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

ANDRÉ CORADIN GULIN

DESENVOLVIMENTO DE SISTEMA GRATUITO DE GERENCIAMENTO DE COMÉRCIOS PARA DISPOSITIVOS MÓVEIS

CURITIBA
2017

ANDRÉ CORADIN GULIN

DESENVOLVIMENTO DE SISTEMA GRATUITO DE GERENCIAMENTO DE COMÉRCIOS PARA DISPOSITIVOS MÓVEIS

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 Ribeiro 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

"Desenvolvimento de Sistema Gratuito de Gerenciamento de Comércios para Dispositivos Móveis"

por

"André Coradin Gulin"

Este Trabalho de Conclusão de Curso foi apresentado às 11:10 do dia 24 de novembro de 2017 na sala B108 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 - UTFPR/Curitiba)**	_____ Profa. Maria Claudia Figueiredo Pereira Emer **(Avaliador 1 – UTFPR/Curitiba)**
_____ Prof. Alexandre Reis Graeml **(Avaliador 2 – UTFPR/Curitiba)**	

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

RESUMO

Tomando por base a crescente acessibilidade a dispositivos móveis, em especial dispositivos com sistema *Android*, por grande parte da população, a necessidade de gerenciamento de atividades relacionadas ao comércio para um melhor desempenho da mesma e a análise de que os aplicativos disponíveis para este fim no mercado atual são pagos, este trabalho apresenta um sistema gratuito de gerenciamento para auxiliar nas diversas atividades comerciárias. Para conceituação do sistema desenvolvido, este trabalho classifica o mesmo como um sistema de gestão de vendas, ou seja, um conjunto de módulos (subsistemas) interligados que fornecem controle total da atividade fim. Por conta desta característica de sistema de gestão de vendas, esse trabalho sugeriu vários módulos para composição do aplicativo que seriam: de cliente, fornecedores, produtos, estoque, vendas e pontos de entrega integrados em uma rota. Todos esses subsistemas foram desenvolvidos e integrados utilizando-se ferramentas consolidadas na atualidade para desenvolvimento para dispositivos móveis com sistemas *Android*. Dentre estas ferramentas vale salientar o uso de bibliotecas de geo localização fornecidas pelo *Google*, banco de dados orientado a objetos *Realm*, gerenciamento de projeto *Scrum* e testes de usabilidade em conjunto com *System Usability Scale*. A aplicação desta última ferramenta confirmou com uma boa margem, a completude do objetivo do aplicativo e o interesse dos usuários avaliados pelo sistema proposto.

Palavras-chave: comércio, aplicativo, desenvolvimento.

ABSTRACT

Considering the increasing accessibility to mobile devices, especially Android-based devices, by a large part of the population, the need to manage trade-related activities for better performance of the same, and the analysis showing that the applications available for this purpose in the current market are paid, this work presents a free management system to assist in various trade-related activities. In order to conceptualize the developed system, this work classifies it as an sales management system, that is, a set of interconnected modules (subsystems) that provide total control of the end activity. Due to this sales management system characteristic, this work suggested several modules for the composition of the application that would be: customer, suppliers, products, inventory, sales and delivery points integrated into a route. All these subsystems have been developed and integrated using currently consolidated tools for development for mobile devices with Android systems. Among these tools it is worth noting the use of geo location libraries provided by Google, Realm object-oriented database, Scrum project management and usability tests used with System Usability Scale. The application of this last tool has confirmed with a good margin the completeness of the objective of the application and the interest of the users evaluated by the proposed system.

Keyword: commerce, app, development.

LISTA DE ILUSTRAÇÕES

SUMÁRIO

1 INTRODUÇÃO

1.1 Contexto

A atividade de comércio não possui uma data de criação ou origem específica, apenas se sabe que, assim que surgiram as primeiras comunidades, o comércio foi estabelecido por conta da necessidade de troca de bens entre as pessoas, já que um indivíduo não era capaz de produzir todos os itens necessários a sua sobrevivência (SOUZA, 2007) (COLÉGIO WEB, 2014). Por ser uma atividade que acompanha a humanidade há tempos, também seguiu os fluxos revolucionários desta. Dentre elas, valem serem destacadas as Revoluções Industriais que tiveram e têm grande influência como a atividade do comércio é realizada.

A partir da primeira revolução industrial, criou-se uma maior oferta de produtos e, por consequência, uma necessidade de fazer com que esses produtos alcançassem mais pessoas e, também a necessidade de mais pessoas especializadas em fazer essa atividade, a saber, os comerciários. Com as revoluções industriais também surgiram novos papéis dentro da cadeia de produção, onde antes existiam apenas o mestre e aprendiz, após a revolução existem operários, supervisores, gerentes etc.

É para estes últimos que se encontra o foco deste trabalho, pois segundo KLADIS e FREITAS (1996), os gerentes representam uma peça importante nessa cadeia produtiva, sendo que tal importância advém de ampla gama de atividades, em especial, a de tomada de decisão. Também é importante destacar, como elencado KLADIS e FREITAS (1996) em sua referência a SIMON (1965, 311p), que um dos fatos que pode limitar o desempenho de um gerente é o quanto este sabe sobre o estado da empresa/negócio. Estes argumentos são corroborados pelo Serviço Brasileiro de Apoio às Micro e Pequenas Empresas (SEBRAE) (SEBRAE, 2010) (SEBRAE, 2015), que salienta que uma das principais atividades de um gerente é planejar (tomar decisão) e controlar/monitorar (possuir conhecimento do estado da atividade).

Assim, por serem atividades reconhecidamente importantes, criou-se a oportunidade de criação de ferramentas para auxiliar gerentes nesse quesito, e a revolução tecnológica trouxe isso com o advento de vários tipos de sistemas, dentre os quais, sistemas de Gestão de Vendas.

Esses sistemas, para atingir seu objetivo de ferramenta auxiliar ao gerente, buscam obter dados em tempo real da empresa/negócio, bem como uma forma objetiva de entregar essa informação ao interessado. Para fazer isso, esse sistema é composto de vários outros subsistemas, como por exemplo, gerenciador de histórico de vendas feitas, de estoque, de notas emitidas etc (JUNIOR, 2017).

1.2 Objetivos

Este trabalho tem por objetivo geral desenvolver um sistema para dispositivos móveis, possível de ser utilizado como ferramenta de controle e suporte no dia-a-dia da atividade comerciária. Para atingir esse objetivo, são propostos também os seguintes objetivos intermediários:

- Levantamento de requisitos com usuários em potencial;
- Análise dos requisitos levantados, gerando como produto modelos para codificação do sistema;
- Codificação do sistema;
- Testes com estudos de casos de diferentes tipos de usuários;
- Com base nos estudos de caso gerados, promover uma análise de aceitação dos usuários, buscando melhorias e novos entendimentos para o problema abordado.

1.3 Motivação

Os principais motivos para o desenvolvimento deste projeto são: o prolongado contato pessoal com a área fim, o aprendizado das tecnologias para aparelhos móveis e desenvolvimento de sistemas, e a visão da necessidade de inovação no método de trabalho usado hoje no ambiente conhecido.

Tal necessidade não se encontra infundada, pois, pela área fim ser a de comércio, há a constante busca de novas formas de alcançar clientes e aumentar vendas. Tome-se o exemplo do *e-commerce* brasileiro, que possui uma estimativa de movimentar em torno de 49 bilhões de reais para esse ano(STATISTA, 2017), sendo que há apenas cinco anos movimentava menos que a metade desse valor.

Para esse crescimento, é possível encontrar várias razões, seja o aumento no número de usuários (INTERNET LIVE STATS, 2017), inclusão digital, aumento de tecnologias e acesso a ferramentas para auxílio à atividade comércio desse tipo etc.

E assim, tem-se ao motivo do desenvolvimento dessa labuta para dispositivos móveis, mais precisamente para smartphones. É inegável a presença de dispositivos móveis como a tecnologia “do momento”, pois há cada vez mais *gadgets* com sistemas embutidos e conectados sendo apresentados à sociedade. Também, é difícil de dizer que os *smartphones* não sejam os *gadgets* mais utilizados dentre os vários dispositivos móveis, seja por serem úteis em várias atividades do dia-a-dia, bons meios de entretenimento ou simplesmente por já terem uma necessidade de compra estabelecida na sociedade atual.

E, portanto, pensando no *smartphone* como auxiliar nas tarefas cotidianas e o desenvolvimento tecnológico como ferramenta para auxílio à atividade do comércio, pergunta-se: por que não um sistema para controle dessa atividade para esses aparelhos? Afinal, para que alguma atividade consiga atingir um objetivo é necessário planejar (SEBRAE, 2017), acompanhar e gerenciá-la.

Assim, esse projeto se propõe a oferecer aos comerciários uma ferramenta disponível em seus celulares que agrupe informações de seus negócios, para que, a qualquer momento, possam obter informações e tomar melhores decisões, sejam elas sobre preparação para sazonalidades, necessidade de novos recursos etc.

1.4 Escopo

Como visto anteriormente, um sistema de Gestão de Vendas robusto, voltado para uma grande empresa teria vários módulos, como por exemplo, emissão de nota fiscal, controle de estoque, dentre outros. Porém, foge do contexto deste trabalho o desenvolvimento de módulo de emissão de nota fiscal, visto que, para a criação de tal funcionalidade, seria necessário maior tempo para busca de intercomunicação com receitas estaduais e/ou federais.

Assim, neste projeto serão desenvolvidos apenas módulos de gerência de clientes, fornecedores, produtos, estoque, vendas e rotas.

1.5 Metodologia

No decorrer deste trabalho será possível verificar as seguintes etapas em seu desenvolvimento:

- Análise de aplicativos já existentes no mercado para observar padrões e usabilidade e apresentação;
- Análise de ferramentas para criação de aplicativos móveis com relação a bibliotecas que tornem mais rápida tal criação;
- Coleta de requisitos do usuário com entrevistas no formato de histórias do usuário;
- Adoção da ferramenta de gerenciamento de projeto *Personal Scrum* e por consequência, criação dos artefatos necessários;
- Elaboração do artefato de código e apresentação neste trabalho via telas e breve explicação de uso;
- Elaboração de teste de usabilidade e seleção de conjunto de avaliadores distintos para uma maior abrangência dos testes;
- Analise dos resultados obtidos utilizando a escala *System Usability Scale.*

1.6 Organização do Trabalho

No Capítulo 2 são apresentadas as tecnologias utilizadas neste trabalho, de forma a fornecer uma base teórica ao leitor para acompanhamento do desenvolvimento do trabalho.

No Capítulo 3 estão descritos os detalhes mais relevantes da implementação do sistema.

No Capítulo 4 são apresentados estudos de caso do sistema proposto com usuários em potencial, bem como análise dos mesmos.

No Capítulo 5 são apresentadas conclusões tiradas tanto no desenvolvimento da aplicação quanto nos testes, verificando se o sistema conseguiu atingir o objetivo de tornar-se uma ferramenta de auxílio à tomada de decisão.

No Capítulo 6 são elencadas todas as referências consultadas para o desenvolvimento desse projeto.

No Capítulo 7 são apresentados os apêndices relevantes deste trabalho.

2 REFERENCIAL TEÓRICO E ANALISE DAS TECNOLOGIAS

2.1 Sistema Gestão de Vendas

Com o passar do tempo e com o crescimento do negócio, donos e gerentes passam a perceber a necessidade de um sistema que integre todas as áreas de sua atividade, seja porque o controle manual está se tornando ineficaz e impraticável ou porque o controle utilizando diversos sistemas diferentes está apresentando problemas de comunicação. Para ambos os casos uma solução é a utilização de um tipo de *software* conhecido por Sistema Gestão de Vendas.

Estes sistemas são soluções de *software* especializadas, desenvolvidas de acordo com a necessidade do negócio. Isso porque são compostos de vários módulos ou subsistemas interligados que gerenciam cada atividade envolvida no escopo da organização.

Então, supondo um negócio fictício que possua uma seção de controle de entregas, essa irá possuir um módulo do sistema de gestão de vendas de controle de entregas. Para este exemplo será suposto que existe uma seção de vendas para esse mesmo negócio específico. Nesta seção de vendas também existirá um módulo do sistema gestão de vendas que será de controle de vendas.

Por serem ambos módulos do mesmo sistema, estes estão integrados e assim, cada vez que surgir uma nova venda e for necessária a entrega, a seção responsável será notificada rapidamente.

Assim, para o desenvolvimento de um *software* que objetiva ser um sistema Gestão de vendas, é fundamental oferecer módulos para todas as atividades envolvidas no caso analisado e fazer com que estas tenham a maior integração possível (JUNIOR, 2017)..

2.1.1 Sistemas disponíveis para dispositivos móveis.

Por ser uma ferramenta importante para controle de negócios e que precisa ser especializada para cada tipo de situação, há uma grande variedade de sistemas de gestão disponíveis no mercado. Para celulares com sistema *Android*, que é a plataforma foco deste trabalho, é possível encontrar um número de aproximadamente 250 (GOOGLE PLAY, 2017) aplicativos na loja de aplicativos *Google Play* quando procurado pelo termo "*gestão de vendas*". A seguir serão elencados alguns destes aplicativos, usando como critério de escolha aplicativos com maior proximidade ao objetivo desse trabalho, avaliação recebida pelos usuários e que tenham linguagem da interface em português.

2.1.1.1 IDS *ERP* Mini

Na figura1 são apresentadas telas do aplicativo IDS *ERP* Mini.

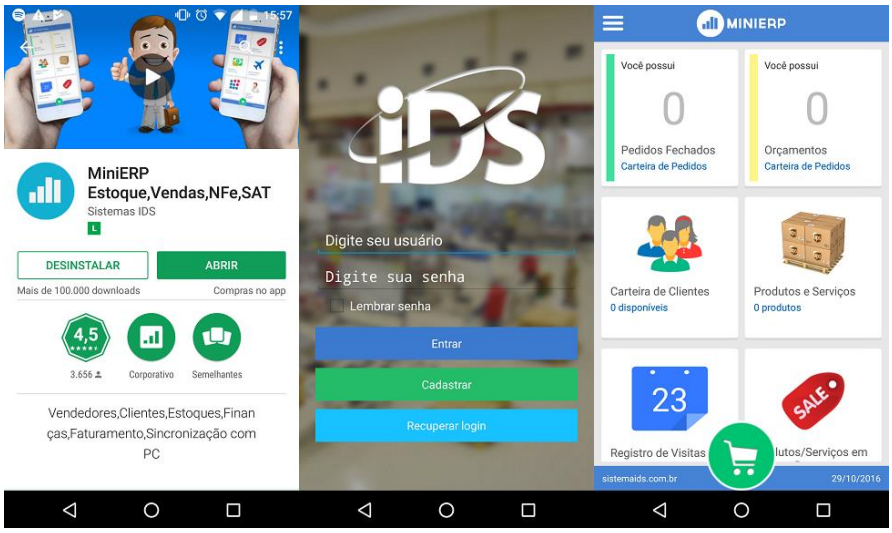


Figura 1 - Captura de telas de IDS *ERP* Mini

Esse aplicativo apresenta os seguintes módulos (GOOGLE PLAY, 2017):

- Gerência de pedidos;
- Gerência de orçamentos;
- Gerência de clientes;
- Gerência produtos/serviços, com funcionalidade para associar um código de barras capturado pela câmera;
- Relatórios, porém apenas disponível na versão paga do aplicativo;
- Exportar dados do sistema para arquivo *PDF*;
- Emissão de nota fiscal eletrônica, porém disponível apenas na versão corporativa do aplicativo.

Este sistema é composto por vários módulos importantes para um negócio simples, juntamente com uma interface, até certo ponto, simples de ser utilizada, porém a apresentação de relatórios para melhor visualização dos dados do negócio está disponível apenas na versão paga. Um diferencial entre as funcionalidades propostas por este aplicativo e as deste trabalho seria a apresentação de rota de entrega para usuário e catálogo de fornecedores.

2.1.1.2 Meus Pedidos - Gestão e controle de pedidos, vendas e clientes

Na figura 2 são apresentadas telas de utilização e apresentação do aplicativo Meus Pedidos.

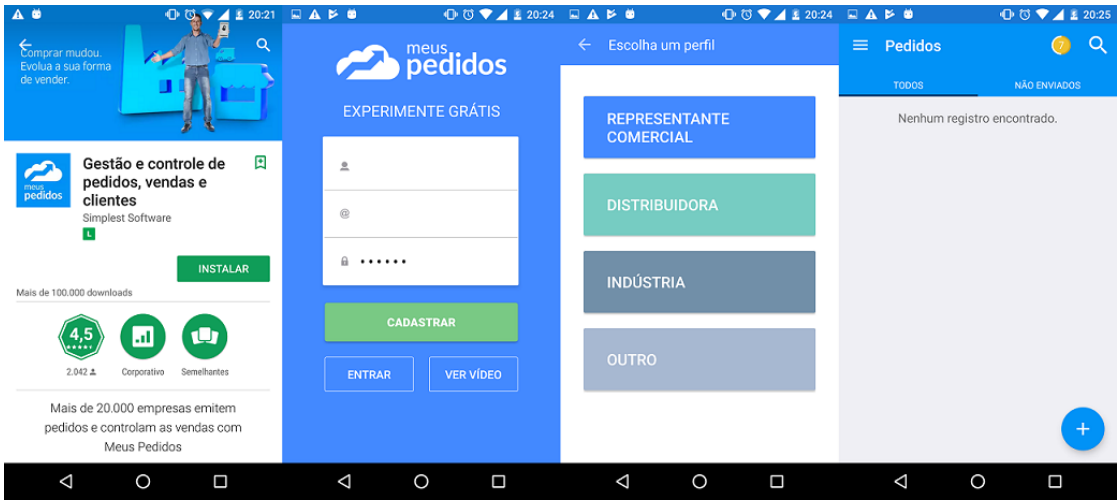


Figura 2 - Captura de telas de Meus Pedidos

Aplicativo contendo os seguintes módulos (GOOGLE PLAY, 2017):

- Gerência de pedidos;
- Gerência de orçamentos;
- Gerência de clientes;
- Gerência produtos;
- Gerência de vendedores;
- Relatórios diversos;
- Integração com *Excel;*
- Exportar dados de pedidos para arquivo *PDF*.

O sistema “Meus pedidos” também possui uso e acesso fáceis, porém diferentemente do anterior, neste é possível utilizar o sistema apenas por um período limitado de tempo, sendo necessária depois desse tempo de avaliação a assinatura do serviço da empresa, sendo que no aplicativo proposto o acesso seria irrestrito.

2.1.1.3 Suas Vendas

Na figura 3 a seguir é apresentada a tela de descrição do aplicativo na loja de aplicativos do *Google,* bem como a tela inicial do mesmo.

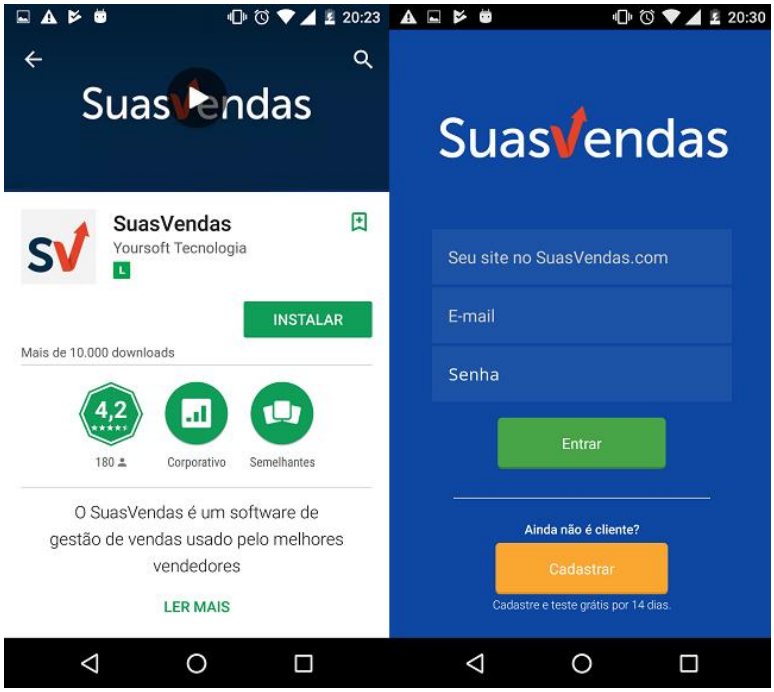


Figura 3 - Captura de telas de Suas Vendas

Aplicativo contendo os seguintes módulos (GOOGLE PLAY, 2017)::

- Gerência de pedidos;
- Gerência de orçamentos;
- Gerência de clientes, com possibilidade de marcação em mapa da localização dos mesmos;
- Gerência produtos;
- Gerência de vendedores, com possibilidade de acompanhar a atividade de cada vendedor em visita à cliente;
- Relatórios diversos.

