitando a lista de fretistas para o WebService e registrando os fretistas no favoritos do aplicativo.

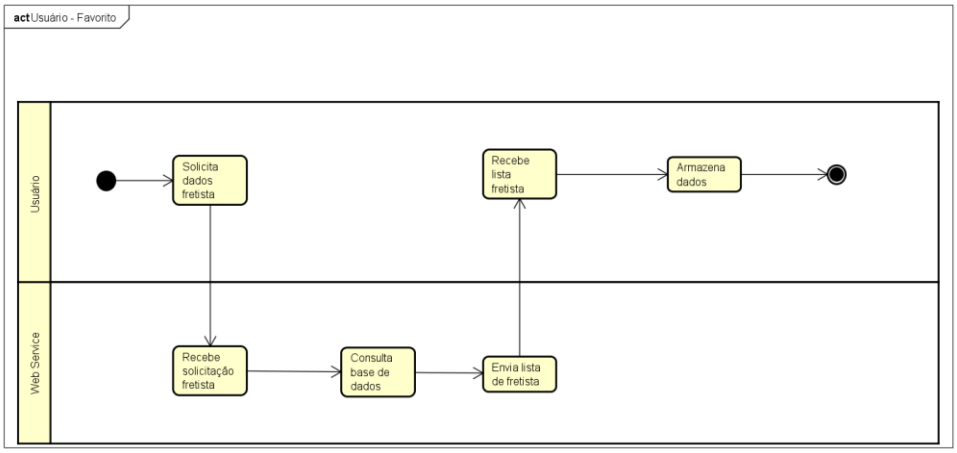


Figura 27 – Diagrama de atividades ‘Cadastrar fretistas no favoritos’

3.8.2. Chat do aplicativo

A Figura 28 descreve o diagrama de atividade ‘Chat do aplicativo’, que ilustra o usuário enviando uma conversa, o WebService registrando a mensagem e depois enviando ao fretista e depois fazendo o caminho inverso.

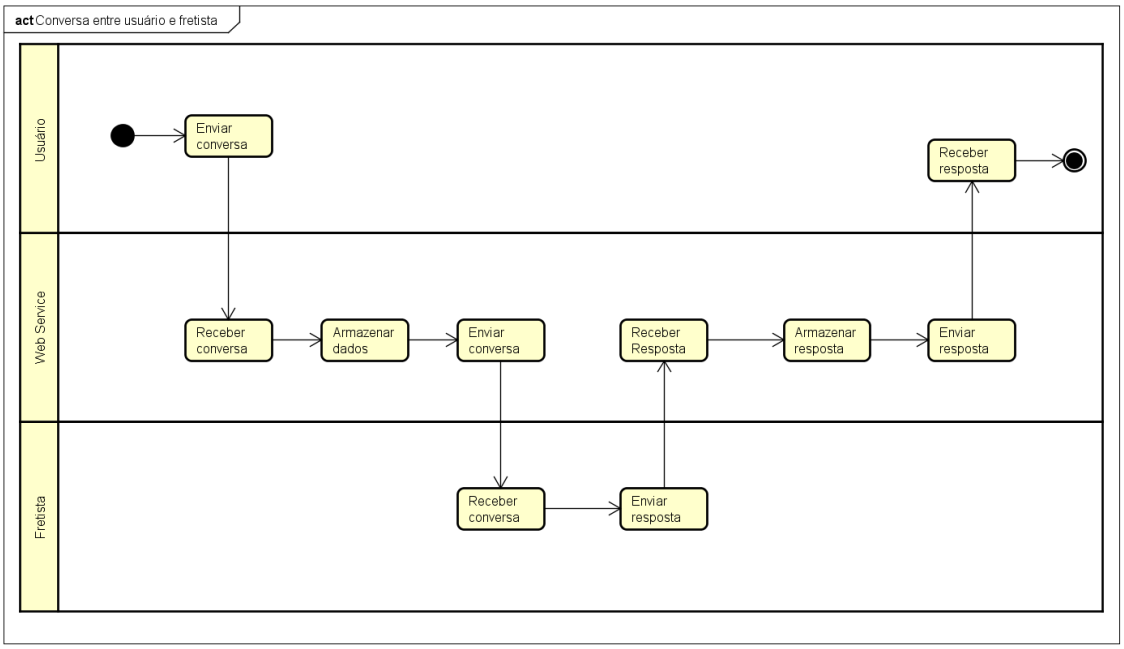


Figura 28 – Diagrama de atividades ‘Conversa entre usuário e o fretista’

3.9. Diagramas de classe

Nessa seção serão ilustrados os principais diagramas de classe do WebService e do Smartphone.