O aplicativo “Suas Vendas” conta com diversas funcionalidades, porém dependente de um longo cadastro e, como o sistema anterior, também dependente de assinatura, sendo assim, também difere do aplicativo por este permitir acesso sem restrição a todas as funções do sistema.

2.2 Tecnologias empregadas

Para elaboração deste projeto serão utilizadas diversas tecnologias, métodos e técnicas, tanto para codificação do *software* em si, como para seu acompanhamento e teste. A seguir estão elencadas todas essas ferramentas:

- *Scrum* – metodologia de desenvolvimento ágil de projeto;
    - *Personal Scrum* – variação do método *Scrum* para um desenvolvedor;
- Linguagem UML – para modelagem dos dados;
    - Diagrama de Classes - modelo utilizado para definir a interação entre os objetos do sistema;
- *Papyrus* – ferramenta para realizar modelagem de dados do sistema;
- Sistema Operacional *Android* – plataforma de dispositivos móveis para qual será desenvolvido esse projeto;
- Linguagem Java – para codificação e teste do sistema;
- IDE *Android Studio* – para escrita da parte programática do sistema e desenvolvimento do design do mesmo;
- Sistema gerenciador de banco de dados – gênero de *software* utilizado para persistência dos dados após encerramento da execução do aplicativo;
    - *Realm* – tecnologia utilizada para armazenamento dos dados tanto no dispositivo móvel quanto em uma máquina servidora;
- Requisições Rest – forma de comunicação utilizada entre aparelho e servidor;
    - Retrofit – *software* utilizado para gerenciamento de requisições cliente servidor;
- API Google Maps – *software* utilizado para apresentação e gerenciamento das rotas e mapas apresentados no sistema.

2.2.1 Scrum

Scrum é uma metodologia de gerenciamento de equipes para desenvolvimento de projetos, principalmente para aqueles especialmente caóticos. Este método tem por base os seguintes conceitos:

- Transparência – toda informação do processo e do produto é conhecido por todos. Nenhuma informação fica restrita a apenas uma pessoa;
- Inspeção – *Scrum* prevê em suas atividades avaliações frequentes para que se seja analisado se o caminho certo para o objetivo está sendo seguido;
- Adaptação – assim como dito anteriormente, existem atividades frequentes de análise do processo e atividades, como resultado destas, a equipe e o processo estão em constante adaptação para se concluir o projeto desejado.

Porém, para que tais conceitos apareçam naturalmente e a metodologia cumpra seu papel, as pessoas no processo de desenvolvimento do projeto precisam aceitar como valores o respeito, foco, comprometimento, coragem e receptividade para com todos da equipe, para com o projeto e para com si mesmos. Esses valores são necessários justamente por *Scrum* ser uma metodologia que busca a constante adaptação e esta não é possível sem dialogo e conversa entre as pessoas envolvidas, e sem os valores salientados o diálogo entre a equipe, muito provavelmente, estará comprometido.

Para o termo equipe, em *Scrum* é possível definí-lo como a composição dos seguintes papéis:

- Dono do produto: pessoa com maior entendimento sobre as regras de negócio e detentora do conhecimento sobre os fluxos de tarefas que devem estar envolvidos no projeto. Normalmente, é uma pessoa da área fim do projeto sendo desenvolvido. Este papel é responsável por ajudar o resto da equipe a gerar “Histórias do usuário”, que são cartões com requisitos do sistema, e classificá-los de acordo com a importância de tal

História, sendo esse conjunto classificado de histórias do usuário conhecido como *Backlog* do produto;

- Equipe de desenvolvimento: recursos com conhecimento técnico para transformar os requisitos do Dono do Produto em incrementos substanciais para o projeto;
- *Scrum master*: pessoal com maior conhecimento na metodologia *Scrum* e com capacidade para resolver impedimentos que estejam atrapalhando o andamento do projeto.

Esta equipe acima definida, para execução da metodologia *Scrum,* possui as seguintes atividades a seguir:

- *Sprint* – período de tempo em que acontece o desenvolvimento do projeto e outras atividades da metodologia, normalmente esse tempo são de duas a quatro semanas;
- Planejamento da *Sprint* – ao inicio de cada *Sprint,* a equipe de desenvolvimento entra em consenso sobre o que é possível executar do *Backlog* do produto no período de tempo estipulado;
- Reunião diária – reunião entre a equipe de desenvolvimento e o *Scrum master,* na qual cada participante responde as seguintes perguntas: O que foi executado no dia anterior? O que será feito hoje? E o que está impedindo a execução das tarefas?
- Revisão da *Sprint* - ao término do tempo estipulado pela *Sprint,* a equipe de desenvolvimento apresenta um incremento no projeto e avalia se o quê foi proposto para entrega foi cumprido? O que foi bom para atingir o objetivo dessa entrega? O que não foi? E o que fazer para melhorar? (SCRUM GUIDE, 2016).

Porém como o trabalho executado neste contexto é por definição individual, foi necessária a busca de uma variação da metodologia acima descrita, encontrando-se o *Personal Scrum* ou *Scrum* para uma pessoa.

2.2.1.1 *Personal Scrum*

Essa variação de *Scrum* apresenta as seguintes mudanças (ANDREWS, 2017):

- Os três princípios base desdobram-se em:
  - Compartilhamento – publicações regulares de novos incrementos para testadores;
  - Quantificação – análise pura da quantidade de trabalho possível de ser produzida por quantidade de tempo;
  - Auto-reflexão – o desenvolvedor é responsável por todas as inspeções da metodologia;
- *Sprints* podem ser de apenas uma semana;
- As reuniões diárias passam a ser feitas através de notas feitas em papel ou vídeos;
- O *backlog* do produto é mantido pelo próprio desenvolvedor com as respostas recebidas pelos testadores do aplicativo desenvolvido e/ou por avaliação própria, ou de outros meios, de novos requisitos;

2.2.2 Linguagem *UML*

A linguagem *UML* surgiu em 1987, com a intenção de unificar as diversas linguagens de modelagem existentes e atualmente encontra-se em sua versão 2.5 (UML, 2017).

Grandes empresas que possuem sistemas de médio e grande porte utilizam-se da notação *UML* para gerar documentação de seus sistemas, visando assim a compreensão do problema, escalabilidade e manutenção. Os modelos *UML* podem garantir isso, pois apresentam um variado conjunto de diagramas que possibilitam a visualização de diversas facetas do problema analisado. Estes vários diagramas são divididos em dois grupos (BELL, 2016):

- Estruturais – diagramas utilizados para realçar características estáticas do sistema, como atributos, dependências etc;

- Comportamentais – diagramas utilizados para realçar características dinâmicas do sistema, como por exemplo, troca de mensagens ao longo do tempo.

Neste trabalho que possui um porte pequeno e apresenta o uso de uma ferramenta chamada *Realm,* que é um banco de dados orientado a objetos, serão utilizados apenas os diagramas de Classes, Casos de Uso e Diagramas de Relacionamento.

2.2.3 *Papyrus*

Papyrus é uma interface de desenvolvimento de modelos criada e mantida pelo *Eclipse.org*. que contempla vários padrões de modelagem de dados, como por exemplo, *UML* 2.5, *SysML* 1.1 & 1.4, *fUML* 1.2.1, *ALF* 1.0.1, *MARTE* 1.1 etc (PAPYRUS, 2017). No quesito *UML*, o objetivo do *software* é atender a especificação aceita pela *OMG* (*Object Management Group*) em sua totalidade, ou seja, é possível criar nessa ferramenta todos os diagramas da *UML* e contar com a validação dos diagramas gerados. Por ser uma especialização do projeto *Eclipse*, esse programa segue suas características de ser altamente configurável pelo usuário e de ser um *software* livre.

2.2.4 Padrões de Projeto

Padrões de projeto são técnicas de solução para problemas recorrentes em projetos, como por exemplo, ter apenas uma instância de uma classe gerenciadora da interface gráfica de um software (GAMMA, 2006). Tais técnicas foram desenvolvidas visando a produção de um projeto de mais fácil manutenção e legibilidade, sendo que tais características são alcançadas por meio da reutilização de código, desacoplamento e coesão garantida por tais padrões de projeto.

Dentre os vários padrões existentes, para o desenvolvimento deste projeto serão utilizados os seguintes:

2.2.4.1 Singleton

Singleton é um padrão de projeto para criação de objetos que faz com que a classe controle a criação de instâncias de si mesmo e assim garanta que apenas um objeto da classe esteja ativo durante todo o funcionamento do programa.

Um projeto pode adotar esse padrão se precisar das seguintes características:

- Quando deve existir apenas uma instância de uma classe no projeto e as outras classes possuem um método de acesso bem definido para este único objeto;
- Quando esse objeto único pode ser estendido por subclasses sem necessidade de reescrita de código nas novas classes.

No caso deste trabalho, o uso do padrão *Singleton* se dará para a classe de acesso à base de dados, por conta de evitar repetição de linhas de código para instância do objeto *Realm*, bem como para que seja possível definir apenas uma vez o “Contexto” de uso do banco de dados para todo o projeto.

2.2.4.2 Mediator

Mediator é um padrão de projeto para definir o comportamento de um objeto para que este intermedie a interação entre classes distintas, mas que precisam de ligação entre si para atender a determinada ação. Um exemplo seria pensar em uma torre de controle de um aeroporto, esta funciona como a mediadora das informações entre aviões distintos, sendo que cada avião tem seus próprios dados e forma de comunicá-los (JORNALDEV, 2016).

Para este projeto, o padrão *Mediator* será utilizado para gerenciar a comunicação entre as classes de exibição e as que implementam a ação desejada pelo usuário.

2.2.5 Sistema Operacional *Android*

Inicialmente o sistema operacional *Android* foi desenvolvido pela empresa *Android Inc.* que tinha apoio financeiro da empresa *Google*, porém, futuramente, acabou por ser adquirida por completo por esta (ANDROID, 2017).

É possível de se dizer que o *Android* é o sistema operacional para dispositivos móveis mais utilizados no mundo (STATISTA, 2017) e tal sucesso pode ser atribuído a diversos fatores, dentre alguns:

- Licença Apache – o código do sistema operacional é aberto, assim toda a comunidade pode contribuir e personalizar o sistema a seu modo;
- *Kernel Linux* – base do sistema desenvolvida usando como fundação o *Linux* 2.6.4, recebendo assim todas as vantagens de um *Kernel Linux* (Burnette, 2008);
- Linguagem Java – embora a base do sistema seja *Linux* e tenha suas bibliotecas nas linguagens C e C++, os desenvolvedores podem utilizar a linguagem Java para o desenvolvimento de programas e contar com uma máquina virtual otimizada para aparelhos móveis que é a Dalvik;
- *Google* – principal empresa mantedora do sistema e que por ter uma abrangência global, dentre outro motivos pelo grande número de aparelhos com *Android*, busca sempre ter um sistema de qualidade e acessível;
- Loja de aplicativos – pelas facilidades oferecidas aos desenvolvedores para gerar aplicativos para o sistema *Android*, por conta da linguagem Java e recursos do sistema, a loja de aplicativos dessa plataforma é abundante (STATISTA, 2017) em relação à de suas concorrentes. Isso, por muitas vezes é fator decisório para o usuário final, pois caso precise de um aplicativo para ajudá-lo em determinada atividade, sabe que acessando a loja de aplicativos, poderá encontrar algum que atenda parcialmente ou até completamente suas necessidades.

E são por esses fatores acima listados, que este sistema operacional foi escolhido como plataforma alvo deste trabalho.

2.2.6 Linguagem Java

Java é uma linguagem de programação orientada a objetos e mantida pela empresa *Oracle* (JAVA, 2017). É uma linguagem amplamente utilizada no mercado para desenvolvimento das mais diversas aplicações, e como visto acima, o sistema operacional *Android* conta com uma máquina virtual otimizada e um vasto conjunto de bibliotecas para esta linguagem, fazendo desta a linguagem padrão para desenvolvimento de aplicativos para esta plataforma.

2.2.7 IDE(Integrated Development Environment) Android Studio

Anteriormente ao lançamento deste *software*, a principal ferramenta para os desenvolvedores de aplicativos para *Android* era uma extensão para a *IDE Eclipse* chamada de *Eclipse ADT* (*Android Development Tools*), porém a equipe do *Google* que mantinha tal extensão optou por encerrar o suporte (ANDROID STUDIO, 2015) ao *software* e assim, em 2013, lançaram o *Android Studio* (ANDROID STUDIO, 2017). Baseado na ferramenta IntelliJ IDEA (JETBRAINS, 2017) da empresa *JetBrains*, esse ambiente de desenvolvimento integra várias ferramentas úteis para o desenvolvimento de aplicativos, como:

- Emulador – módulo que cria um dispositivo virtual, sendo possível escolher a versão destino do sistema operacional *Android* e com diversas opções de configurações e ajustes para deixar a simulação o mais fiel possível;
- *Gradle* – ferramenta que gerencia as dependências de *software* da aplicação gerada e também as opções de compilação, facilitando o momento de construção da versão executável do aplicativo;
- Execução imediata – quando usando um emulador ou um dispositivo conectado diretamente ao computador para teste do aplicativo, é possível aplicar as alterações de código diretamente ao *software* em execução sem necessidade de uma nova compilação e encerramento da execução atual, isso através da opção “*Instant Run*”.

2.2.8 Sistema Gerenciador de Banco de Dados (SGBD)

Na aplicação desenvolvida, é de extrema importância o armazenamento de dados para posterior consulta, seja em relatórios simples ou processamentos mais complexos, para tanto será necessário um banco de dados e um sistema para geri-lo (ARAÚJO, 2017). Programas que executam tal gerenciamento são conhecidos no ramo da computação como sistemas gerenciadores de bancos de dados. Sistemas dessa classe costumam atendem a certos requisitos para garantir segurança, integridade e acesso aos dados, bem como apresentam diversas variações de acordo com a forma de modelagem dos dados que esses sistemas devem armazenar, por exemplo, sistemas gerenciadores de bancos de dados relacionais, dimensionais, orientados a objetos etc. Atualmente, há no mercado várias opções desses *softwares* para serem utilizadas, porém neste trabalho, será utilizada a ferramenta *Realm*, por conta das características apresentadas a seguir.

2.2.8.1 *Realm*

Realm é uma solução de *software* para gerenciamento de banco de dados orientado a objetos, construída para atender as novas necessidades de desenvolvimento de aplicativos, que seriam (REALM, 2017):

- Sincronização de dados - dados gerados em um dispositivo móvel, possivelmente são enviados para algum servidor, para tratamento, armazenamento e distribuição;
- Funcionamento *off-line* – como os usuários de dispositivos móveis podem entrar ou sair de áreas de cobertura de redes de celulares, sofrerem de congestionamento da comunicação ou qualquer outro fator que afete a conectividade do dispositivo com a rede, o funcionamento *off-line* de aplicativos é um novo requisito necessário;
- Multiplataforma - a grande variedade de marcas e modelos aparelhos móveis disponíveis no mercado, faz com que seja necessário aos desenvolvedores pensar e testar seus aplicativos para vários

dispositivos diferentes, para que assim possam contemplar o maior número de usuários.

Para atender a tais necessidades, o sistema conta com o módulo de banco de dados móvel *Realm* que oferece as seguintes funcionalidades:

- Modelo de dados entre plataformas – Como *Realm* se baseia na definição da classe para gerar os objetos e armazená-los tanto no servidor quanto nos vários dispositivos móveis (*Android* ou *IOs*), ele garante que o modelo de dados é o mesmo em todos os lugares;
- Tratador de eventos – para atender a algumas regras de negócio do *software* sendo desenvolvido, é necessário que assim que um dado tenha algum tipo de alteração, seja realizado um processamento no servidor e distribuído entre os dispositivos móveis, *Realm* garante isso através da codificação de gatilhos no servidor de objetos que respondem as variações de dados;
- *Backup* – como visto acima, um dos requisitos para SGBDs é a segurança dos dados e para atendê-lo, o servidor de objetos permite a configuração de um servidor réplica em um local completamente diverso do servidor principal, para casos de emergenciais;
- Integração de dados – para alguns aplicativos é necessária a integração com sistemas legados de alguma empresa, para contemplar esses casos a ferramenta oferece conectores para o servidor de objetos para o acesso e sincronia dos dados;
- Segurança dos dados – para garantir a sincronização dos dados, é necessário que exista um servidor atendendo requisições e que dispositivos enviem dados pela rede, porém para que não exista interceptação de dados ou uso indevido do servidor e seus dados, é necessário algum tipo de segurança. No caso de *Realm*, para garantir tal segurança é utilizado encriptação dos dados pelo algoritmo *AES*-256, além de outros padrões.

Assim, pela ferramenta oferecer as características acima, ser de fácil utilização e aceitar várias linguagem de programação, dentre as quais Java, ela foi escolhida como o SGBD para este trabalho. Ressalva-se, porém, que seu uso será

apenas através do módulo de banco de dados móvel, pois neste trabalho não foi compreendida a criação de servidores de dados.

2.2.9 Requisições *Rest*

Software como serviço é um conceito que já está entre a comunidade de tecnologia de informação há alguns anos e hoje existem grandes empresas como *Google*, *Amazon* etc, que provêm vários serviços online e vários destes podendo ser úteis e necessários para novas ideias de aplicativos para dispositivos móveis (CÉSAR, 2017).

Para que tais serviços sejam consultados e consumidos por aplicativos é necessário uma forma de requisitá-los, dentre as disponíveis para utilização, a que será empregada neste trabalho será a forma *REST (Representational state transfer)*.

REST funciona sobre o protocolo *HTTP* e foi escolhida pelos seguintes motivos (NORDIC APIS, 2017):

- Facilidade de uso;
- Dados podem ser retornados em vários formatos (*XML*, *JSON* etc);
- Alto consumo de dados da outra forma de requisição, *SOAP*.

Para gerenciar as requisições *REST* e seus retornos, desenvolvedores de aplicativos contam com uma ferramenta que será utilizada neste trabalho, conhecida como Retrofit.

2.2.9.1 Retrofit

Para realizar requisições *REST*, a plataforma *Android* regulamenta que seja utilizada uma chamada assíncrona ao serviço necessário, ou seja, o aparelho não ficará “travado” esperando a resposta para o serviço (RETROFIT, 2017). Porém, toda tratativa de definição da chamada assíncrona é por conta do desenvolvedor, o que gera uma grande quantidade de código e, dependendo do nível de conhecimento do desenvolvedor, pode não ter um desempenho satisfatório, tornando o aplicativo lento e ruim para o usuário final.

Assim, o Retrofit é apresentado como uma ferramenta que facilita requisições *REST*, pois gerencia as chamadas assíncronas e seu desempenho, fazendo com que seja necessário apenas definir uma Interface Java para especificação dos serviços utilizados. A seguir, na figura 4, é apresentada a codificação de *Interface* para o uso do *Retrofit.*

```
public interface GitHubService {
  @GET("users/{user}/repos")
  Call<List<Repo>> listRepos(@Path("user") String user);
}
```

Figura 4 - Interface Java Retrofit

Ainda com o Retrofit, no momento de criação da instanciação do serviço de consulta *REST*, é possível associá-lo a um analisador de *JSON* para que, assim que a requisição seja feita, o retorno da mesma seja tratado e transformado em objetos Java. No próprio site da ferramenta são sugeridos vários analisadores, para este trabalho será utilizado o *Google-GSON* (GSON, 2017), analisador fornecido pela *Google* e que, dentre outros detalhes, não necessita de anotações em código para seu funcionamento. Um exemplo de instanciação do *Retrofit* é possível de ser visto na figura 5.

```
Retrofit retrofit = new Retrofit.Builder()
    .baseUrl("https://api.github.com/")
    .build();

GitHubService service = retrofit.create(GitHubService.class);
```

Figura 5 - Instância Retrofit

2.2.10 *API Google Maps*

O serviço de mapas do *Google* já é uma ferramenta consolidada no mercado para fornecimento de informações relacionadas a localização (GOOGLE DEVELOPERS, 2017). Além do aplicativo de mapas, a empresa fornece serviços relacionados (consultados via *REST*, como visto acima) e uma *API* (*Application Programming Interface*) para ser utilizada por desenvolvedores para que seus aplicativos incorporem e utilizem-se do *Google Maps*. Com esta ferramenta é possível vários níveis de personalização e utilização e os que serão usados neste trabalho serão:

- Inclusão de marcadores – no mapa é possível adicionar uma imagem para marcar um local em especial. Neste trabalho será utilizado para marcar um possível cliente que é necessário fazer uma entrega;
- Rotas – adição de caminhos que ligam um ponto a outro do mapa, levando em conta a forma de locomoção utilizada e trânsito. Tal função é necessária para estre trabalho por conta da apresentação de todos os clientes necessários a se visitar e as condições de trânsito entre eles.

2.2.11 Biblioteca de *Data Binding*

No âmbito de desenvolvimento para sistema *Android* utilizando a linguagem Java, normalmente, é utilizado algo como o trecho de código a seguir (figura 6) para o acesso aos elementos da interface com o usuário:

```
46      EditText editText;
47      editText = (EditText) view.findViewById(R.id.nomeInfoProdutoFragment);
48      editText.getText().toString();
49
```

Figura 6 - Exemplo de acesso a elemento da tela sem *Data Binding*

Para evitar que esse trecho de código seja repetido diversas vezes, de acordo com a quantidade de elementos da tela a serem acessados nas diversas classe que necessitam dos elementos da interface, os desenvolvedores podem

utilizar a biblioteca *Data Binding* (GOOGLE DEVELOPERS, 2017) fornecida pela própria *Google.* Tal solução foi escolhida por ser de fácil utilização e configuração, sendo necessárias apenas algumas simples alterações do arquivo de dependências do aplicativo, layout da tela e do próprio arquivo que fará o acesso aos elementos da tela. A seguir na figura 7, a titulo de comparação, é apresentado como é feito o acesso ao mesmo elemento da interface da imagem anterior utilizando-se da biblioteca:

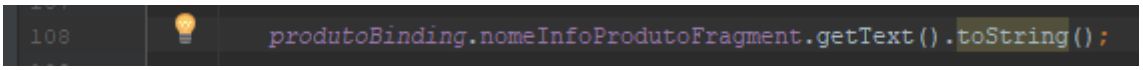


Figura 7 - Exemplo de acesso a elemento da tela com Data Binding

2.2.12 Biblioteca *PdfDocument*

A apresentação de dados para um ambiente externo ao do sistema sendo construído é um de seus requisitos básicos, uma das formas para que esta exposição de dados seja feita é através da geração de relatórios e para auxiliar nesta tarefa, o *Android* dispõe de uma biblioteca nativa chamada *PdfDocument*(GOOGLE DEVELOPERS, 2017).

Com o uso desta ferramenta, basta criar a aparência do relatório da mesma forma que seria criada qualquer tela de algum aplicativo em *Android,* podendo até mesmo ser utilizado a biblioteca *DataBinding* citada anteriormente e com algumas configurações de página, tal relatório já estará disponível para ser apresentado, compartilhado etc.

2.2.13 Usabilidade

No contexto de desenvolvimento de software é importante a garantia de que aquilo que foi criado atende, melhora e/ou satisfaz a necessidade de um conjunto de usuários. Dentre as várias formas de se medir que determinado software garante esses requisitos, escolheu-se para esse trabalho a medida de Usabilidade.

Conforme retratado em GATTO (2017), o termo “Usabilidade” é definido na norma ISSO-9241-11 da Organização Internacional de Normalização (*ISO*) e trato pela Associação Brasileira de Normas Técnicas na norma NBR-9241-11. Neste documento técnico, usabilidade é definida como:

“Medida na qual um produto ser usado por usuários específicos para alcançar objetivos específicos com: eficácia, eficiência e satisfação em um contexto específico de uso.”

Sobre a citação acima, FERREIRA e DRUMOND (2002) definem:

- Usuário: pessoa que interage com o produto;
- Contexto de uso: usuários, tarefas, equipamentos (hardware, software e materiais), ambiente físico e social em que o produto é usado;
- Eficácia: precisão e completeza com que os usuários atingem objetivos específicos, acessando a informação correta ou gerando os resultados esperados;
- Eficiência: precisão e completeza com que os usuários atingem seus objetivos, em relação à quantidade de recursos gastos;
- Satisfação: conforto e aceitabilidade do produto, medidos por meio de métodos subjetivos e/ou objetivos.

Assim, para a realização de um teste de usabilidade é necessária a criação de vários contextos de tal forma que o conjunto criado abranja a maioria, senão todas as funcionalidades do produto sendo testado.

2.2.13.1 System Usability Scale

Ainda, para que seja possível comparar os testes realizados no aplicativo sendo desenvolvido, é importante o uso de alguma escala reconhecida, e dentre as várias disponíveis optou-se para esse projeto a *System Usability Scale* (SAURO, 2011).

Esta escala é composta por um questionário de dez perguntas feitas ao usuário após a realização das tarefas definidas no teste de usabilidade, tais questões versam sobre a satisfação do usuário ao utilizar o sistema e as questões utilizadas neste trabalho serão:

- Eu acho que gostaria de usar esse sistema com frequência;
- Eu acho o sistema desnecessariamente complexo;
- Eu achei o sistema fácil de usar;
- Eu acho que precisaria de ajuda de uma pessoa com conhecimentos técnicos para usar o sistema;
- Eu acho que as várias funções do sistema estão muito bem integradas;
- Eu acho que o sistema apresenta muita inconsistência;
- Eu imagino que as pessoas aprenderão como usar esse sistema rapidamente;
- Eu achei o sistema atrapalhado de usar;
- Eu me senti confiante ao usar o sistema;
- Eu precisei aprender várias coisas novas antes de conseguir usar o sistema.

Cada pergunta é respondida com um valor entre um e cinco, sendo o valor “um” um indicativo de que a pessoa discorda totalmente da afirmação e “cinco” que concorda plenamente com o afirmado pela questão. Com base nessas respostas, para que seja possível aferir o grau de usabilidade segundo *SUS,* utilizam-se as seguintes regras:

- Para perguntas ímpares, subtrair um da resposta dada pelo avaliador;
- Para perguntas pares, fazer a seguinte conta 5 – número da resposta dada pelo avaliador;
- Somar todos os números obtidos;
- Multiplicar o resultado da soma obtido por 2.5.

Após o cálculo, é possível obter uma referência quantitativa sobre a usabilidade do produto sendo avaliado e comparar tal referência com a imagem apresentada na figura 8 para verificação do resultado obtido.

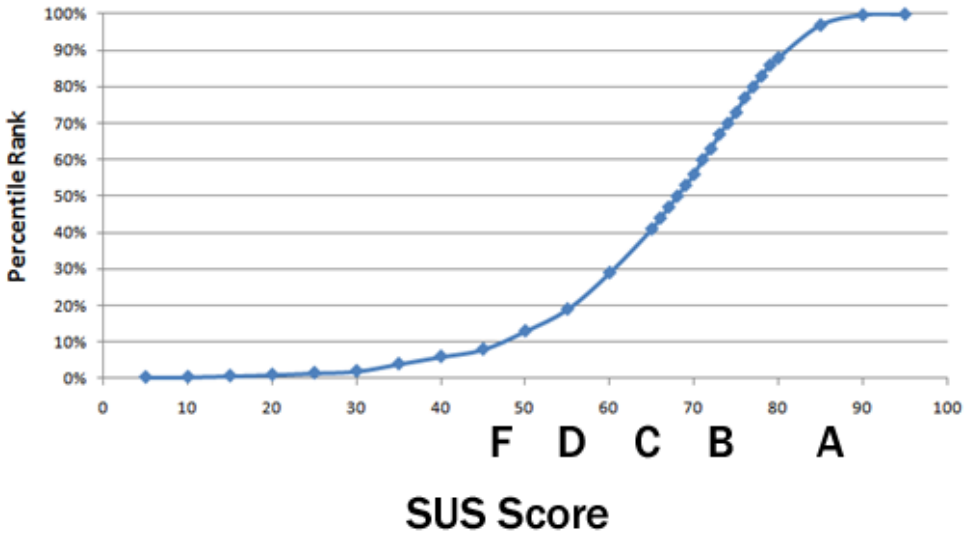


Figura 8 - Tabela de referência da *SUS*

Na figura 8, o valor quantitativo obtido deve ser encontrado no eixo “SUS Score” para ser possível verificar o nível de usabilidade em percentagem que o sistema avaliado está oferecendo para o usuário, sendo que, segundo SAURO (2011), sistemas com valores abaixo de 70 pontos no eixo “SUS Score”, estão com um baixo nível de usabilidade que necessita revisão por parte dos criadores do mesmo.

3 DESENVOLVIMENTO

Nos capítulos anteriores foram estabelecidas as bases para o desenvolvimento do aplicativo desejado, assim, neste capítulo serão apresentados passos e marcos relevantes na produção de tal software. Para que tais pontos sejam apresentados de forma mais completa, levando em consideração as várias *sprints* do projeto, as subseções a seguir terão seus respectivos artefatos descritos várias vezes, isto para que seja possível analisar o desenvolvimento e amadurecimento de determinado artefato ao longo da execução do projeto.

3.1 Levantamento de requisitos

Para o levantamento de requisitos optou-se pelo uso de Histórias de Usuário, pois descrevem comportamentos esperados do sistema e são de fácil produção/entendimento por parte do Dono do Produto (*Product owner*). A seguir, serão apresentadas várias versões do *backlog* do produto de acordo com as suas alterações durante o desenvolvimento do projeto. A figura 9 a seguir, apresenta a versão do artefato na *sprint 2.*

Tema	História do Usuário	Estimativa	Prioridade
Produto	Como administrador quero cadastrar produtos para possuir uma listagem dos mesmos e poder gerenciá-los	7h	Alta
Produto	Como administrador quero alterar o cadastro de um produto, podendo excluir ou atualizar dados para manter meus dados atualizados	7h	Alta
Produto	Como administrador quero cadastrar foto para um produto para ser possível formar um catálogo mais significativo para ser passado a meus clientes	5h	Alta
Estoque	Como administrador gostaria de acompanhar meus produtos em estoque e ver a variação de quantidade de itens por período	5h	Média
Pedido	Como entregador gostaria de acompanhar os pedidos para entrega no meu dia de trabalho e a melhor rota para entregá-los para assim fazer entregas eficientes.	15h	Média
Produto	Como administrador desejo ver meus produtos cadastrados e quais são os possíveis fornecedores, em caso de apenas um, alerta-me, assim terei informações para buscar novos fornecedores ou contactar os que já possuo	5h	Média
Vendedor	Como administrador quero cadastrar vendedores para possuir uma listagem dos mesmos e poder gerenciá-los	7h	Média
Vendedor	Como administrador quero alterar o cadastro de um vendedor, podendo excluir ou atualizar dados para manter meus dados atualizados	7h	Média
Cliente	Como administrador/vendedor desejo visualizar meu clientes num mapa para ter melhor noção de suas localizações	9h	Baixa
Fornecedor	Como administrador quero cadastrar fornecedores utilizando informações da minha agenda do celular para facilidade de cadastro	8h	Baixa

Figura 9 - *Backlog* do Produto na *sprint* 2

Neste trecho do *backlog* do produto é possível verificar que além da história do usuário em si existem outras colunas com informações auxiliares. Aqui será tomada como exemplo a linha da história “Como administrador quero cadastrar produtos para possuir uma listagem dos mesmos e poder gerenciá-los”, nesta é possível visualizar que tal história foi categorizada como do tema “Produto”, tal agrupamento foi feito para facilitar a visualização de qual faceta do sistema geral está sendo contemplada pela história do usuário em questão. Ainda nesta linha é possível notar que há uma estimativa de horas de trabalhos necessárias para atender a tal história e também uma coluna indicando a prioridade que esta história deve ter em relação às outras, em vista da importância desta para o sistema.

Na figura 10 é apresentada a progressão deste artefato na *sprint* 10.

Tema	História do Usuário	Estimativ	Prioridade	Atendido
Pedido	Como vendedor desejo ver um relatório com os itens solicitados por um cliente em um pedido para assim apresentá-lo de forma impressa e/ou enviá-lo por email	10h	Alta	Parcial
Produto	Como administrador quero cadastrar produtos para possuir uma listagem dos mesmos e poder gerenciá-los	7h	Alta	Feito
Produto	Como administrador quero alterar o cadastro de um produto, podendo excluir ou atualizar dados para manter meus dados atualizados	7h	Alta	Feito
Produto	Como administrador quero cadastrar foto para um produto para ser possível formar um catálogo mais significativo para ser passado a meus clientes	5h	Alta	Feito
Estoque	Como administrador gostaria de acompanhar meus produtos em estoque e ver a variação de quantidade de itens por período	5h	Média	
Pedido	Como entregador gostaria de acompanhar os pedidos para entrega no meu dia de trabalho e a melhor rota para entregá-los para assim fazer entregas eficientes.	15h	Média	
Produto	Como administrador desejo ver meus produtos cadastrados e quais são os possíveis fornecedores, em caso de apenas um, alerta-me, assim terei informações para buscar novos fornecedores ou contactar os que já possuo	5h	Média	
Vendedor	Como administrador quero cadastrar vendedores para possuir uma listagem dos mesmos e poder gerenciá-los	7h	Média	
Vendedor	Como administrador quero alterar o cadastro de um vendedor, podendo excluir ou atualizar dados para manter meus dados atualizados	7h	Média	
Cliente	Como administrador/vendedor desejo visualizar meu clientes num mapa para ter melhor noção de suas localizações	9h	Baixa	
Fornecedor	Como administrador quero cadastrar fornecedores utilizando informações da minha agenda do celular para facilidade de cadastro	8h	Baixa	Feito

Figura 10 - Backlog do Produto na sprint 10

Na figura 10, está representada a estrutura do *backlog* do produto que, na *sprint* de número dez, teve como mudança a inclusão da coluna “Atendido”, sendo que a utilidade de tal campo é representar o andamento de determinado requisito no projeto. Como exemplo, é possível verificar que o requisito “Como vendedor desejo ver um relatório com os itens solicitados por um cliente em um pedido para assim apresenta-lo de forma impressa e/ou enviá-lo por email” do tema “Pedido”, durante a *sprint* dez, estava parcialmente atendido.

3.2 Design

A seguir serão apresentados os artefatos gerados no design do comportamento do software proposto.

3.2.1 Casos de Uso

O primeiro caso de uso apresentado será o da figura 11, que representa o momento de cadastro da venda por um vendedor.

3.2.1.1 Caso de Uso - Cadastrar venda

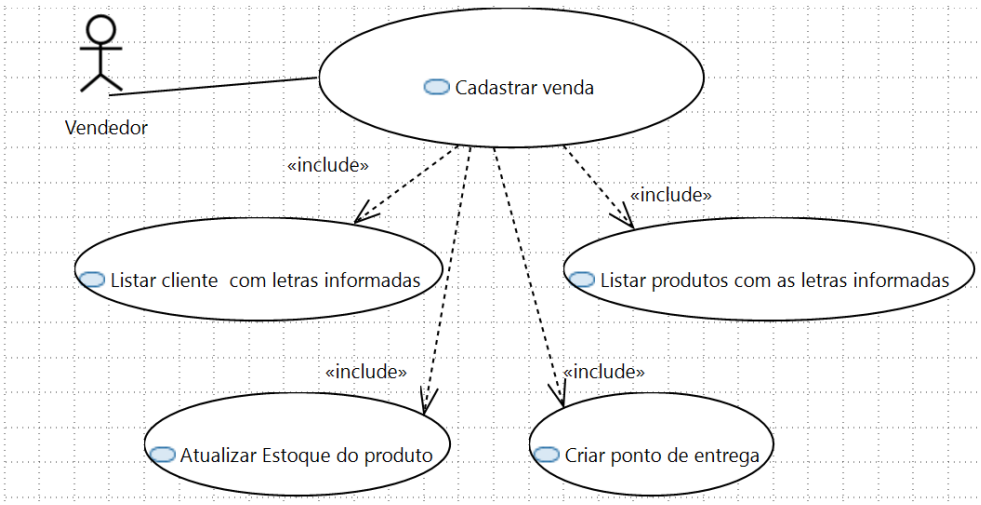


Figura 11 - Caso de Uso - Cadastrar Venda

Nome do Caso de Uso: Cadastrar venda.

Descrição: O sistema apresenta a tela de cadastro de vendas, na qual usuário deve informar o nome de cliente e produto previamente cadastrados, juntamente com as informações de quantidade e preço unitário do produto. Caso exista cadastro de estoque para o produto informado, o sistema atualiza quantidade em estoque automaticamente e caso, no cadastro do cliente exista um endereço, sistema automaticamente gera um ponto de entrega .

Eventos:

- Sistema apresenta tela de cadastro de vendas.

- Sistema apresenta sugestão de cliente cadastrado com base nas letras iniciais informadas pelo vendedor.

- Sistema apresenta sugestão de produto cadastrado com base nas letras iniciais informadas pelo vendedor.

- Vendedor informa quantidade e preço unitário para o item desejado.

- Sistema grava informação da venda

- Sistema gera ponto de entrega para ser apresentado o mapa de rota de entregas.

Atores:

- Vendedor

Pré-Condições:

- Existir cadastro do produto desejado.

-Existir cadastro de cliente desejado.

Pós-Condições:

1. Conclusões com sucesso:

- Histórico de venda gravado.

2. Conclusões sem sucesso:

- Apresentação de mensagem de erro informando falha ao gravar a venda.

Fluxo básico:

1. Sistema apresenta a tela de cadastro de venda;
2. Vendedor informa as primeiras letras do nome do cliente;
3. Vendedor seleciona o cliente da lista sugerida pelo sistema;
4. Vendedor informa dados da venda;
5. Vendedor informa as primeiras letras do nome do produto;
6. Vendedor seleciona o produto da lista sugerida pelo sistema;
7. Vendedor informa a quantidade e o valor unitário do produto selecionado;

8. Vendedor pressiona botão para gravar venda;
9. Sistema grava venda; (A2)
10. Sistema atualiza estoque do produto;
11. Sistema gera ponto de entrega;
12. Caso de uso é finalizado;

Fluxos alternativos:

A1. Sistema tenta gravar venda com informações fornecidas;

A1.2 Sistema encontra erro ao tentar gravar alguma informação;

A1.3 Sistema apresenta mensagem em tela informando falha ao gravar venda;

A1.4 Sistema encerrar o fluxo de cadastro de venda e fecha tela de cadastro.

3.2.1.2 Caso de Uso - Cadastrar estoque

O próximo caso de uso apresentado será o de cadastro de estoque, ilustrado na figura 12.

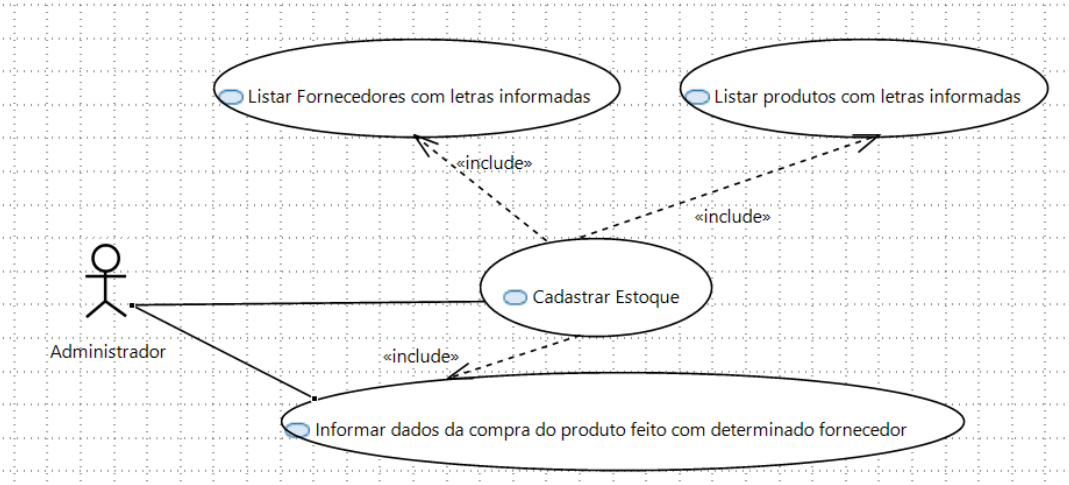


Figura 12 - Caso de uso - Cadastrar Estoque

Nome do Caso de Uso: Cadastrar estoque.

Descrição: O sistema apresenta a tela de cadastro de estoque, na qual usuário deve informar o nome de produto e fornecedor previamente cadastrados, juntamente com as informações de quantidade e preço unitário do produto adquirido de determinado fornecedor.

Eventos:

- Sistema apresenta tela de cadastro de estoque;
- Sistema apresenta sugestão de produto cadastrado com base nas letras iniciais informadas pelo vendedor;
- Sistema apresenta sugestão de fornecedor cadastrado com base nas letras iniciais informadas pelo vendedor;
- Administrador informa quantidade e preço unitário para o produto adquirido;
- Sistema grava informação do estoque;

Atores:

- Administrador.

Pré-Condições:

- Existir cadastro do produto desejado;
- Existir cadastro de fornecedor desejado.

Pós-Condições:

1. Conclusões com sucesso:

- Estoque gravado.

2. Conclusões sem sucesso:

- Apresentação de mensagem de erro informando falha ao gravar a estoque.