3.9.1. Web Service

A Figura 29 ilustra o diagrama de classes do ‘Web Service’ e suas interações com outras classes.

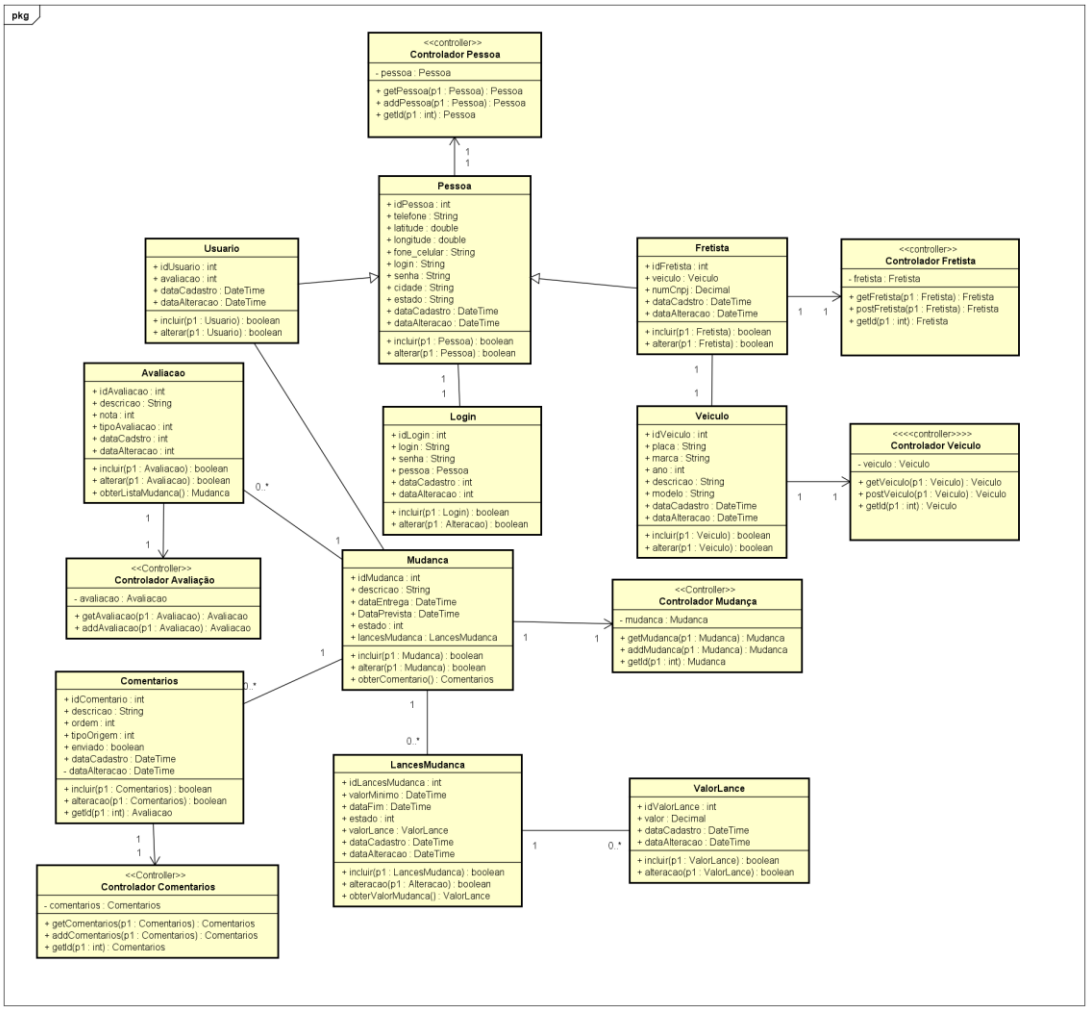


Figura 29 – Diagrama de classe ‘WebService’

As principais classes do ‘Web Service’ são: Usuario, Fretista, Avaliacao e Comentarios.

- Usuario: Contém os dados do usuário;

- Fretista: Contém os dados do fretista;
- Avaliacao: Contém as avaliações do fretista e do usuário;
- Comentarios: Contém os comentários do fretista e do usuário.

Os principais relacionamentos são:

- As classes Usuario e Fretista herdam as propriedades e métodos da classe Pessoa;
- A classe Mudanca está associada com as classes Comentarios e Avaliacao, onde todos os comentários e avaliações estão indiretos a uma mudança.

3.9.2. Smartphone

A Figura 30 ilustra o diagrama de classe ‘Smartphone’ e suas interações com outras classes.