Fluxo básico:

1. Sistema apresenta a tela de cadastro de estoque;
2. Administrador informa as primeiras letras do nome do produto;
3. Administrador seleciona o produto da lista sugerida pelo sistema;

4. Administrador informa as primeiras letras do nome do fornecedor;
5. Administrador seleciona o fornecedor da lista sugerida pelo sistema;
6. Administrador informa a data da compra, quantidade e o valor unitário do produto selecionado;
7. Administrador pressiona botão para gravar estoque;
8. Sistema grava estoque; (A2)
9. Caso de uso é finalizado;

Fluxos alternativos:

A1. Sistema tenta gravar estoque com informações fornecidas;

A1.2 Sistema encontra erro ao tentar gravar alguma informação;

A1.3 Sistema apresenta mensagem em tela informando falha ao gravar estoque;

A1.4 Sistema encerrar o fluxo de cadastro de estoque e fecha tela de cadastro.

3.2.1.3 Caso de Uso – Listar pontos de entrega.

O último caso de uso apresentado será o de listagem de pontos de entrega, que é possível de ser visualizado na figura 13.

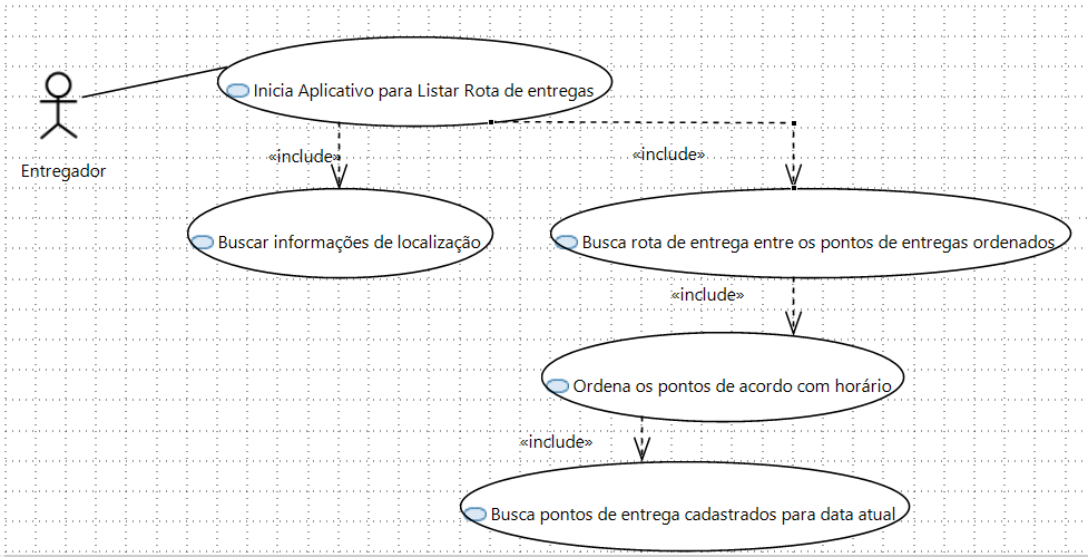


Figura 13 - Listar Pontos de Entrega

Nome do Caso de Uso: Listar pontos de entrega.

Descrição: Ao ser iniciado o aplicativo, sistema apresenta diretamente um mapa, centralizado na região do entregador. No mapa apresentado, sistema irá encontrar os pontos de entrega com data para realização da entrega igual a data atual do sistema. Após encontrar os pontos, sistema irá numerá-los e encontrar uma rota para visita-los na ordem crescente desta numeração. Após todo esse processamento, sistema irá apresentar os pontos e a rota no mapa da tela apresentada.

Eventos:

- Aplicativo é iniciado;
- Sistema apresenta mapa na tela;
- Sistema busca informação da localização do entregador;
- Sistema centraliza mapa;
- Sistema de busca pontos de entrega para o dia atual de acesso ao aplicativo;

- Sistema numera pontos;
- Sistema encontra rota de entrega entre os pontos encontrados;
- Sistema apresenta os pontos no mapa juntamente com rota ligando-os.

Atores:

- Entregador.

Pré-Condições:

- Existir conexão com internet;
- Existir permissão concedida para obter dados de informação da localização do usuário do aplicativo;
- Existir permissão concedida para utilização de internet;
- Existir pontos de entrega cadastrados.

Pós-Condições:

1. Conclusões com sucesso:

- Rota de entrega entre pontos apresentada.

2. Conclusões sem sucesso:

- Mapa apresentado sem pontos de entrega;

Fluxo básico:

1. Aplicativo é iniciado pelo Entregador;
2. Sistema apresentada na tela um mapa;
3. Sistema busca informações sobre localização do Entregador; (A1)
4. Sistema centraliza mapa com base nas informações obtidas;
5. Sistema busca informações de pontos de entregas cadastrados, filtrados de acordo com a data de entrega igual a data atual de acesso ao sistema; (A2)
6. Sistema numera os pontos obtidos em ordem crescente de acordo com a data de entrega;
7. Sistema calcula rota de entrega entre os pontos, levando em consideração a numeração estabelecida; (A3)
8. Sistema apresenta pontos e rota no mapa da tela.

9. Caso de uso é finalizado;

Fluxos alternativos:

A1. Sistema tenta buscar informações de localização do entregador;

A1.2 Não há permissão concedida para acessar tal informação ou falha ao obtê-la;

A1.3 Sistema procede no fluxo no passo 5.

A2. Sistema tentar buscar pontos cadastrados;

A2.1. Sistema não encontra pontos cadastrados;

A2.2 Sistema procede no fluxo no passo 9;

A3. Sistema tenta calcular rota entre pontos;

A3.1. Sistema não possuí permissão de acesso a internet ou não há conectividade.

A3.2 Sistema procede no fluxo no passo 9;

Os casos de uso apresentados são os que compreendem as funcionalidades mais importantes dos sistemas, por isso optou-se pela sua apresentação neste trabalho, para auxílio da explanação do aplicativo pretendido.

3.2.2 Modelagem de dados

Como dito no Capítulo dois, por conta do uso de um banco de dados orientado a objetos, a modelagem de dados foi feita e será apresentada por meio de diagrama de classes e diagrama entidade relacionamento. A seguir, é possível acompanhar o desenvolvimento de ambos.

3.2.2.1 Diagrama de Classes

Na figura 14 é apresentada uma versão inicial do diagrama de classes.

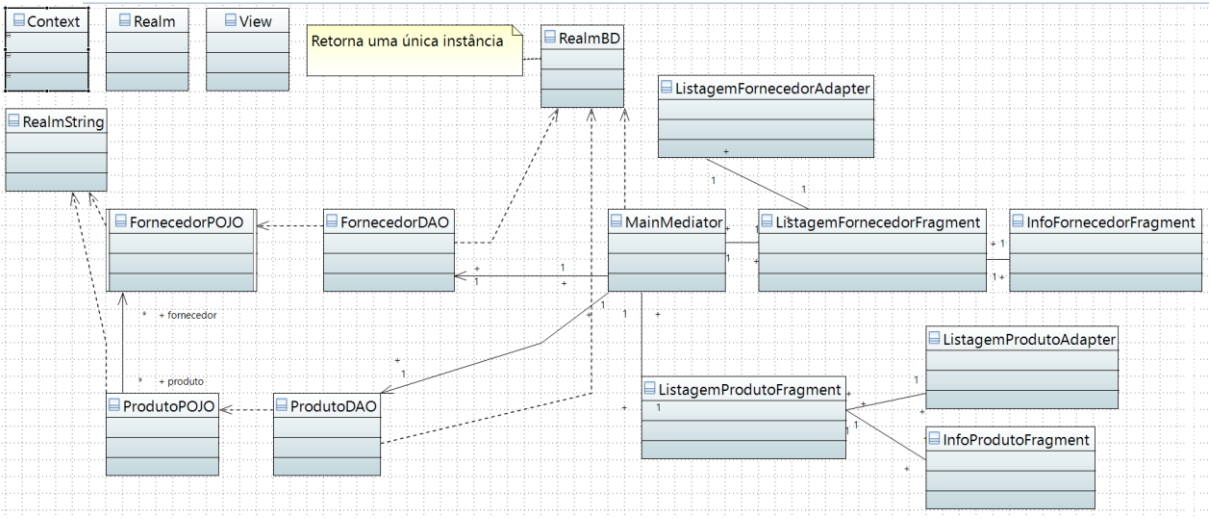


Figura 14 - Diagrama de classes *sprint 2*

Nesta versão buscou-se explicitar o uso dos padrões de projeto *Mediator* e *Singleton* através das classes *MainMediator* e *RealmBD*, respectivamente, sendo que a primeira está responsável por receber as mensagens das classes de apresentação (*Fragments*) e entregá-las às devidas classes responsáveis por tratar a ação desejada (classes de acesso ao objeto). A classe *RealmBD*, busca atender ao requisito de *Singleton* através de seu atributo *instance* privado e o método de acesso *getInstance()* que faz o controle para que exista apenas uma instância ativa. Ainda, no diagrama são apresentadas duas classes que correspondem a entidades bases do sistema, que são:

- *FornecedorPOJO*: classe que, no momento da s*print 2,* possui os atributos de nome, telefone e endereço de algum fornecedor de produtos;

- *ProdutoPOJO*: classe que, no momento da s*print 2,* possui os atributos de nome, unidade (unidade, grama, litros etc), peso e possíveis fornecedores.

Aqui é possível notar que o relacionamento entre estas duas classes se dá da forma que um produto pode ser fornecido por nenhum, um ou vários fornecedores, assim é possível classificar tal associação como uma associação de agregação “um para n”, ou seja, um produto para zero ou mais fornecedores.

Por conta de visibilidade a evolução do diagrama de classes é apresentada na seção de Apêndices. Na *sprint* dez, este artefato passou a conter as classes:

- *FornecedorPOJO:* mesma classe apresentada na versão deste artefato para *sprint 2* com adição de atributo de e-mail;
- *ProdutoPOJO:* alterada classe apresentada na *sprint 2* para conter imagem do produto e removido atributo que armazenava possíveis;
- *ClientePOJO:* classe que, até a *sprint* 10, contém os seguintes atributos de algum cliente do sistema: nome, telefones, e-mails, endereços, razão social e número de documento (atributo para armazenar cpf ou cnpj);
- *EstoquePOJO:* classe que no momento da *sprint* 10 possuia os seguintes atributos:
  - *produto:* objeto da classe *ProdutoPOJO;*
  - quantidade: quantidade em estoque de determinado produto
  - compras: conjunto de objetos da entidade de ligação *FornecedorEstoquePOJO* descrita a seguir;
- *FornecedorEstoquePOJO:* classe de ligação entre as entidades *Estoque* e *Fornecedor,* que durante a *sprint 10* possui os seguintes atributos relevantes do relacionamento entre as duas entidades citadas:
  - *fornecedor:* objeto da classe *FornecedorPOJO* que representa de qual fornecedor foi feito compra do produto em estoque;
  - quantidade: quantidade do produto em estoque adquirido deste *fornecedor;*

    - valor unitário: preço pago pelo produto em estoque de determinado fornecedor;
    - data compra: data quem que o produto em estoque teve um compra efetuado com determinado fornecedor.
- *VendaPOJO*: na execução da *sprint 10* esta classe possui os seguintes atributo relacionados a uma venda:
    - *cliente:* objeto da classe *ClientePOJO* que é usado para indicar qual cliente está envolvido em determinada venda;
    - orçamento*:* campo utilizado para indicar se determinada venda será apenas um orçamento e caso positivo, não irá alterar estoque de produtos quando venda for armazenada no banco de dados;
    - observação: campo livre para indicação de qualquer característica relevante relativa a determinada venda;
    - data venda: atributo gerado automaticamente quando venda é armazenada representando exatamente o este horário em que a entidade foi salva no banco de dados;
    - data pagamento: atributo que indica se pagamento da venda ainda está pendente, caso indique uma data futura;
    - data entrega: atributo indicando se determinada venda ainda deve ser entregue ao cliente, caso data indicada seja uma data futura;
    - produtos: atributo que é um conjunto de de objetos da classe *ProdutoVendaPOJO* definida a seguir;
- *ProdutoVendaPOJO:* classe de que armazena os dados relevantes entre o relacionamento das entidades *Produto* e *Venda*, sendo que tais dados são:
    - *produto:* instância de objeto da classe *ProdutoPOJO* que presenta o objeto participante da venda;
    - quantidade*:* quantidade de produtos reservados para determinada venda;
    - *valorUnitario:* valor unitário deste produto para determinada venda;

Nesta versão do artefato é possível verificar que o relacionamento entre as entidades *Fornecedor e Produto* foi removido, visto que a relação de fato acontecia no momento de geração de estoque de determinado produto e como foi analisado que o relacionamento entre as entidades *Estoque e Fornecedor* possuía atributos relevantes em si, foi decidido pela externalização do relacionamento na classe *FornecedorEstoquePOJO,* conforme apresentado em mais detalhes na figura 15.

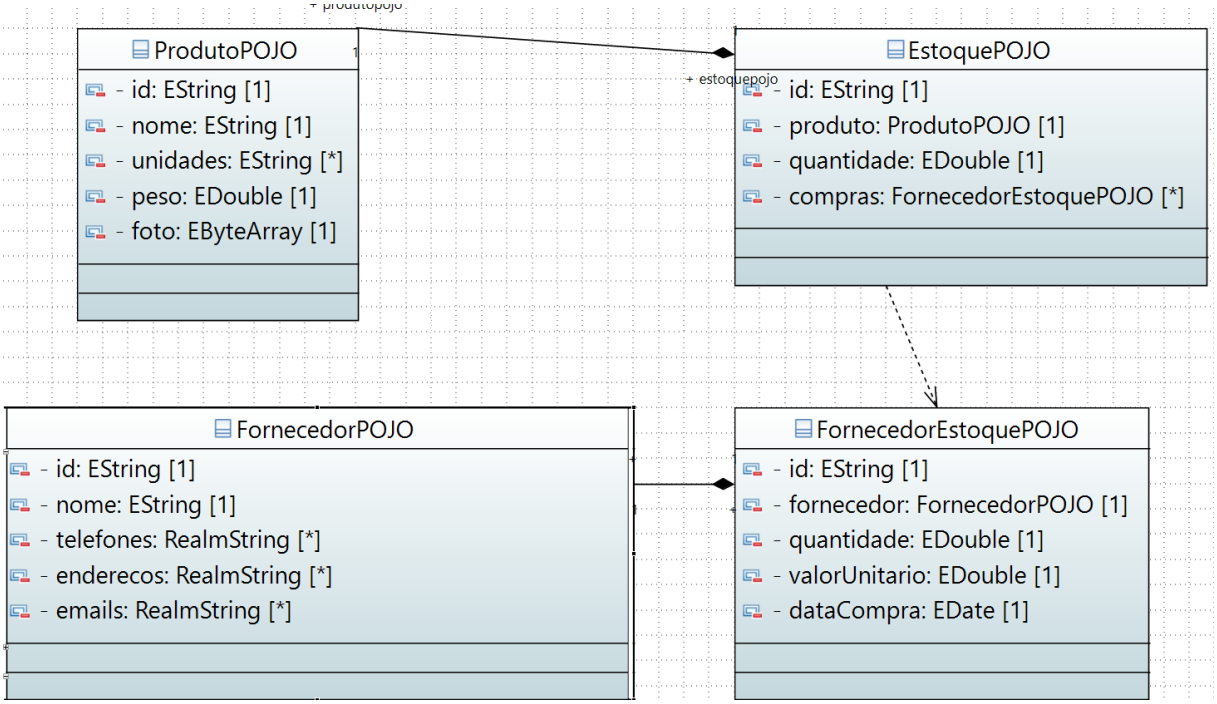


Figura 15 - Relacionamento Fornecedor Estoque

O mesmo se aplica a interação das entidades *Produto* e *Venda,* a qual também teve a criação de uma classe específica, no caso *ProdutoVendaPOJO,* como apresentado na figura 16.

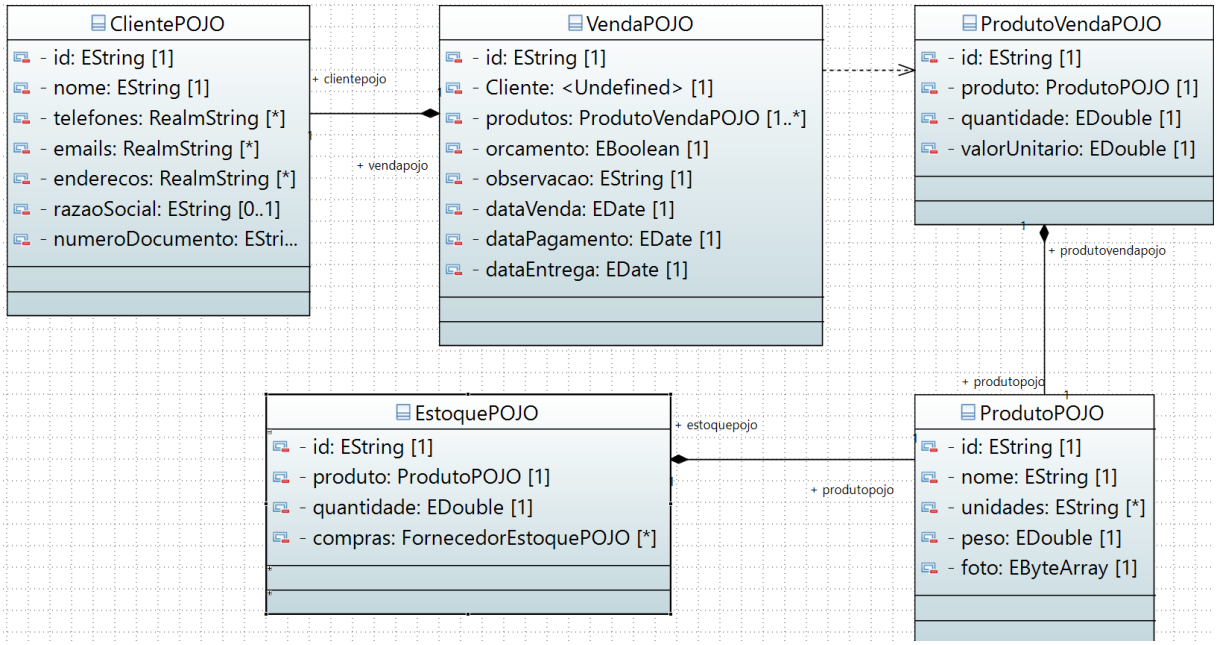


Figura 16 - Relacionamento Produto Venda

Por fim, também é possível notar o incremento de alguns atributos relevantes definidos pelos testadores e *product owner* durantes as *sprints* inicias do projeto, como por exemplo, foto para *ProdutoPOJO* e e-mail para *FornecedorPOJO*.

Por fim, na Seção 7.2 apresenta a versão deste artefato na execução da *sprint* treze, nesta versão a única alteração para versão anterior foi a criação de nova classe *PontoPOJO*, esta criada para armazenar os destinos/pontos a serem apresentados no mapa da tela inicial do aplicativo, podendo estes destinos serem gerenciados por cadastro próprio, sendo que nesse cadastro podem ser alterados os seguintes atributos da classe *PontoPOJO:*

- nome: nome definido pelo usuário para o ponto;
- latitude: atributo gerado automaticamente pelo sistema, após usuário fornecedor o endereço do ponto;
- longitude: atributo gerado automaticamente pelo sistema, após usuário fornecedor o endereço do ponto;
- endereço: endereço de entrega para esse ponto;
- dataVisita: data e hora para a entrega/passagem por este ponto.

Ainda, esta classe *PontoPOJO* relaciona-se com a classe *VendaDAO* de forma que, a cada venda persistida no banco de dados também é criado um novo ponto de visita no mapa caso o campo "*dataEntrega*" da venda tenha sido informado e seja uma data futura. Tal relacionamento é ilustrado na figura 17 a seguir:

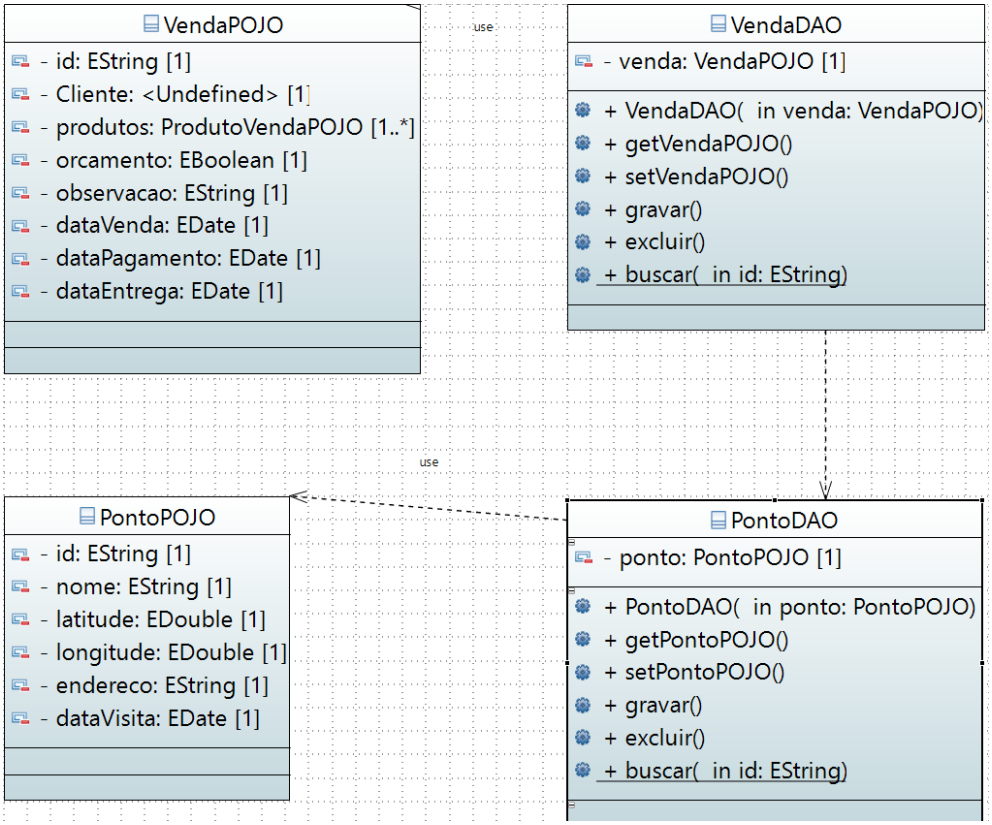


Figura 17 - Relacionamento entre Venda e Ponto

3.2.2.2 Diagrama Entidade Relacionamento

Na figura 18 a seguir, são apresentados os relacionamentos entre as entidades na *sprint* 2.

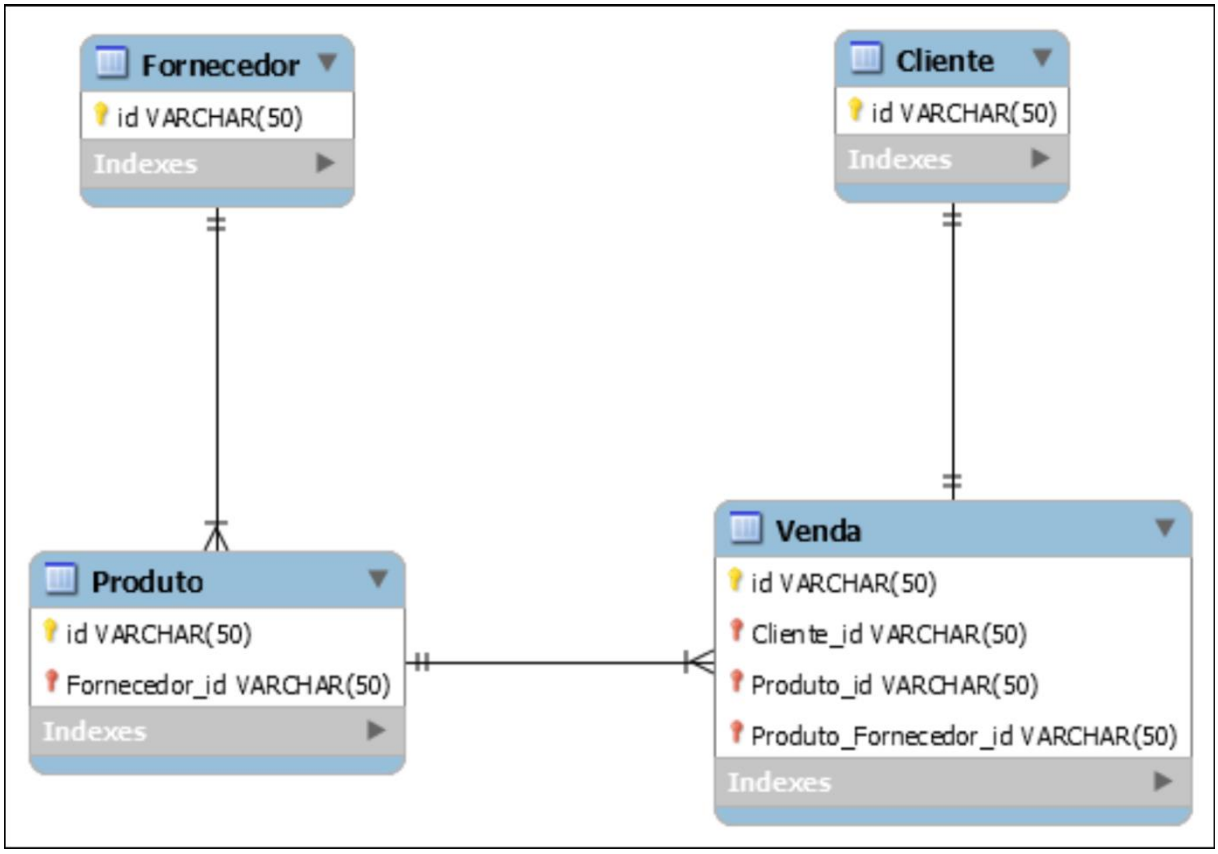


Figura 18 - Diagrama Entidade Relacionamento na *sprint* 2

Neste diagrama, o foco foi apresentar as relações entre as principais entidades do sistema e quais são os relacionamentos entre elas. No diagrama é possível verificar que:

- Um determinado produto pode ser fornecido por vários fornecedores;
- Uma determinada venda para um cliente pode conter vários produtos.

Uma nova versão deste artefato foi gerado na *sprint 10* podendo o mesmo ser visto na figura 19.

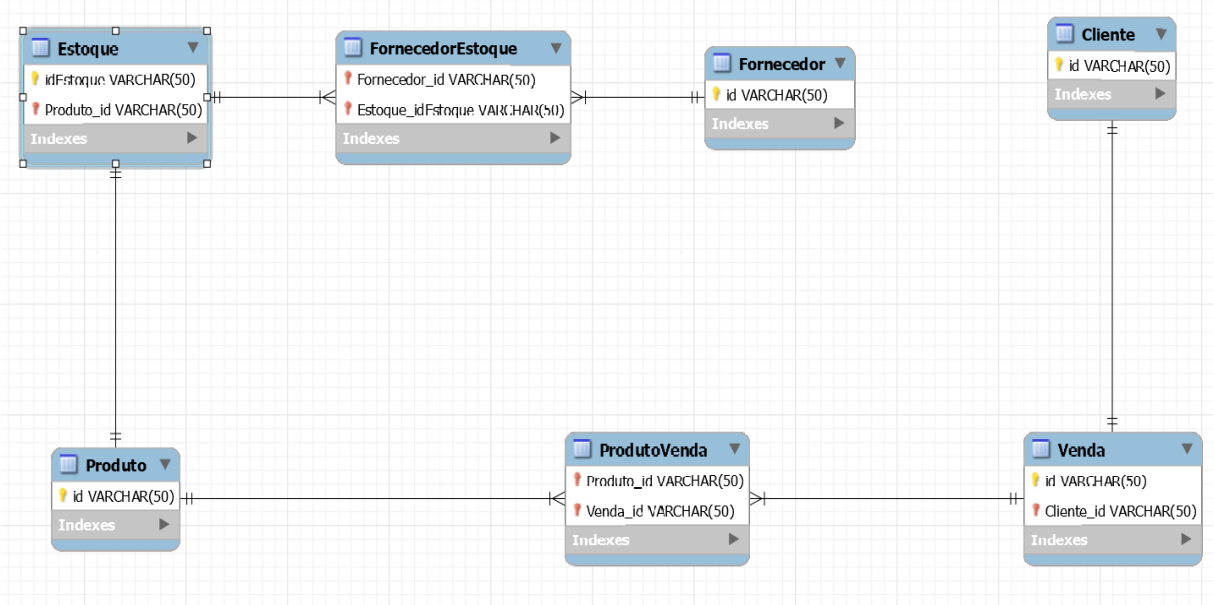


Figura 19 - Diagrama Entidade Relacionamento na sprint 10

Nesta última imagem está o reflexo da alteração na forma de representar os relacionamentos entre classes apresentada na seção anterior quando do momento de exposição do artefato de diagrama de classes na *sprint 10*. As características que podem ser observadas nessa nova versão são:

- Cada *Estoque* representa apenas um *Produto*;
- Cada *Estoque* pode ser composto por várias compras (representada no diagrama por "*FornecedorEstoque*"), sendo que cada compra é feita com um determinado *Fornecedor*;
- Cada *Venda* é realizada apenas a um *Cliente;*
- Cada *Venda* pode ser composta de várias requisições de produtos (representada no diagrama por "*ProdutoVenda*") e cada requisição, por sua vez, contém apenas um produto.

Após o decorrer do projeto, chegou-se a última versão deste artefato que é apresentada na figura 20.

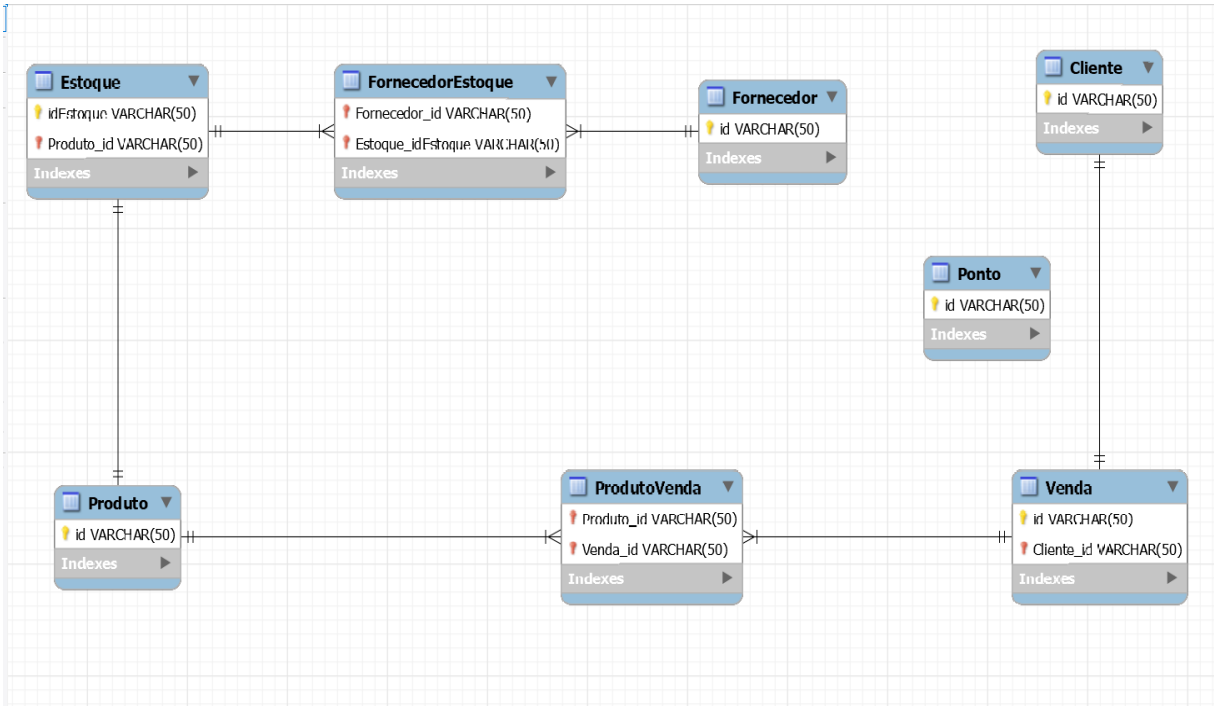


Figura 20 - Diagrama Entidade Relacionamento na sprint 13

Nesta última imagem está representada a alteração do artefato apresentado na Figura 19 para contemplar a criação da entidade *Ponto,* porém como é possível ver pela representação, não há uma relação específica desta com nenhuma outra entidade. Isso foi planejado para que o usuário tenha opção de incluir livremente pontos de destino em seu mapa da tela inicial do aplicativo.

3.3 Gerenciamento de projeto

Como se optou pelo uso da ferramenta *Personal Scrum* para gerenciamento da execução deste projeto, gerou-se os artefatos inerentes a essa metodologia e todos eles seguindo como base os exemplos de (ANDREWS, 2017).

Aqui vale ressaltar que, após o inicio da criação do Capítulo três deste texto, passou a utilizar o *Scrum* tanto para o controle das tarefas do desenvolvimento de software, quanto das tarefas de criação deste texto em si, assim é possível ver nos artefatos apresentados a seguir, tanto tarefas de desenvolvimento do aplicativo, quanto tarefas de escrita e ajuste do texto desta monografia. Nas seções anteriores, alguns destes artefatos já foram apresentados, sendo que nesta serão vistos os faltantes e as devidas explanações.

3.3.1 Planejamento da *sprint*

Para organização das atividades a serem executadas no período de uma *sprint*, no caso deste projeto o período de uma semana, a ferramenta sugere a criação de um artefato que é o documento de planejamento da *sprint.* Neste projeto, esse artefato evolui conforme mostrado nas figuras 21 e 22 a seguir:

Número Sprint	Objetivos	Tarefas	Pontos	Situação Tarefa	Total
2	Escrever parte do capítulo3 - análise	Gerar imagens de diagrama de classes	1	Feito	
2		Gerar imagens de backLog do produto	1	Feito	
2		Gerar imagens de Sprint Planning	1	Feito	
2		Gerar imagens de DailyReview	1	Feito	
2		Gerar imagens de SprintReview	1	Feito	
2		Escrever conteudo para subtópico 3.1 Análise	5	Feito	
2	Refatorar Código gerado para CRUD de Fornecedor	Análisar funções que podem ser reutilizaveis	3	Feito	
2		Modificar as funções identificadas	5	Feito	
2		Estabilizar aplicativo com o refatoramento aplicado	2	Feito	
2	Atualizar diagrama de classes	Refletir mediator e Singleton no modelo de classes	3	Feito	
2		Gerar classe de produto	2	Feito	
					25

Figura 21 - Planejamento da *sprint* 2

Aqui são apresentadas as colunas:

- Número *sprint* – indicador de qual *sprint* está sendo planejada;
- Objetivos – campo para definir macro atividade a ser afetada pela tarefa;
- Tarefas – atividades a serem executadas;

- Pontos – estimativa de esforço para determinada atividade. Notar que esses pontos não medem horas e sim uma medida pessoal de dificuldade da tarefa, sendo que 1 seria uma tarefa fácil e 5 seria uma tarefa difícil/trabalhosa/complexa;
- Situação Tarefa – indicador usado para apresentar se determinada tarefa já foi feita;
- Total: somatório do total de pontos de uma *sprint,* usado para rastrear quantos pontos são possíveis de serem executados em uma *sprint* e numa visão mais ampla quantas tarefas.

Será analisada como exemplo a segunda linha da imagem apresentada, nela pode-se ver que um dos objetivos para essa *sprint* é “Escrever parte do capítulo 3 – análise”. Para atingir tal objetivo houve a criação de várias tarefas, dentre elas a de “Gerar imagens de diagrama de classes”. Para a execução desta tarefa tomada como exemplo foi previsto um esforço necessário equivalente a um ponto, conforme apresentado pela coluna “Pontos”. Por fim, tal tarefa já havia sido completada no momento em que a imagem deste artefato foi gerada, como visto pela coluna ”Situação Tarefa”.

Número Sprint	Objetivos	Tarefas	Pontos	Situação Tarefa	Total
9	Escrever Monografia	Gerar imagens das telas do aplicativo sendo construído	3	Feito	
9		Com as imagens das telas, gerar passo-a-passo de como usar o sistema	4	Feito	
9	Refatorar Código	Refatorar código para adapters	4	Feito	
9		Refatorar código para criar botoes de adicionar	5	Parcial	
9		Refatorar outros trechos passiveis de parametrização	3	Feito	
9	Cadastro de cliente	Possiblitar escolher endereço do google maps	3		
					22
10	Cadastro de cliente	Permitir escolher cliente da agenda	2	Feito	
10		Possiblitar escolher endereço do google maps	4	Feito	
10	Cadastro de venda	Gerar relatório da venda	7	Parcial	
10		Compartilhar relatório via e-mail e outros meios	7		
10	Avaliar usabilidade	Elaborar forma de avaliar facilidade de uso	2		
10		Após ajuste do cadastro de cliente, avaliar facilidade de uso com os usuários finais	2		
10	Escrever Monografia	Com a avaliação acima, iniciar escrita capítulo 4	2		
10		Gerar novas imagens de artefatos para capítulo 3	2		
10		Acrescentar acima imagens ao capítulo 3	2		
					30
11	Escrever Monografia	Com a avaliação de usabilidade, iniciar escrita capítulo 4	2		
11		Gerar novas imagens de artefatos para capítulo 3	2		
11		Acrescentar acima imagens ao capítulo 3	2		
11	Avaliar usabilidade	Elaborar forma de avaliar facilidade de uso	2		
11		Avaliar facilidade de uso com os usuários finais	2		
11	Cadastro de venda	Finalizar elatório da venda	3		
11		Compartilhar relatório via e-mail e outros meios	7		
					20

Figura 22 - Planejamento da sprint 11

Até a *sprint* 11 não aconteceram alterações neste artefato, sendo sua apresentação apenas para salientar o uso para acompanhamento/planejamento das atividades.

3.3.2 Revisões diárias

Como previsto pelo *Scrum,* utilizou-se um documento para tomar nota das atividades diárias e suas dificuldades. A seguir, nas figuras 23 e 24, serão apresentados o documento que contém essas revisões e como foi a evolução deste artefato no projeto.

Sprint	Data	Feito	Por fazer	Impedimento
2	07/08/2017	Consegui arrumar o modelo de dados para melhor refletir o que estava sendo feito, escrevi parte do capítulo 3.1 - análise e refatorei partes do código desejaveis	Testar parte do código que ficou faltando, revisar texto já feito para o capítulo 3 e começar a gerar classe de produto	Não houve impedimento no dia anterior
2	08/08/2017	Consegui testar o aplicativo com o código refatorado, revisar parte do texto inicial do capítulo 3 e começar a gerar parte da classe de produtos	Finalizar classe de produtos, gerar imagem do diagrama de classes dessa sprint, desenvolver mais o texto do capítulo 3.1, gerar imagens de artefatos do *Scrum*	Não houve impedimento no dia anterior
3	09/08/2017	Consegui gerar a classe inicial de produtos, mas imagens dos artefatos e elaborar melhor o texto do capítulo 3 sobre requisitos e modelo de dados	Analisar referencia sobre capítulo 3 e pensar em forma de apresentar artefatos do *scrum*	Por conta do último dia de *sprint,* foi um dia apertado na questão de tempo de trabalho para executar todas as tarefas.
3	10/08/2017	Consegui gerar subtópico para para gerenciamento de projeto para apresentadar artefatos scrum	Enviar texto do capítulo 3 para avaliação do orientador	Não houve impedimento no dia anterior

Figura 23 - Revisões diárias até *sprint 2*

Nesta imagem do artefato são apresentadas as colunas:

- *Sprint*: coluna usada para indicar a qual *sprint* determinada revisão diária pertence;
- Data: campo para registrar o dia da revisão;
- Feito: espaço para anotar as tarefas que foram feitas no dia anterior;
- Por fazer: espaço para que sejam anotadas as tarefas por serem feitas no dia em questão;
- Impedimentos: campo usado para registrar qualquer forma de impedimento que tenha surgido durante o dia anterior.

Tomando-se a segunda linha da imagem apresentada como exemplo, é possível verificar na mesma que para o dia 07/08/2017 deveriam ser executadas as tarefas de testar parte do código gerado, revisar o texto do capítulo três deste trabalho e início da geração de novo código para classe *Produto*. Também é possível analisar que no dia anterior a este analisado, foi possível melhorar o modelo de dados, mais precisamente o diagrama de classes, produzir conteúdo para o

capítulo três do seguinte volume e refatorar partes do código do aplicativo. Por fim, não houve impedimentos que afetaram estas últimas tarefas descritas.