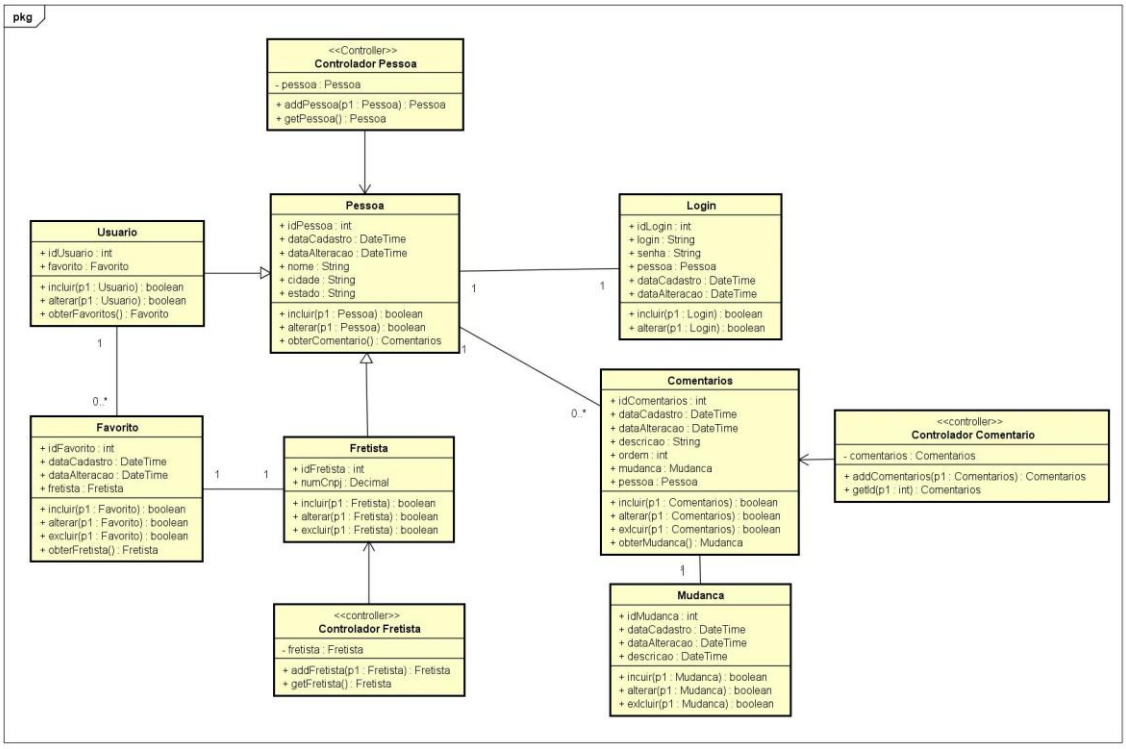


Figura 30 – Diagrama de classe ‘Smartphone’

As principais classes do ‘Smatrphone’ são: Usuario, Fretista, Favoritos e Comentarios.

- Usuario: Contém os dados do usuário;
- Fretista: Contém os dados do fretista;
- Favoritos: Contém os dados dos fretistas favoritos;
- Comentarios: Contém os comentários do fretista e do usuário.

Os principais relacionamentos são:

- As classes Usuario e Fretista herdam as propriedades e métodos da classe Pessoa;
- A classe Pessoa está associada com a classe Comentarios, onde todos os comentários e avaliações estão indiretos a classe Pessoa.

3.10. Projeto de interface

Nessa seção serão ilustradas algumas telas do aplicativo.

3.10.1. Menu

A figura 31 ilustra o ‘Menu’ do aplicativo, nele é possível acessar as principais funcionalidades do aplicativo como: Meu cadastro, Mudança, Favoritos e Avaliação.

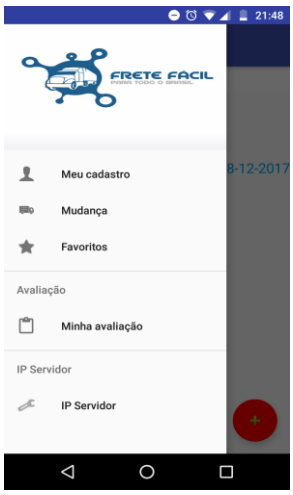


Figura 31 – Tela ‘Menu’

3.10.2. Tela do Chat

A figura 32 ilustra a tela ‘Chat’, essa tela é utilizada para que o usuário e o fretista possam se comunicar e retirar eventuais dúvidas.

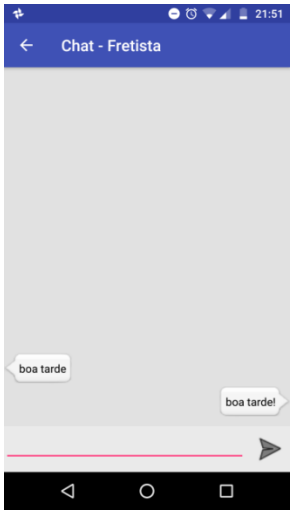


Figura 32 – Tela ‘Chat’

3.10.3. Tela da Mudança

A figura 33 ilustra a tela ‘Mudança’, nessa tela é possível visualizar todos os dados da mudança que o usuário cadastrou.

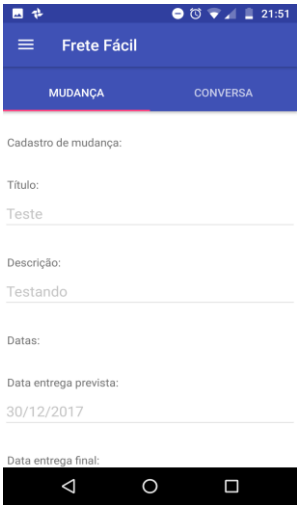


Figura 33 – Tela ‘Mudança’

4. Avaliação do Aplicativo

Nesse estudo de caso, será realizada uma pesquisa com os usuários para analisar os principais aspectos do aplicativo como: usabilidade, interface e outros, com objetivo de identificar a opinião dos usuários.

Para realizar a pesquisa foi utilizada a ferramenta do “*Google Forms*”, nela foram adicionadas perguntas que foram respondidas por três usuários com os seguintes perfis:

- Viviane
  - Idade: 35 anos;
  - Profissão: Supervisora de campo;
  - Freqüência de uso de aplicativos: Diariamente;
- Silvano
  - Idade: 44 anos;
  - Profissão: Caminhoneiro;
  - Freqüência de uso de aplicativos: Diariamente;
- Felipe
  - Idade: 26 anos;
  - Profissão: Programador CNC;
  - Freqüência de uso de aplicativos: Diariamente;

Segue as perguntas realizadas e as repostas obtidas:

- A interface do aplicativo é intuitiva?

A maioria dos usuários não achou a interface do aplicativo intuitiva, como pode ser visto na Figura 34.

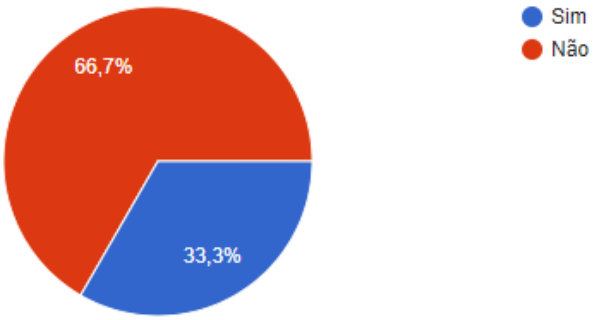


Figura 34 - Gráfico sobre a interface do aplicativo é intuitiva

- O tempo de resposta do aplicativo é satisfatório?

Todos os usuários acharam que o tempo de resposta foi satisfatório, como pode ser visto na Figura 35.

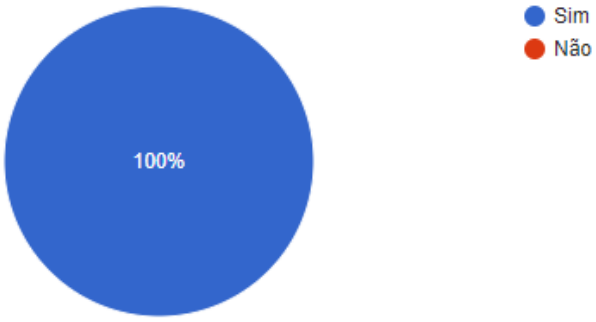


Figura 35 - Gráfico sobre o tempo de resposta do aplicativo foi satisfatório

- A navegação do aplicativo é intuitiva?

A maioria dos usuários achou a navegação intuitiva, como pode ser visto na Figura 36.

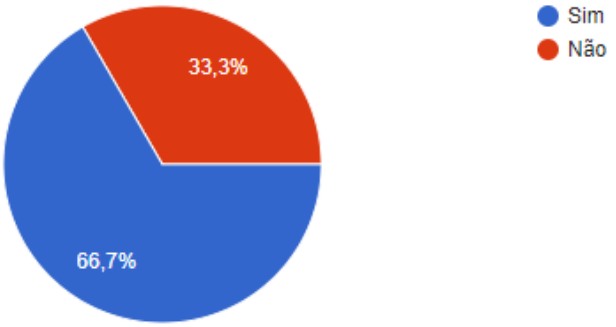


Figura 36 - Gráfico sobre a navegação do aplicativo

- Ocorreu algum erro durante o teste do aplicativo?