Sprint	Data	Feito	Por fazer	Impedimento
9	20/09/2017	Não consegui refatorar o código apenas finalizei a *sprint*	Gerar imagens do aplicativo	Não consegui ter nenhuma idéia para refatorar o código existente
9	21/09/2017	Gerado imagens do aplicativo no Google Drive	Inserir imagens no texto da monográfia e criar texto de uso	Não houve impedimentos no dia anterior
10	27/09/2017	Finalizado a *sprint* e testado refatoramento	Gerar código para buscar cliente da agenda e iniciar codificação para escolher endereço do google maps	Não houve impedimentos no dia anterior
10	28/09/2017	Consegui gerar o código para buscar cliente da agenda de contatos	Buscar como escolher endereços via Google maps e iniciar codificação	Não houve impedimentos no dia anterior
10	29/09/2017	Iniciado codificação para uso da API do google maps	Continuar codificação do uso do Maps	Não houve impedimentos no dia anterior
10	30/09/2017	Abertura de maps e escolha do local funcionando	Salvar endereço do Realm e arrumar apresentação do endereço no campo especifico. Buscar como gerar relatórios no Android	Não houve impedimentos no dia anterior
10	01/10/2017	Endereço sendo registrado no Realm do cliente e iniciado desenvolvimento do layout do relatório de vendas	Gerar relatório pdf da venda e iniciar pesquisa de como compartilhar o relatório	Não houve impedimentos no dia anterior
10	02/10/2017	Relatório tomou mais tempo do que previsto, ajustando mesmo para ser gerado na pasta DOWNLOADS	Fazer relatório disponivel na pasta downloads	Não houve impedimentos no dia anterior
10	03/10/2017	Relatório tomou mais tempo do que previsto, não saindo como o previsto no layout.xml do mesmo	Ajustar layout.xml do relatório e finalizar *sprint*	Não houve impedimentos no dia anterior
11	05/10/2017		Iniciar correções propostas pelo orientador	Indisponibilidade de tranalho no dia de ontem
11	06/10/2017	Iniciado correções	Continuar correções propostas pelo orientador	Não houve impedimentos no dia anterior
11	07/10/2017	Continuo das correões	Finalizar correções propostas pelo orientador, gerar imagens do capítulo 3 e continuar codificação do relatório de venda	Não houve impedimentos no dia anterior

Figura 24 - Revisões diárias até sprint 11

Até a *sprint* 11 não aconteceram alterações neste artefato, sendo sua apresentação apenas para salientar o uso para acompanhamento das atividades e interrupções diárias.

3.3.3 Revisões de Sprint

Por fim, o último artefato gerado foi o de revisão da s*print*. Embora muito parecido com a revisão diária, é um artefato útil para que seja visualizado o que foi conseguido ao longo de uma *sprint* e possíveis melhorias. A seguir, nas figuras 25 e 26 é possível ver a utilização deste artefato no projeto e como foi a progressão deste no decorrer das *sprints.*

A	B
Sprint	Retrospectiva
02/08/2017 - 08/08/2017	Nesta *sprint* consegui gerar parte do texto do capítulo 3.1 - análise, refatorar o código para facilitar criação de novos CRUDs, gerar artefatos para uso do *scrum* e com isso consegui atingir fazer todo que havia me programado para essa rodada. A forma de controle com o personnal *scrum* foi bem interessante por isso acredito q tenha sido algo bom. Houve trabalhos externos ao projeto que limitaram o tempo para execução das atividades dessa *sprint,* sendo assim, para as próximas talvez seja possível mais pontos de atividade se for possível limitar interferências.

Figura 25 - Retrospectivas da *sprint dois*

Aqui as colunas apresentadas para este artefato são:

- *Sprint*: Coluna usada para apresentar o período de duração da s*print*;
- Retrospectiva: campo para que seja resumido o que foi possível de ser realizado com a determinada *sprint,* durante esse tempo se houve algo de bom/ruim que deve ser registrado e se há algo que pode ser mantido/melhorado de acordo com esse último registro.

Na imagem apresentada é tido como exemplo a *sprint* que ocorreu no período de 02/08/2017 a 08/08/2017 e a retrospectiva da mesma foi: a possibilidade de gerar conteúdo para o capítulo três deste trabalho, refatorar o código do aplicativo e gerar artefatos da metodologia *Scrum*. Neste período, o ponto positivo foi o contato com a forma de controle de trabalho oferecido pelo *personal scrum.* Os pontos negativos foram os trabalhos fora do contexto do projeto durante o período reservado para execução deste. E por fim, como sugestão de melhoria para próxima *sprint* foi decidido um melhor controle destas interrupções a fim de proporcionar um aumento na quantidade de tarefas no momento de planejamento da *sprint.*

Sprint	Retrospectiva
04 - 16/08/2017 - 22/08/2017	Nesta *sprint* foi possível apenas realizar correção das sugestões do orientador, pois estar demandaram retrabalho e junto a esta fato, aconteceram várias interrupções externas. Não foi possível desenvolver mais conteúdo do texto e no aplicativo foi possível apenas melhorar a escolha do fornecedor.
05 - 23/08/2017 - 29/08/2017	Nesta *sprint* foi ajustar o cadastro de produto para incluir foto do mesmo e realizar o CRUD de cliente. Também foi possível enviar mais um pouco de conteúdo para revisão do orientador
06 - 30/08/2017 - 05/09/2017	Nesta *sprint* foi possível gerar vários trechos de conteúdo para o capítulo 3.4 - implementação e melhor detalhar o conteúdo dos artefatos *scrum*. Também foi possível alterar o cadastro de fornecedor, buscando informações de contato da agenda e criar o CRUD para tela de vendas, porém o esforço para esta tarefa foi medido incorretamente, sendo necessário mais tempo que o previsto para executá-la. Para próxima *sprint* melhor a pontuação para tarefas envolvendo CRUD.
07 - 06/09/2017 - 12/09/2017	Nesta *sprint* apenas finalizei o cadastro de estoque e alguns elementos pré textuais do texto da monografia. Não foi possível desenvolver mais por conta de viagem no feriado de Setembro.
08 - 13/09/2017 - 19/09/2017	Nesta *sprint* gerei mais conteúdo para o capítulo 3 e propus forma de abordagem para capítulo 4. Além disso foi possível atualizar estoque de acordo com a venda, atualizar topo do aplicativo com o nome do cadastro que usuário está, atualizar diagramas (DER e de classe) e começar a refatorar código. Novamente avaliei errado o esforço necessário para atividade, porém desta vez para atividade de atualização de diagramas que demorou mais do que o previsto
09 - 20/09/2017 - 26/09/2017	Nesta *sprint* consegui apresentar telas do aplicativo no texto da monográfia, bem como fazer um breve manual do usuário e gerar mais conteúdo para especificação dos pacotes, além de enviar monográfia para avaliação do Orientador. No aplicativo foi possível refatorar trechos de código que serão reaproveitados em novas funcionalidades e/ou reformulação de CRUDs já feitos. Algumas tarefas não foram possíveis de serem executadas por conta de atividades externas, porém a avaliação do esforço parece estar mais justa.
10 - 27/09/2017 - 03/10/2017	Nesta *sprint* foi desenvolvido a escolha do endereço do cliente com base na API de mapas do Google e foi dado inicio ao desenvolvimento do relatório de venda para compartilhamento. Ambas funcionalidades descritas, não se tinha conhecimento prévio para implantação, assim foi interessante a busca de "como fazer", porém, por conta da falta de experiência com tais funcionalidades, especialmente a de relatório tomou mais tempo que o esperado.

Figura 26 - Retrospectivas até *sprint 10*

Até a *sprint* 11 não aconteceram alterações neste artefato, sendo sua apresentação apenas para salientar o uso para revisão do que foi possível executar em cada *sprint.*

3.4 Implementação

Para a implementação deste projeto, por conta da característica do sistema como um todo ser composto por vários módulos, optou-se por também dividir a codificação do aplicativo em vários pacotes, assim, ao analisar a estrutura do projeto, é possível ver claramente os módulos e os componentes necessários ao seu funcionamento. Portanto, a apresentação da estrutura do aplicativo é algo semelhante ao que segue na figura 27:

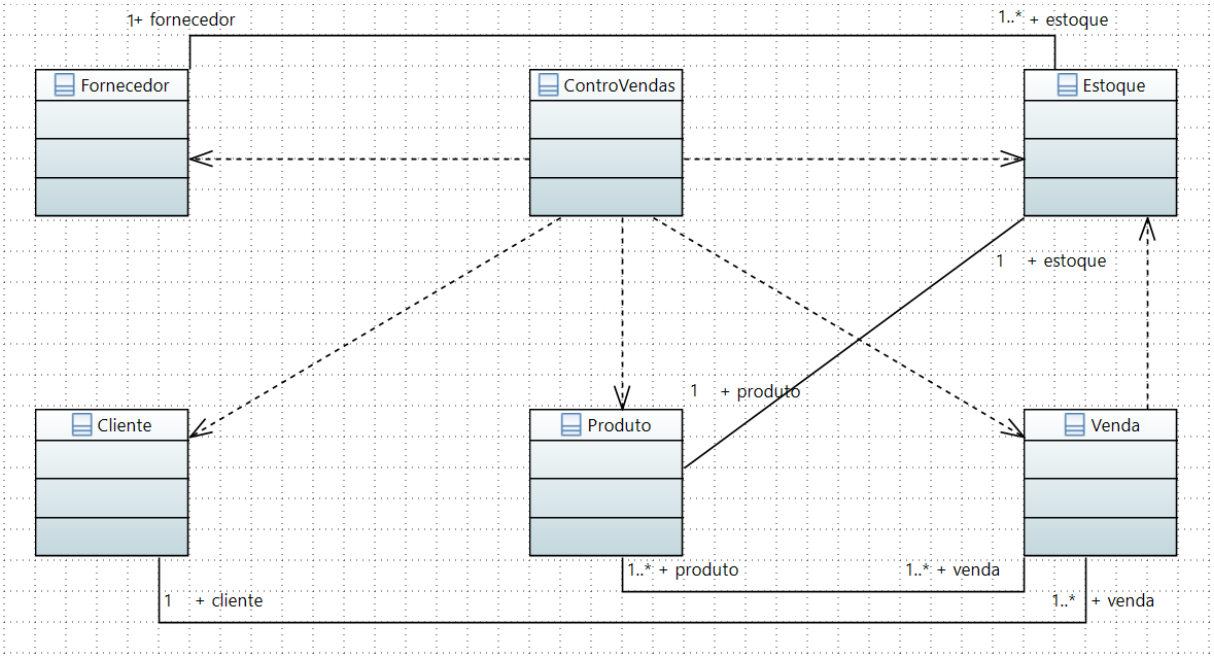


Figura 27 - Estrutura dos pacotes do aplicativo

- *ControVendas*: Pacote inicial do aplicativo, responsável por oferecer ao usuário formas de acessar os demais módulos do sistema e gerenciar a execução destes.
- *Fornecedor*: Pacote responsável por apresentar (aqui o conceito de apresentar, engloba tanto a listagem em tela dos fornecedores quanto apresentação deste em relatórios.), incluir, alterar e excluir informações de fornecedores;
- *Cliente*: Pacote responsável por apresentar, incluir, alterar e excluir informações relativas a clientes;
- *Produto*: Pacote responsável por apresentar, incluir, alterar e excluir informações relativas a produtos;
- *Venda*: Pacote responsável por apresentar, incluir, alterar e excluir informações relativas a vendas;
- *Estoque*: Pacote responsável por apresentar, incluir, alterar e excluir informações relativas ao estoque de produtos.

A seguir, serão apresentadas as estruturas dos pacotes acima descritos com relação aos artefatos de código. Neste momento, vale ressaltar que durante a codificação dos diversos módulos, buscou-se a utilização de uma estrutura padrão para as várias entidades do aplicativo. Assim, para evitar a repetição da mesma

explanação nos trechos a seguir, será feita uma explicação de tal estrutura base via *template* e, caso algum pacote tenha alguma diferença desta estrutura, será explicitado.

3.4.1 Pacote *ControVendas*

Neste pacote estão os seguintes artefatos de código apresentados na figura 28:

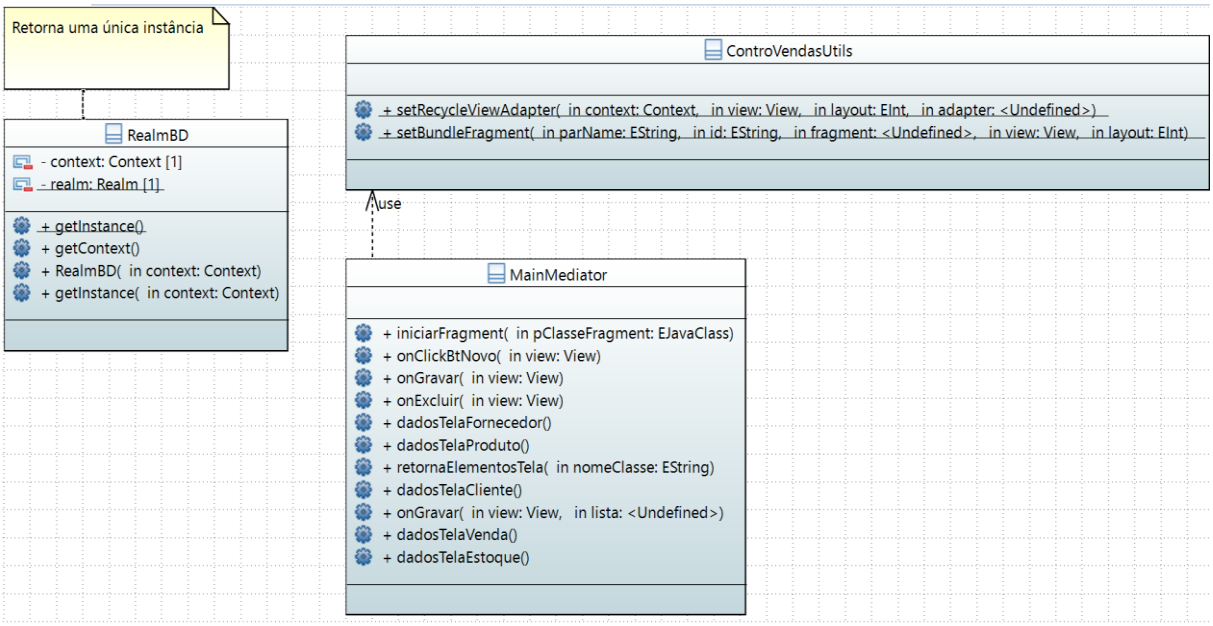


Figura 28 – Estrutura de classes do Pacote ControVendas

- *MainMediator* – responsável pelo menu do aplicativo, acesso aos módulos e troca de mensagens entre classes da interface e de acesso a dados;
- *ControVendasUtils* – responsável por agrupar códigos reutilizáveis que podem ser utilizados pelos diversos módulos, como por exemplo: criação de instâncias de listas, passagem de parâmetros entre telas etc;
- *RealmBD* – responsável pelo acesso a base de dados.

Estas classes acima descritas colaboram para que o funcionamento da seguinte tela seja possível:

- Tela de menu – menu lateral a partir do qual o usuário pode acessar os outros módulos, apresentado no aplicativo conforme figura 29;

Figura 29 – Menu do sistema

3.4.2 *Template* de Estrutura das Entidades

Como dito anteriormente, buscou-se durante a codificação deste sistema, a utilização de uma mesma estrutura de construção para todas as entidades, tanto para facilitar o entendimento por terceiros quanto para a codificação em si. Assim, pode-se dizer que todas as entidades deste sistema seguiram um mesmo *template* de composição de estrutura interna de classes que seria a que segue apresentada na figura 30:

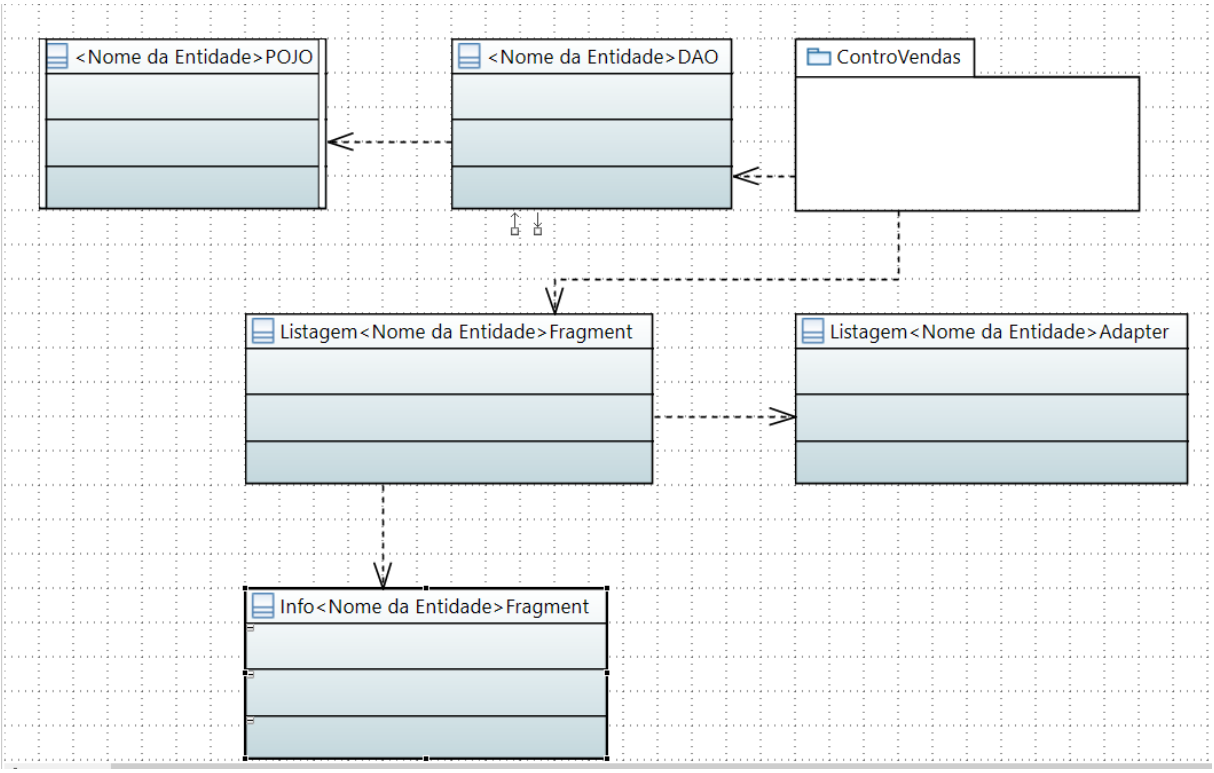


Figura 30 - Estrutura de classes do *Template* aplicado

- <Nome da Entidade>*POJO* – responsável por delimitar os atributos básicos de determinada entidade para o sistema;
- <Nome da Entidade>*DAO* – responsável por acessar e alterar dados de determinada entidade no banco de dados;
- Info<Nome da Entidade>*Fragment* – responsável pela apresentação da tela de cadastro de determinada no sistema;
- Listagem<Nome da Entidade>*Adapter* – responsável pela criação de listagem de determinada a ser apresentada;
- Listagem<Nome da Entidade>*Fragment* – responsável por listar os registros de determinada cadastrados do sistema, direcionar o usuário para tela de cadastro/alteração da mesma e/ou para tela de relatórios.

3.4.3 Pacote *Fornecedor*

Aqui será utilizada a entidade *Fornecedor* para exemplificar a aplicação da estrutura padrão , conforme apresentado na figura 31 a seguir:

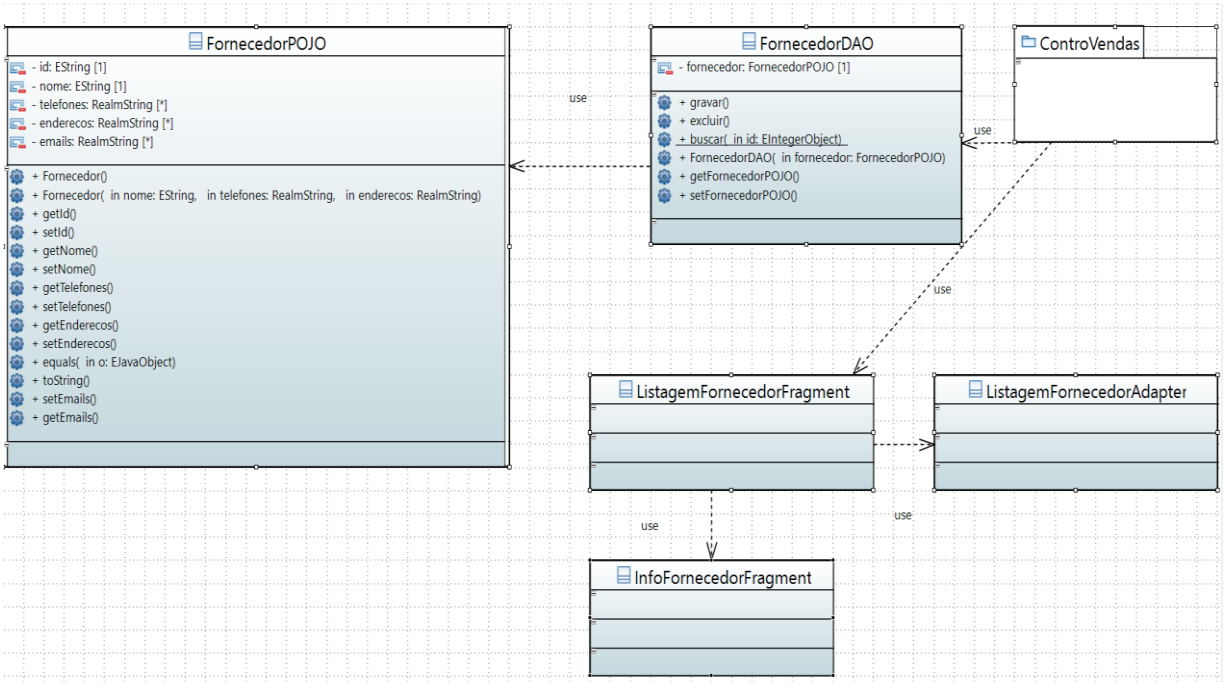


Figura 31 - Estrutura de classe de entidade *Fornecedor*

- *FornecedorPOJO* – responsável por delimitar os atributos básicos de Fornecedor para o sistema;
- *FornecedorDAO* – responsável por acessar e alterar dados do Fornecedor no banco de dados;
- *InfoFornecedorFragment* – responsável pela apresentação da tela de cadastro de fornecedores no sistema;
- *ListagemFornecedorAdapter* – responsável pela criação de listagem de fornecedores a ser apresentada;
- *ListagemFornecedorFragment* – responsável por listar os fornecedores do sistema, direcionar o usuário para tela de cadastro/alteração de fornecedores e/ou para tela de relatórios.