Os usuários informaram que não ocorreu nenhum erro no aplicativo durante os testes, como pode ser visto na Figura 37.

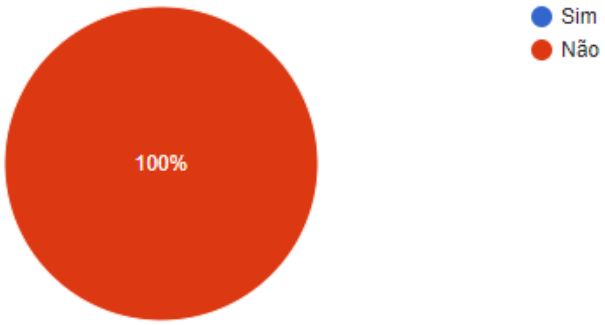


Figura 37 - Gráfico sobre erro durante o teste do aplicativo

- Você utilizaria esse aplicativo?

A maioria dos usuários não voltariam a utilizar o aplicativo, como pode ser visto na Figura 38.

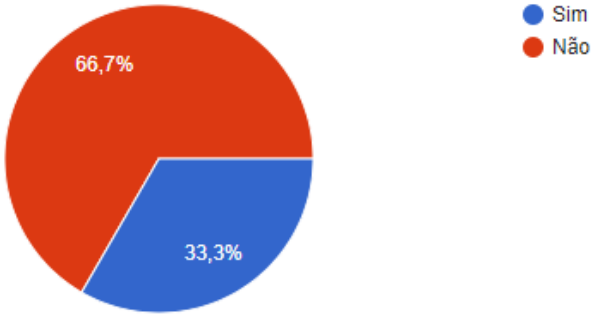


Figura 38 - Gráfico sobre se o usuário voltaria a utilizar o aplicativo

- O aplicativo fez o que prometia?

Os usuários informaram que o aplicativo fazia o que foi prometido, como pode ser visto na Figura 39.

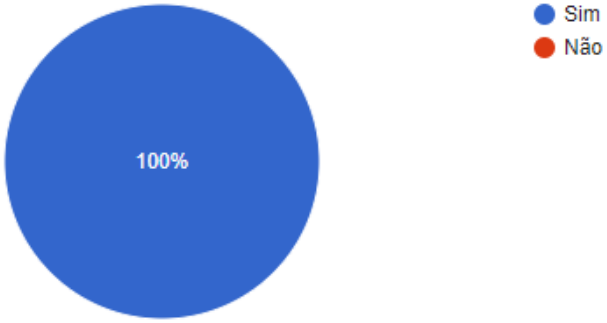


Figura 39 - Gráfico sobre se o aplicativo fez o que prometia

4.1. Resultados dos testes

Os testes realizados ilustraram que a interface do aplicativo apesar de ser intuitiva para maioria dos usuários que responderam o questionário é um item que deve ser melhorado como pode ser visto na Figura 33.

Na questão de tempo de resposta e erros ocorrido durante o teste do aplicativo, os usuários se mostraram satisfeitos, no qual nenhum problema foi relatado.

Todos os usuários afirmaram que o aplicativo faz o que foi prometido, porém a maioria não voltaria a utilizá-lo novamente, a maior quantidade de reclamações ficou na qualidade da interface.

As principais sugestões de melhoria foram:

- Tela de “Cadastro de Mudança”: melhorar a disposição dos controles na tela, para que as informações não fiquem muito juntas;
- Tela de “Pesquisa de Fretista”: Retornar as avaliações dos fretistas junto com seus dados.

5. Conclusões

O aplicativo Frete Fácil foi desenvolvido com o objetivo de atender uma necessidade do mercado de mudanças, no qual de um lado existe o usuário com interesse em fazer uma mudança e do outro lado o fretista à espera de um frete.

No sistema desenvolvido, em poucos passos tanto o usuário quanto o fretista pode cadastrar seu dados e começar a ver as opções disponíveis dentro do sistema. Para isso basta ter um "*smartphone"* com o sistema operacional "*Android"* e estar conectado à internet. A principal vantagem é a aplicação ser executada no *smartphone*, facilitando o seu acesso e uso em qualquer lugar que tenha sinal de operadora.

Para gerenciar todo o sistema foi desenvolvido um *web service*, ele é responsável por controlar toda a lógica do sistema e os acessos. A vantagem de utilizar esse sistema é que todas as operações ficam disponíveis na internet, podendo ser consumida por qualquer aplicação.

Os principais objetivos do projeto foram atingidos com sucesso, resultando no desenvolvimento de um sistema móvel, capaz de facilitar a comunicação entre o usuário e o fretista, trazendo benefícios para o usuário e maior oportunidade de negócios para o fretista. Já o sistema de lance não será implementado nesse primeiro momento, por questão de tempo.

Entre as dificuldades encontradas, ficou no desenvolvimento do *chat* por causa dos vários usuários que podem trocar informações ao mesmo tempo, o controle do sistema de mudanças e o desenvolvimento do *web service* para gerenciar toda a lógica do sistema.

Sendo assim, o aplicativo Frete Fácil se mostrou relevante para o mercado atual, mostrando que é uma solução viável e que pode ajudar na relação entre usuário e fretista trazendo facilidade e mais negócios.

Para trabalho futuro é sugerido à implementação do rastreamento dos veículos. Com essa opção será possível mostrar no mapa a posição real de cada fretista e aonde se encontra o fretista contratado. Algumas melhorias

podem ser feitas no *layout* e na segurança dos dados trafegados pelo aplicativo.

6. Referências

127Bytes. AndroidManifest. Disponível em https://127bytes.wordpress.com/2010/02/19/androidmanifest-xml-demystified, acessado em 16/09/2017.

Abril. Conheça a história do Android, o sistema operacional mobile da Google. Disponível em https://super.abril.com.br/galeria/conheca-a-historia-do-android-o-sistema-operacional-mobile-da-google/ >, acesso em 17/09/2017.

Android Arquitetura. Arquitetura da plataforma. Disponível em <https://developer.android.com/guide/platform/index.html?hl=pt-br>, acessado em 28/09/2017.

Android Atividades. Atividades. Disponível em <https://developer.android.com/guide/components/activities.html >, acessado em 29/09/2017.

Android Criar um Projeto. Criar um Projeto. Disponível em <https://developer.android.com/studio/projects/create-project.html>, acessado em 28/09/2017.

Android Introduction. Introduction to Activities. Disponível em <https://developer.android.com/guide/components/activities/intro-activities.html>, acessado em 25/09/2017.

Android Manifesto. Manifesto do Aplicativo. Disponível em <https://developer.android.com/guide/topics/manifest/manifest-intro.html?hl=pt-br>, acessado em 26/09/2017.

Android Pilha. Tarefas e pilha de retorno. Disponível em <https://developer.android.com/guide/components/tasks-and-back-stack.html?hl=pt-br>, acessado em 25/09/2017.

AndroidPro.Activity: O que é e como usar corretamente. Disponível em <http://www.androidpro.com.br/activity-intro/>, acessado e 24/09/2017

Android Studio. Conheça o Android Studio. Disponível em https://developer.android.com/studio/intro/index.html?hl=pt-br, acessado e 24/09/2017.

SaibaMais. O que é Design Sprint: como funciona e como aplicar no seu projeto. Disponível em < http://www.saiba-mais.com/2016/01/26/design-sprint/ >, acessado e 24/09/2017.

Caelum. Experiência do Usuário. Disponível em < https://www.caelum.com.br/apostila-ux-usabilidade-mobile-web/experiencia/>, acessado em 26/09/2017.

Correio do Povo. Acesso à internet móvel no Brasil cresce 70%, revela IBGE em <http://www.correiodopovo.com.br/Noticias/Tecnologia/2016/4/583727/Acesso-a-internet-movel-no-Brasil-cresce-70,-revela-IBGE>, acessado em 08/02/2016.

DevMedia. Entendendo o ciclo de vida de uma aplicação Android. Disponível em <http://www.devmedia.com.br/entendendo-o-ciclo-de-vida-de-uma-aplicacao-android/22922>, acessado em 28/09/2017.

DevMedia WSDL. Introdução às tecnologias Web Services: SOA, SOAP, WSDL e UDDI - Parte1 em < https://www.devmedia.com.br/introducao-as-tecnologias-web-services-soa-soap-wsdl-e-uddi-parte1/2873>, acessado em 06/02/2018.

EBC. Acesso à internet chega a 49,4% da população brasileira. Disponível em <http://www.ebc.com.br/tecnologia/2015/04/acesso-internet-chega-494-da-populacao-brasileira>, acesso em 01/09/2017.

EBC. IBGE: celular se consolida como o principal meio de acesso à internet no Brasil. Disponível em <http://agenciabrasil.ebc.com.br/geral/noticia/2016-12/ibge-celular-se-consolida-como-o-principal-meio-de-acesso-internet-no-brasil>, acesso em 01/09/2017.

EricoFileno. Design Interação. Disponível em <https://ericofileno.wordpress.com/tag/conceito/> , acesso em 17/09/2017.