Além da estrutura básica, este pacote ainda conta com o seguinte artefato de código:

- *FornecedorAdapter* – responsável pela listagem dos diversos fornecedores cadastrados no sistema. Tal listagem pode ser utilizada por exemplo, em campo de Autocompletar, sendo necessário que o usuário digite apenas duas letras para que seja feito a sugestão de algum fornecedor.
-

O conjunto de todas as classes descritas acima culmina nas funcionalidades apresentadas nas seguintes telas:

- Tela de listagem dos fornecedores cadastrados no sistema – Nesta tela, quando o usuário clicar sobre o nome do fornecedor poderá ver as informações detalhadas do mesmo ou ao clicar sobre o botão “Novo”, cadastrar um novo fornecedor. A apresentação de tal tela no aplicativo se dá conforme figura 32;

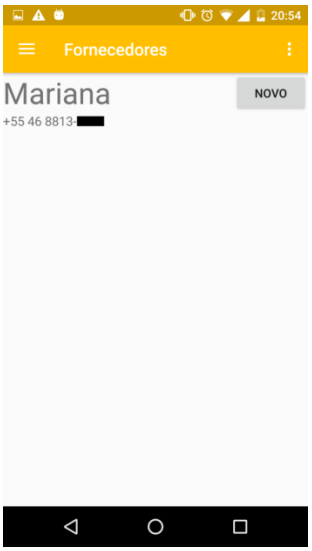


Figura 32 - Listagem de Fornecedores

- Tela de informações de fornecedor – Nesta tela o usuário pode ver as informações detalhadas do fornecedor, alterá-las ou, em caso de novo cadastro, informá-las. Aqui solicitante pode pressionar o botão “Contatos” para que seja aberta a agenda de contatos do telefone e assim, possa escolher um contato, fazendo com que o sistema automaticamente busque as informações necessárias daquela aplicação. Ainda, ao clicar sobre “Gravar”, o fornecedor será salvo e

sobre “Excluir”, irá removê-lo do sistema. A apresentação de tal tela no aplicativo se dá conforme figura 33;

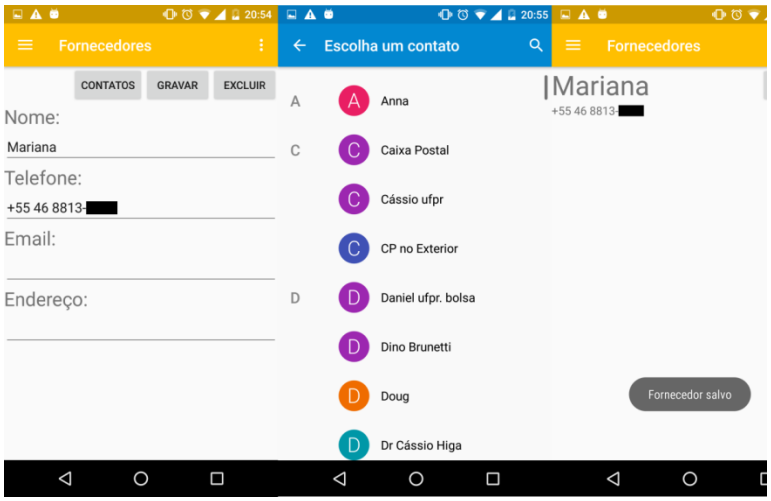


Figura 33 - Cadastro Fornecedor

3.4.4 Pacote *Cliente*

Pacote com a mesma estrutura do pacote *Fornecedor*, sendo que neste está o artefato *ClienteAdapter* para sugestão de clientes em campos de AutoCompletar.

Neste pacote, as classes da entidade *Cliente* fornecem o funcionamento das seguintes telas:

- Tela de listagem de clientes – Funcionamento similar à mesma tela de listagem do pacote *Fornecedor*. A apresentação de tal tela no aplicativo se dá conforme figura 34;

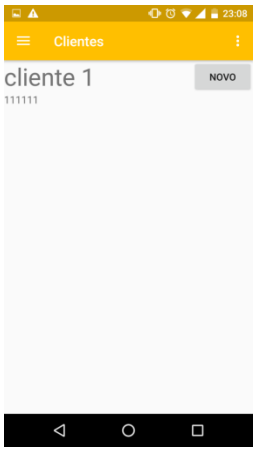


Figura 34 - Listagem de Clientes

- Tela de informações do cliente – nesta tela o usuário pode cadastrar um cliente, visualizar detalhes de algum já cadastrado, alterar informações ou excluí-lo. A apresentação de tal tela no aplicativo se dá conforme figura 35.

Figura 35 - Cadastro de Cliente

3.4.5 Pacote *Produto*

Pacote com a mesma estrutura de Fornecedores, sendo que neste está o artefato *ProdutoAdapter* para sugestão de produtos em campos de AutoCompletar.

As classes contidas neste pacote permitem o funcionamento das telas a seguir:

- Tela de listagem de clientes – Funcionamento similar à mesma tela de listagem do pacote *Fornecedor*. A apresentação de tal tela no aplicativo se dá conforme figura 36;

Figura 36 - Listagem de Produtos

- Tela de informações do produto – nesta tela o usuário pode cadastrar um novo produto com opção de tirar foto do mesmo para montagem de catálogo, visualizar detalhes de algum já cadastrado, alterar informações ou excluí-lo. A apresentação de tal tela no aplicativo se dá conforme figura 37.

Figura 37 - Cadastro de Produto

3.4.6 Pacote ProdutoVenda

Neste pacote está presente apenas o artefato de código *ProdutoVendaPOJO*, que é responsável pela definição dos atributos relevantes da associação entre as entidades *Produto* e *Venda*. Como esta é uma entidade que existe apenas da interação destas duas descritas, não foi encontrado necessidade da criação dos outros artefatos de códigos da estrutura básica neste pacote, por conta da listagem e cadastro dos elementos desta entidade estarem contidos no pacote de *Venda*.

3.4.7 Pacote *FornecedorEstoque*

Assim como o Pacote *ProdutoVenda*, neste também há apenas um artefato que seria *FornecedorEstoquePOJO*, no qual são descritos os atributos relevantes da associação entre as entidades *Fornecedor* e *Estoque*.

3.4.8 Pacote *Estoque*

Pacote sem diferenças da estrutura básica, não sendo necessário nenhum detalhamento adicional.

Para este pacote, todas as classes envolvidas fornecem as funcionalidades apresentadas nas telas a seguir:

- Tela de listagem de produtos em estoque – Funcionamento similiar à mesma tela de listagem do pacote *Fornecedor.* Aqui, a data apresentada logo abaixo do nome do produto corresponde à data da última compra de itens para produto em questão. A apresentação de tal tela no aplicativo se dá conforme figura 38;

Figura 38 - Listagem de estoques de produtos

- Tela de informações do estoque – Nesta tela, o usuário pode incluir informações de um estoque, alterar algum já cadastrado ou excluí-lo. Aqui, para a escolha do produto, o usuário apenas precisa informar as duas primeiras letras do nome em questão, pois o sistema irá mostrar uma pequena listagem com sugestões dos produtos desejados. Depois de escolhido o produto, é possível manualmente informar uma quantidade de itens do produto em estoque ou adicionar compras feitas com fornecedor para aquele produto. Para adicionar compras realizadas, basta o usuário informar o nome do fornecedor e escolhê-lo na lista de sugestões que aparecerá, informar quantidade adquirida na compra em questão e o valor pago por unidade do produto. Após

estas informações, o sistema irá calcular o preço de determinada compra e sendo necessário para finalizar apenas clicar sobre o botão “+”. Isso fará com que os campos preenchidos anteriormente sejam limpos e que a lista de compras do estoque seja incrementada. Ainda, caso uma compra seja indevidamente cadastrada, basta que seja pressionado sobre o botão “x” para que determinada compra seja removido do estoque. Vale ressaltar que, a cada compra cadastrada ou excluída, o sistema irá atualizar o campo “*Quantidade*” do estoque, bem como quando uma *Venda* for realizada, deste que esta não se trate de um orçamento. A apresentação de tal tela no aplicativo se dá conforme figura 39.

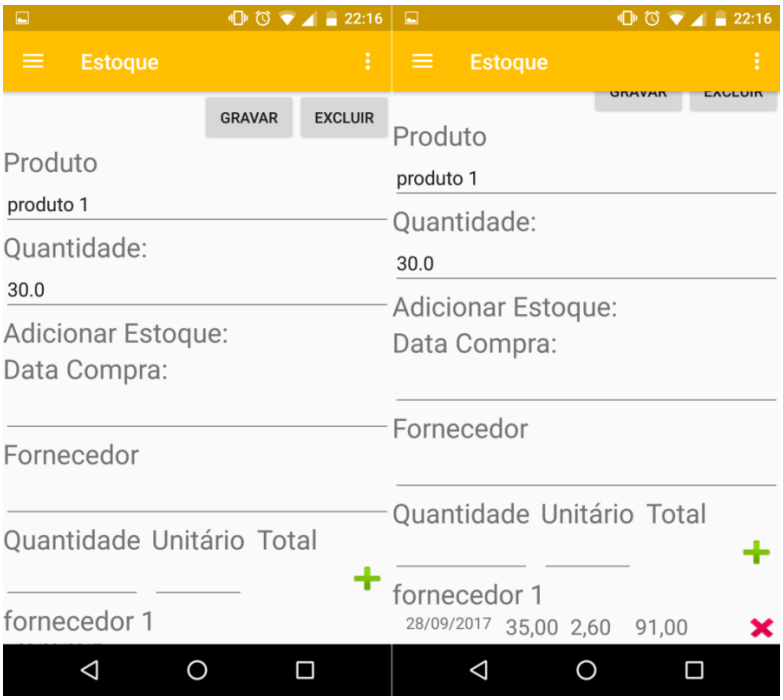


Figura 39 - Cadastro de estoque

3.4.9 Pacote *Venda*

Pacote sem diferenças da estrutura básica, não sendo necessário nenhum detalhamento adicional.

Neste pacote, o conjunto de todas as classes envolvidas, dispõe as funcionalidades apresentadas nas telas a seguir:

- Tela de listagem de vendas – Funcionamento similar à mesma tela de listagem do pacote *Fornecedor*. Vendas listadas aqui são ordenadas pela data da venda A apresentação de tal tela no aplicativo se dá conforme figura 40;

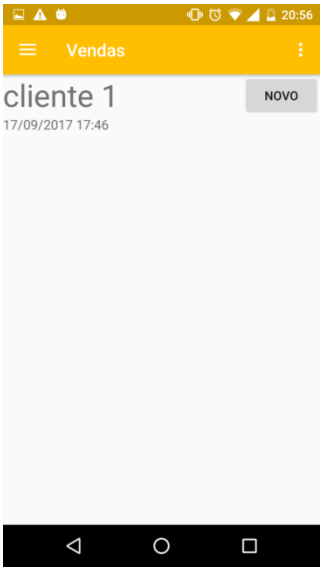


Figura 40 - Listagem de Vendas

- Tela de informações da venda – Nesta tela, o usuário pode incluir uma nova venda, alterar alguma já cadastrada ou excluí-la. Aqui, para realizar o registro, é possível marcar se este cadastro é um orçamento ou não, bastando para tanto, clicar sobre o botão “Orçamento”. Tanto para a escolha do cliente quanto do produto, é necessário apenas informar as duas primeiras letras do nome em questão, pois o sistema irá mostrar uma pequena listagem com sugestões dos clientes/produtos desejados. Depois de escolhido o produto desejado, o usuário pode informar a quantidade e valor unitário do produto e o sistema irá calcular o subtotal para o mesmo. Quando todas as informações do produto estiverem corretas, é possível incluir novos produtos ao cadastro clicando sobre o botão “+”, isso fará com que os campos preenchidos anteriormente sejam limpos e que a lista de produtos da venda seja incrementada. Ainda, caso um produto seja indevidamente incluído, basta que seja pressionado sobre o botão “x” para que determinado produto seja

removido da venda. A apresentação de tal tela no aplicativo se dá conforme figura 41.

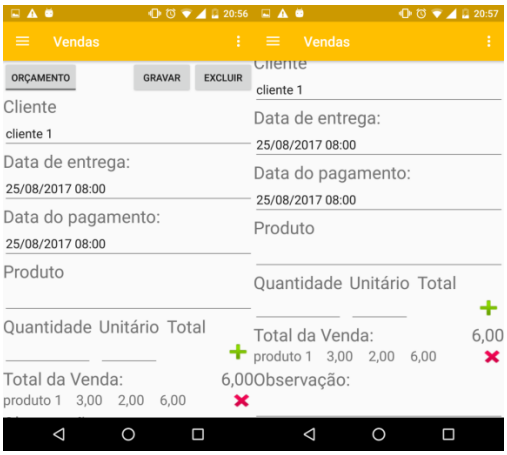


Figura 41 - Cadastro de Venda

3.4.10 Pacote Ponto

Pacote sem diferenças da estrutura básica, não sendo necessário nenhum detalhamento adicional.

Aqui, a união das classes contidas nesse pacote fornece as seguintes funções:

- Tela de listagem de pontos - Funcionamento similar à mesma tela de listagem do pacote *Fornecedor*. Pontos listados aqui são ordenados pela data da visita a determinado ponto/destino. A apresentação de tal tela no aplicativo se dá conforme figura 42;

Figura 42 - Listagem de pontos/destinos

- Tela de informações do ponto – nesta tela o usuário pode cadastrar, alterar ou excluir um ponto cadastrado. Aqui é possível informar um

nome para o ponto em questão, horário que tal ponto deve ser visitado e o endereço do mesmo, sendo que é possível a utilização do botão “*Mapa*” para a escolha de tal informação. A apresentação de tal tela no aplicativo se dá conforme figura 43.

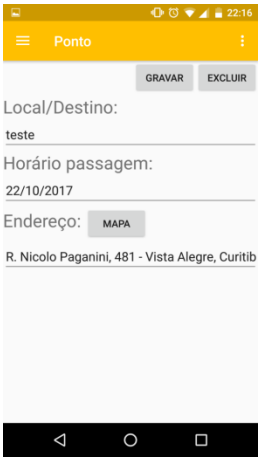


Figura 43 - Cadastro de ponto/destino

3.4.11 Pacote Rota

Este pacote não segue o *template* padrão de estrutura de classes por apenas conter a classe *MapaRotaFragment*, responsável por buscar e listar os pontos a serem apresentados no mapa, bem como de chamar o serviço responsável por calcular a rota entre os pontos listados e apresentá-la. Os pontos aqui listados são todos os com data de visitação igual à data atual e são classificados nas cores vermelho, amarelo e verde de acordo com a data de visita em relação a data atual, sendo que o ponto vermelho representada pontos que faltam menos de uma hora para a data de visitação, amarelo entre uma e duas horas e verde mais do que duas horas.

A seguir, na figura 44, é possível visualizar imagens da tela de mapa apresentada como tela inicial do aplicativo.

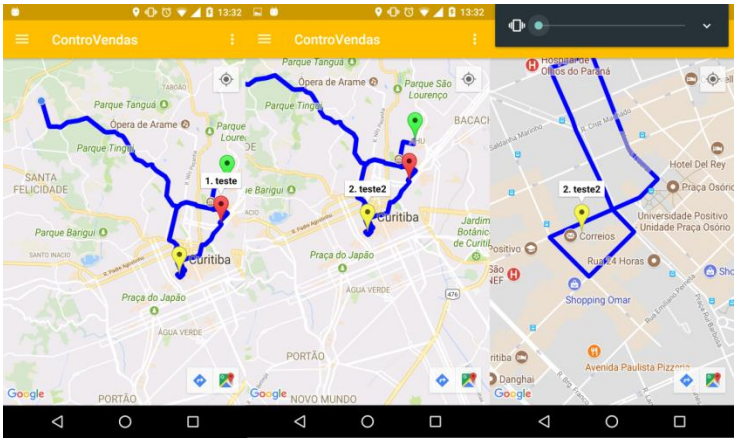


Figura 44 - Mapa com pontos/destinos

4 TESTE DE USABILIDADE

O estudo de usabilidade deste trabalho é composto por diversos testes de usabilidade aplicados a alguns avaliadores escolhidos. Esses avaliadores foram selecionados de acordo com sua disponibilidade para realizar o teste, contato com a atividade fim alvo deste trabalho e idade. Embora o nível de afinidade tecnológica tenha sido uma questão respondida pelo próprio avaliador, de acordo com um analise pessoal sobre si, também se pensou nessa característica no momento de escolha dos avaliadores.

Para iniciar o estudo, serão definidas primeiramente as tarefas a serem completadas pelos avaliadores no teste de usabilidade, que são:

- Acessar menu do sistema;
- Acessar pacote de gerenciamento de fornecedores;
    - Realizar as quatro atividades básicas (consultar, cadastrar, alterar e excluir) envolvendo esta entidade;
    - Acessar informações da agenda de contato para cadastrar um fornecedor;
- Acessar pacote de gerenciamento de clientes;
    - Realizar as quatro atividades básicas envolvendo esta entidade;
    - Acessar informações da agenda de contato para cadastrar um cliente;
- Acessar pacote de gerenciamento de produtos;
    - Realizar as quatro atividades básicas envolvendo esta entidade;
- Acessar pacote de gerenciamento de estoques;
    - Realizar as quatro atividades básicas envolvendo esta entidade;
    - Cadastrar nova adição de itens a determinado estoque;
    - Realizar a remoção de itens de determinado estoque via interface de estoque;
- Acessar pacote de gerenciamento de vendas;

    - Realizar as quatro atividades básicas envolvendo esta entidade;
    - Cadastrar nova adição de itens a determinada venda;
    - Realizar a remoção de itens de determinada venda;
    - Compartilhar relatório da venda;

- Acessar pacote de gerenciamento de pontos;
    - Realizar as quatro atividades básicas envolvendo esta entidade;
    - Verificar pontos adicionados no mapa da tela inicial.

Após o acompanhamento e verificação de completude de tais tarefas pelos avaliadores, foram feitas as seguintes perguntas pertencentes ao *System Usability Scale*:

- Eu acho que gostaria de usar esse sistema com frequência;
- Eu acho o sistema desnecessariamente complexo;
- Eu achei o sistema fácil de usar;
- Eu acho que precisaria de ajuda de uma pessoa com conhecimentos técnicos para usar o sistema;
- Eu acho que as várias funções do sistema estão muito bem integradas;
- Eu acho que o sistema apresenta muita inconsistência;
- Eu imagino que as pessoas aprenderão como usar esse sistema rapidamente;
- Eu achei o sistema atrapalhado de usar;
- Eu me senti confiante ao usar o sistema;
- Eu precisei aprender várias coisas novas antes de conseguir usar o sistema.

Cada uma destas questões foi respondida pelos avaliadores com números de um a cinco, sendo “um” correspondente a “Discordo plenamente” e “cinco” como “Concordo plenamente”.