GitHub. Retrofit. Disponível em <http://square.github.io/retrofit/>, acesso em 05/09/2017.

IBM. Relacionamento entre UDDI e WSDL. Disponível em

< https://www.ibm.com/support/knowledgecenter/pt-br/SS8PJ7_9.6.1/org.eclipse.jst.ws.consumption.ui.doc.user/concepts/cwsdlud.html>, acesso em 15/09/2017.

IMaster Google Sprint. Participando do Google Design Sprint. Disponível em <https://imasters.com.br/design-ux/participando-do-google-design-sprint/?trace=1519021197>, acesso em 14/09/2017.

iMaster. Protocolo de Transporte Padrão – SOAP. Disponível em <https://imasters.com.br/artigo/4379/web-services/protocolo-de-transporte-padrao-soap/?trace=1519021197&source=single>, acesso em 04/09/2017.

InternetDasCoisas. Descobrindo o conceito REST. Disponível em <https://internetdetodasascoisas.wordpress.com/2015/10/31/descobrindo-o-conceito-rest/ >, acesso em 04/09/2017.

Kirupa. XML Files Sequentially. Disponível em <https://www.kirupa.com/net/readingXML_pg1.htm>, acesso em 19/10/2017.

Macoratti. XML – Introdução e conceitos básicos. Disponível em <http://www.macoratti.net/xml.htm>, acessado em 18/10/2017.

Microsoft. Arquivos de formato XML de exemplo. Disponível em <https://technet.microsoft.com/pt-br/library/ms191234(v=sql.105).aspx>, acesso em 16/10/2017.

Microsoft. WebService . Disponível em <https://msdn.microsoft.com/pt-br/library/aa480728.aspx>, acesso em 16/10/2017.

Netbeans. Netbeans IDE. Disponível em <https://netbeans.org/>, acesso em 22/09/2017.

Novell. What is UDDI. Disponível em <https://www.novell.com/documentation/oes/uddi/?page=/documentation/oes/uddi/data/aioge19.html>, acesso em 11/09/2017

OpenSoft. Web service: o que é, como funciona, para que serve?. Disponível em <https://www.opensoft.pt/web-service/>, acesso em 03/09/2017.

Portal Educação. Detalhando as diferenças Entre URI e URL. Disponível em <https://www.portaleducacao.com.br/conteudo/artigos/idiomas/detalhando-as-diferencas-entre-uri-e-url/20377 >, acesso em 04/09/2017.

SQLite. SQLite. Disponível em <https://www.sqlite.org/>, acesso em 05/09/2017.

Stackoverflow. SOAP vs REST. Disponível em

<https://stackoverflow.com/questions/19884295/soap-vs-rest-differences>, acesso em 15/09/2017.

Tecmundo. O que é XML. Disponível em <https://www.tecmundo.com.br/programacao/1762-o-que-e-xml-.htm>, acesso em 15/09/2017.

TheClub. Ciclo de vida de uma atividade. Disponível em <http://theclub.com.br/Restrito/Revistas/201209/andr1209.aspx>, acesso em 12/09/2017.

UBER Dirija. UBER. Disponível em <https://www.uber.com/pt-BR/>, acesso em 19/10/2017.

UDDI . WSDL and UDDI. Disponível em <http://w3schools.sinsixx.com/wsdl/wsdl_uddi.asp.htm>, acesso em 06/02/2018.

Unicamp. Introdução a Linguagens de Marcação: HTML, XHTML, SGML, XML. Disponível em <http://www.ic.unicamp.br/~celio/inf533/docs/markup.html>, acesso em 14/09/2017.

Univast. O que é Design Interação. Disponível em < http://www.univasf.edu.br/~jorge.cavalcanti/cap_01_design_interacao.pdf >, acesso em 17/09/2017.

UOL. Smartphones estão nas mãos de 62% dos brasileiros, diz Google. Disponível em < http://www1.folha.uol.com.br/tec/2017/02/1862362-smartphones-estao-nas-maos-de-62-dos-brasileiros-diz-google.shtml>, acesso em 01/09/2017.

Vogella. Using Retrofit 2.x as REST client – Tutorial. Disponível em <http://www.vogella.com/tutorials/Retrofit/article.html>, acesso em 05/09/2017.

XML. W3C. Disponível em <https://www.w3.org/standards/xml/core>, acesso em 17/01/2018.

W3. W3C. Disponível em <https://www.w3.org/TR/cooluris/>, acesso em 12/10/2017.

Zarelli, Qual a diferença entre SOAP 1.1 e SOAP 1.2, acesso em

<https://zarelli.wordpress.com/2012/03/22/como-funciona-o-soap-protocolo-simples-de-acesso-a-objetos/>, acesso em 19/09/2017.

7. Apêndices

Diagrama de classe do WebService - Usuário

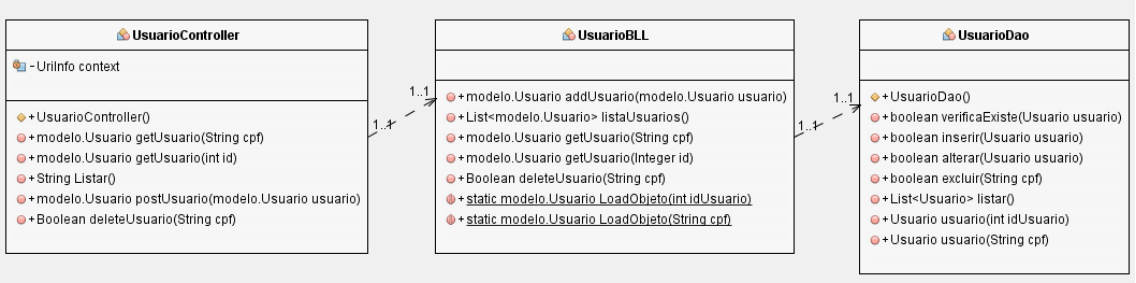


Diagrama de classe do WebService – Chat

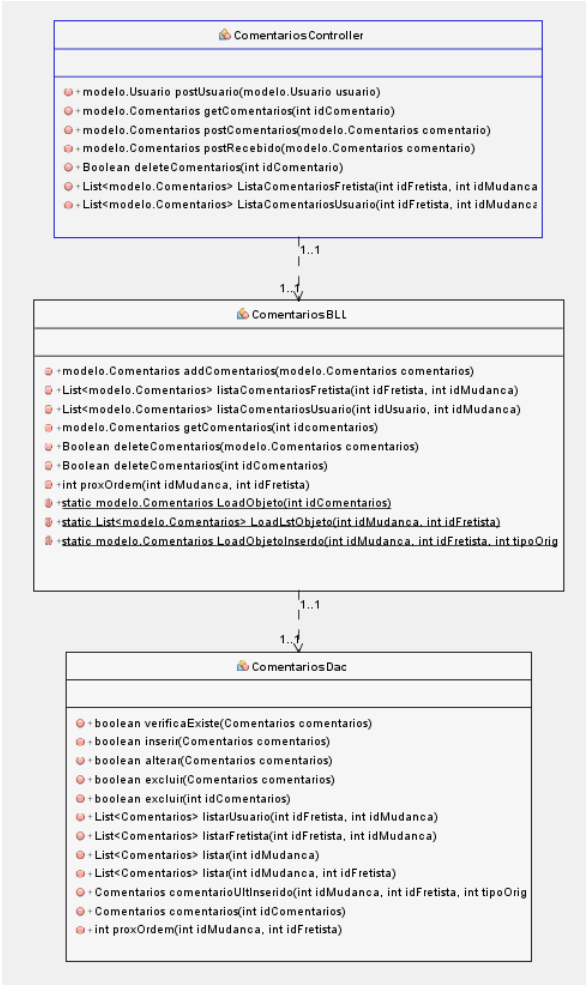
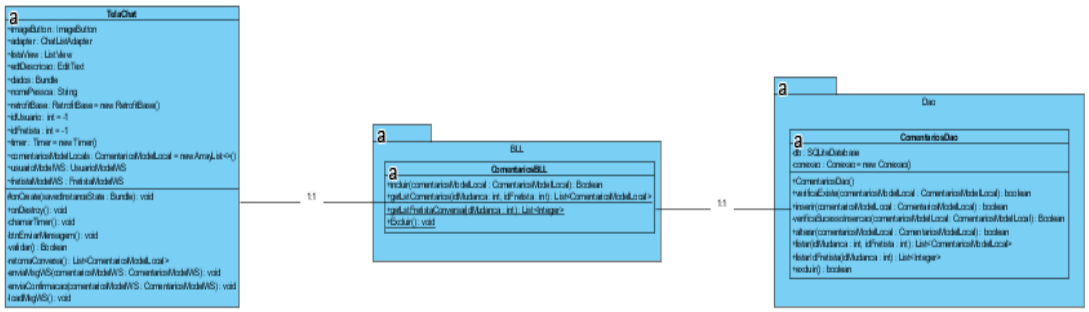


Diagrama de classe do '*SmarthPhone* '– Chat


TelaChat
BLL
ComentariosBLL
Dao
ComentariosDao
1.1
1.1