A seguir, na figura 45, é apresentado um exemplo de um dos documentos contendo tanto a avaliação das tarefas quanto das questões:

Avaliador	Mariana
Idade	26
Afinidade com tecnologia	Média
Dispositivo	Motorola moto X 2ª geração
Tarefas	
Acessar menu o sistema	Ok
Acessar pacote de gerenciamento de fornecedores	Ok
Realizar as quatro atividades básicas envolvendo esta entidade	Ok
Acessar informações da agenda de contato para cadastrar um fornecedor	Ok
Acessar pacote de gerenciamento de clientes	Ok
Realizar as quatro atividades básicas envolvendo esta entidade	Ok
Acessar informações da agenda de contato para cadastrar um cliente	Ok
Acessar pacote de gerenciamento de produtos	Ok
Realizar as quatro atividades básicas envolvendo esta entidade	Falha ao cadastrar Produto. Campo unidade não sendo facilmente compreendido
Acessar pacote de gerenciamento de estoques	Falha ao acessar
Realizar as quatro atividades básicas envolvendo esta entidade	Tela não deixa claro onde informar quantidade
Cadastrar nova adição de itens a determinado estoque	-
Realizar a remoção de itens de determinado estoque via interface de estoque	-
Acessar pacote de gerenciamento de vendas	Ok
Realizar as quatro atividades básicas envolvendo esta entidade	Falha ao Gravar
Cadastrar nova adição de itens a determinada venda	Ok
Realizar a remoção de itens de determinada venda	Ok
Compartilhar relatório da venda	Ok
Acessar pacote de gerenciamento de pontos	Ok
Realizar as quatro atividades básicas envolvendo esta entidade	Falha ao informar campo de horário
Verificar pontos adicionados no mapa da tela inicial	Ok
Teste SUS	
1.Eu acho que gostaria de usar esse sistema com frequência	5
2.Eu acho o sistema desnecessariamente complexo	1
3.Eu achei o sistema fácil de usar	5
4.Eu acho que precisaria de ajuda de uma pessoa com conhecimentos técnicos para usar o sistema	1
5.Eu acho que as várias funções do sistema estão muito bem integradas	5
6.Eu acho que o sistema apresenta muita inconsistência	5
7.Eu imagino que as pessoas aprenderão como usar esse sistema rapidamente	3
8.Eu achei o sistema atrapalhado de usar	3
9.Eu me senti confiante ao usar o sistema	5
10.Eu precisei aprender várias coisas novas antes de conseguir usar o sistema	1

Figura 45 – Primeiro teste de Usabilidade aplicado

Para esse primeiro teste é possível verificar que o aplicativo falha em permitir o usuário realizar determinadas tarefas. No acompanhamento do teste foi notado que tais falhas advinham de nomes de campos pouco significativos, entradas de dados demoradas (com necessidade de muitos toques) e falta de tratamento pelo próprio aplicativo de tais entradas do usuário, que em geral, geravam o encerramento automático do aplicativo. Ainda nesse primeiro teste, após sua execução, foi realizada a tomada de respostas para as questões do *System Usability Scale (SUS)*, sendo que as respostas obtidas podem ser conferidas na figura 45. Com base nessas respostas, para que seja possível aferir o grau de usabilidade segundo *SUS,* utilizam-se as seguintes regras da mesma:

- Para perguntas ímpares, subtrair um da resposta dada pelo avaliador;

- Para perguntas pares, fazer a seguinte conta 5 – número da resposta dada pelo avaliador;
- Somar todos os números obtidos;
- Multiplicar o resultado da soma obtido por 2.5.

Portanto, no caso deste primeiro teste a aplicação destas regras resulta em:

- Questão 1: 5 – 1 = 4;
- Questão 2: 5 – 1 = 4;
- Questão 3: 5 – 1 = 4;
- Questão 4: 5 – 1 = 4;
- Questão 5: 5 – 1 = 4;
- Questão 6: 5 – 5 = 0;
- Questão 7: 3 – 1 = 2;
- Questão 8: 5 – 3 = 2;
- Questão 9: 5 – 1 = 4;
- Questão 10: 5 – 1 = 4;
- Soma de todos os números obtidos: 32;
- Multiplicação por 2.5 e resultado final: 80 pontos.

Após a aplicação deste primeiro teste de usabilidade, foi possível avaliar melhorias substanciais que poderiam ser aplicadas ao aplicativo e que foram executadas logo na sequência, como:

- Utilização de *DatePicker e TimePicker* para campos relacionados a data e hora;
- Utilização de *Spinner* para melhor guiar o usuário no cadastro de produto, no campo relacionado a “Unidade”;
- Diversos tratamentos de campos vazios para evitar a mensagem de encerramento prematuro do aplicativo, como por exemplo, na tela de detalhes de produtos, quando um produto não possuía foto;

Após a aplicação destas melhorias, aplicou-se o mesmo teste de usabilidade em avaliadores diferentes e foi possível aferir os seguintes resultados:

- Teste de Usabilidade – Everson;

    - Pontos na escala *SUS:* 70;
    - Indicativo de que não era possível sair do sistema pelo botão voltar (botão triângulo);
    - Quando ao preencher campos, botão de próximo item gerava nova linha, ao invés de realizar salta para próximo item do formulário;
    - Remover necessidade informar hora em campos onde tal informação não fazia sentido;
- Teste de Usabilidade – Tatiana;
    - Pontos na escala *SUS:* 75;
    - Sugestão de transformar campos *AutoComplete* em campos *Spinner* para melhor guiar usuário no cadastro;
- Teste de Usabilidade – Irlete;
    - Pontos na escala *SUS:* 82;
    - Sugestão de que sistema busca informações do produto com base no histórico de vendas para determinado cliente e não no cadastro base do produto.

As figuras destes três testes de usabilidade elencados podem ser verificadas nas Seções 7.3, 7.4 e 7.5. Aqui vale salientar que todos os testes aplicados foram feitos da seguinte forma:

- Instalação do aplicativo nos dispositivos dos avaliadores;
- Verificada e habilitada conexão de dados do dispositivo móvel;
- Habilitada opção de “Localização” do dispositivo;

Muito embora o aplicativo possa ser utilizado sem os dois últimos requisitos, sem a habilitação dos mesmos algumas tarefas que deveriam ser avaliadas nos teste de usabilidade ficaram impossibilitadas.

Por fim, tomando por base a referência da *SUS* de que valores acima de 68 pontos representam um bom resultado, infere-se que, embora o aplicativo apresente algumas falhas e instabilidades há interesse, por parte dos avaliados, nas funcionalidades que o sistema oferece, confirmando assim a validade do aplicativo proposto.

5 CONCLUSÃO

Ao longo deste trabalho foi possível notar as inúmeras ferramentas e tecnologias aplicáveis à criação de aplicativos para dispositivos móveis, seja em soluções para bancos de dados voltados a este ambiente como o *Realm*, *API's (Application Programming Interface)* desenvolvidas pela própria comunidade para facilitar uma tarefa, como por exemplo, a *API Retrofit*, ou até serviços e bibliotecas providos pelos próprios mantenedores do *Android,* a *Google* como, por exemplo, a biblioteca *PdfDocument* ou o serviço de geo-localização. Também foi possível ter contato neste trabalho com uma versão da ferramenta *Scrum* para gerência de projeto individual, que se apresentou deveras necessária na produção tanto do aplicativo quanto da produção deste texto em si, auxiliando principalmente na organização das tarefas a serem executadas semanalmente para cumprimento do objetivo final.

Além do contato com as tecnologias para produção de aplicativos, também foi possível esbarrar nos desafios oferecidos por esse tipo de desenvolvimento, a elencar:

- Formas de modelar o aplicativo para que este seja o mais independente possível de conexão com a rede, principalmente no contexto especifico da área fim deste trabalho, pois nem sempre o vendedor estará em uma região com cobertura de rede móvel;
- Adotar formas de entrada de dados que sejam compatíveis com a velocidade necessária para esta atividade fim, pois no caso de um vendedor, quanto maior o número de clientes visitados maior seu potencial de venda;
- Desenvolver uma interface que seja compreensível e utilizável por usuários de várias faixas etárias e de vários níveis de afinidade tecnológica, afinal, como relembrado por vários professores deste curso, dificilmente estará sendo produzido um software que será utilizado pelos próprios desenvolvedores apenas, sendo assim, este deve ser o mais intuitivo possível.

A necessidade dos dois últimos pontos ficou muito bem explicitada no momento da realização dos vários testes de usabilidade com diversos usuários, onde realmente ficou clara a necessidade de um melhor tratamento destes pontos salientados. Assim, de posse destes entendimentos, também foi possível definir os próximos passos para esse projeto, que são, principalmente, um maior amadurecimento em relação à criação de interfaces de usuário intuitivas e a finalização de todos os outros pontos presentes e ainda não atendidos no *backlog* do produto. Sendo que dentre esses pontos estão, a criação de perspectivas para diversos papéis de usuário e desenvolvimento de uma aplicação servidora *Realm* para utilização de mesmos dados por vários usuários de uma empresa.

Por conta da não criação de um servidor de dados, uma das limitações deste aplicativo seria o não compartilhamento de dados do banco de dados interno entre aparelhos. Além desta limitação existe também a necessidade de conexão com internet para o cálculo de rota de entrega, sendo que a não existência da mesma impede, além do cálculo das rotas, a escolha de endereços de clientes e pontos de entrega via aplicativo botão “Mapa”.

Por fim, salienta-se a oportunidade propiciada por esse projeto de se gerar experiência no desenvolvimento, gerenciamento e teste de aplicativos móveis, que tem como premissa o alcance em larga escala de pessoas.

6 REFERÊNCIAS

ANDREWS, Alex. **Scrum Of One: How to Bring Scrum into your One-Person Operation.** Disponível em: <http://www.scrumguides.org/scrum-guide.html>. Acesso em: 03 jul. 2017.

ANDROID. **Android**. Disponível em: <https://www.android.com/intl/pt-BR_br/>. Acesso em: 28 jun. 2017.

ANDROID STUDIO. **ADT Plugin (UNSUPPORTED).** Disponível em: <https://developer.android.com/studio/tools/sdk/eclipse-adt.html>. Acesso em: 29 jun. 2017.

_____.**Baixar o Android Studio e as SDK Tools.** Disponível em: <https://developer.android.com/studio/index.html?hl=pt-br#features>. Acesso em: 29 jun. 2017.

ARAÚJO, Camila C. et al. **Gerenciamento de Banco de Dados: Análise comparativa de SGBD'S.** Disponível em: <http://www.devmedia.com.br/gerenciamento-de-banco-de-dados-analise-comparativa-de-sgbds/30788>. Acesso em: 30 jun. 2017.

BELL, Donald. **Fundamentos básicos de UML: O Diagrama de Classes.** Disponível em: <https://www.ibm.com/developerworks/br/rational/library/content/RationalEdge/sep04/bell/index.html>. Acesso em: 27 jun. 2017.

Burnette, Ed. **Patrick Brady dissects Android.** Disponível em: <http://www.zdnet.com/article/patrick-brady-dissects-android/>. Acesso em: 28 jun. 2017.

CÉSAR, Paulo. **Primeiros passos com serviçoes REST**. Disponível em: <http://www.devmedia.com.br/primeiros-passos-com-os-servicos-rest/28912/>. Acesso em: 02 jul. 2017.

COLÉGIO WEB. **Evolução do comércio – Das aldeias aos dias de hoje.** Disponível em: <https://www.colegioweb.com.br/historia/evolucao-comercio-das-aldeias-aos-dias-de-hoje.html>. Acesso em: 08 jun. 2017.

FERREIRA, Kecia A.M; DRUMOND, Elayne C. Normas ISO para Usabilidade**.** Disponível em: <http://homepages.dcc.ufmg.br/~clarindo/arquivos/disciplinas/eu/material/seminarios-alunos/normas-iso-kecia-elayne.pdf>. Acesso em: 14 out. 2017.

GAMMA, Erich et al. **Padrões de Projeto - Soluções Reutilizaveis de Software Orientado a Objetos**. São Paulo: Bookman, 2006.

GATTO, Elaine C. **ISSO/IEC 9241-11.** Disponível em: <https://pt.slideshare.net/elainececiliagatto/isoiec-924111>. Acesso em: 14 out. 2017.

GOOGLE DEVELOPERS. **Google Maps APIs.** Disponível em: <https://developers.google.com/maps/?hl=pt-br>. Acesso em: 02 jul. 2017.

_____. **Data Binding Library.** Disponível em: <https://developer.android.com/topic/libraries/data-binding/index.html?hl=pt-br>. Acesso em: 01 ago. 2017.

_____.**PdfDocument.** Disponível em: <https://developer.android.com/reference/android/graphics/pdf/PdfDocument.html>. Acesso em: 14 out. 2017.

GOOGLE PLAY. **Erp – Apps para Android no Google Play**. Disponível em: <https://play.google.com/store/search?q=erp&c=apps>. Acesso em: 26 jun. 2017.

_____. **MiniERP Estoque, Vendas, NFe, SAT– Apps para Android no Google Play.** Disponível em: <https://play.google.com/store/apps/details?id=br.com.sistemaids.idsmobilevendas>. Acesso em: 26 jun. 2017.

_____. **Gestão e controle de pedidos, vendas e clientes – Apps para Android no Google Play.** Disponível em: <https://play.google.com/store/apps/details?id=br.com.meuspedidos&hl=pt>. Acesso em: 26 jun. 2017.

_____. **SuasVendas – Apps para Android no Google Play.** Disponível em: <https://play.google.com/store/apps/details?id=ys.mobile.droid.suasvendas&hl=pt>. Acesso em: 26 jun. 2017.

GSON. **GitHub – google/gson.** Disponível em: <https://github.com/google/gson >. Acesso em: 02 jul. 2017.

INTERNET LIVE STATS. **Brazil Internet Users**. Disponível em: <http://www.internetlivestats.com/internet-users/brazil/>. Acesso em: 04 jun. 2017.

JAVA. **Java.com.** Disponível em: <https://www.java.com/pt_BR/>. Acesso em: 28 jun. 2017.

JETBRAINS. **IntelliJ IDEA: The Java IDE for Professional Developers by JetBrains.** Disponível em: <https://www.jetbrains.com/idea/>. Acesso em: 29 jun. 2017.

JORNALDEV. **Mediator Design Pattern in Java.** Disponível em: <http://www.journaldev.com/1730/mediator-design-pattern-java>. Acesso em: 23 jul. 2017.

JUNIOR, José Carlos R. **Software de Gestão de Vendas: o que é e por que ter um?** Disponível em: < https://conube.com.br/blog/software-de-gestao-de-vendas/>. Acesso em: 02 dez. 2017.

KLADIS, Constantin M; FREITAS, Henrique M.R de. **O gerente nas organizações: funções, limitações e estilos decisórios**. Disponível em: <http://www.ufrgs.br/gianti/files/artigos/1996/1996_043_rev_sh.pdf>. Acesso em: 08 jun. 2017.

NORDIC APIS. **Rest vs Soap.** Disponível em: <http://nordicapis.com/rest-vs-soap-nordic-apis-infographic-comparison/>. Acesso em: 02 jul. 2017.

PAPYRUS. **Papyrus.** Disponível em: <https://eclipse.org/papyrus/>. Acesso em: 28 jun. 2017.

REALM. **Realm: Create reactive mobile apps in a fraction of the time.** Disponível em: <https://realm.io/>. Acesso em: 30 jun. 2017.

_____. **Build Better Apps, Faster, with Realm**. Disponível em: <https://www2.realm.io/whitepaper/realm-overview>. Acesso em: 02 jul. 2017.

RETROFIT. **Retrofit.** Disponível em: <http://square.github.io/retrofit/>. Acesso em: 02 jul. 2017.

SAURO, Jeff. **MeasuringU: Measuring Usability with the System Usability Scale (SUS).** Disponível em: <https://measuringu.com/sus/>. Acesso em: 15 out. 2017.

SCRUM GUIDE. **Scrum Guide.** Disponível em: <http://www.scrumguides.org/scrum-guide.html>. Acesso em: 03 jul. 2017.

SEBRAE. **10 dicas para começar bem.** Disponível em: <http://www.sebrae.com.br/sites/PortalSebrae/artigos/10-dicas-para-comecar-bem,c098d8e04c09d410VgnVCM1000003b74010aRCRD#this>. Acesso em: 06 jun. 2017.

_____. **Importância da gerência de vendas.** Disponível em: <http://www2.rj.sebrae.com.br/boletim/importancia-da-gerencia-de-vendas/>. Acesso em: 08 jun. 2017.

_____. **A gestão de produção é essencial para a sua empresa crescer.** Disponível em: <https://www.sebrae.com.br/sites/PortalSebrae/artigos/a-gestao-de-producao-e-essencial-para-a-sua-empresa-crescer,4b73438af1c92410VgnVCM100000b272010aRCRD> . Acesso em: 08 jun. 2017.

SIMON, Herbert A. **Comportamento Administrativo: Estudo dos Processos Decisórios nas Organizações Administrativas**. Rio de Janeiro: Aliança para o Progresso, 1965.311p.

SOUZA, Rainer. **As atividades comerciais.** Disponível em: <http://brasilescola.uol.com.br/historia/historia-do-comercio.htm>. Acesso em: 08 jun. 2017.

STATISTA. **Brazil: retail e-commercesales 2021**. Disponível em <https://www.statista.com/statistics/289746/brazil-retail-e-commerce-sales/.04/06/2017>. Acesso em: 04 jun. 2017.

_____. **B2C e-commerce revenue in Brazil 2017**. Disponível em: <https://www.statista.com/statistics/222115/online-retail-revenue-in-brazil-projection/>. Acesso em: 04 jun. 2017.

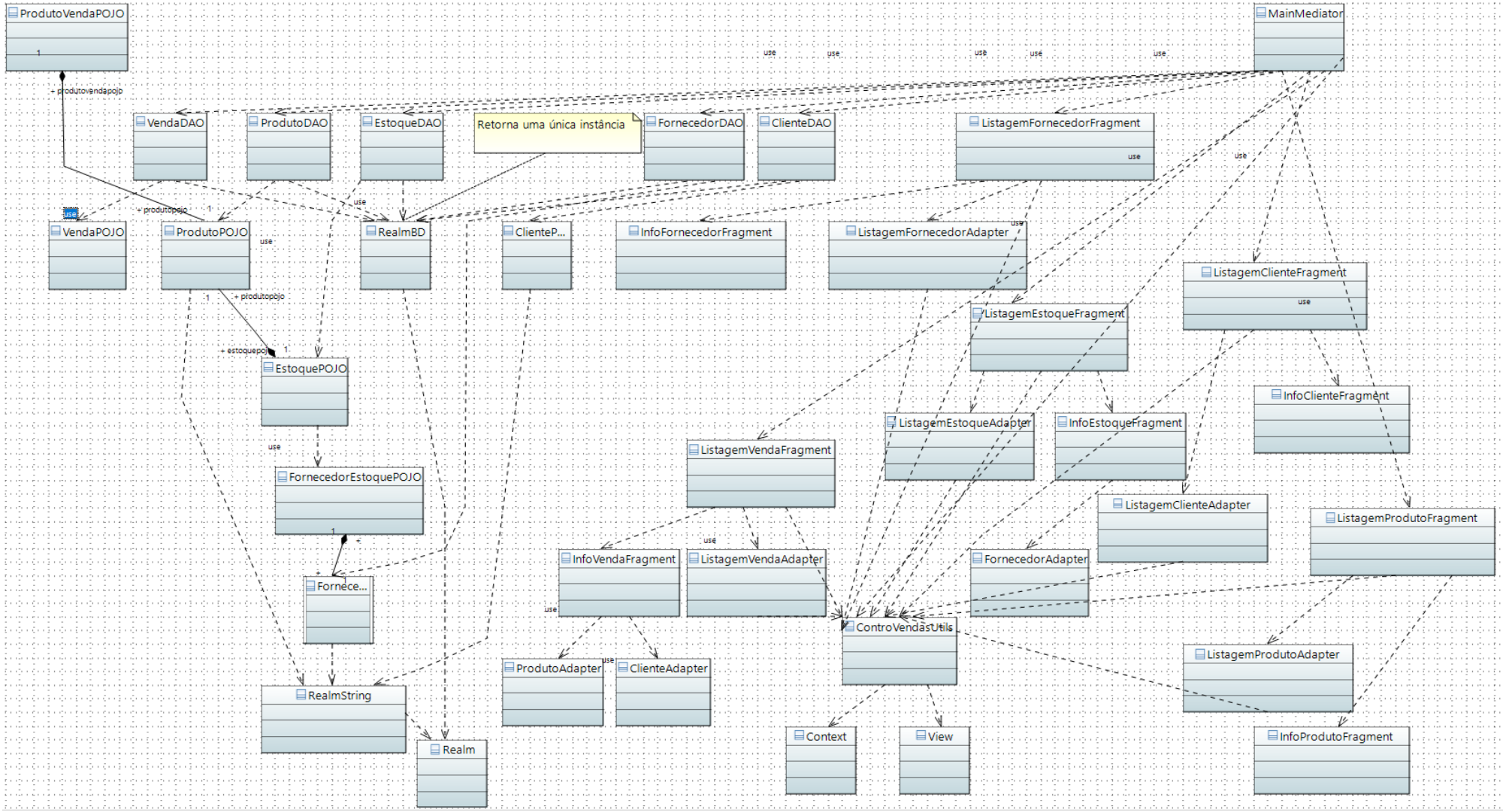
_____.**Mobile OS Market share 2017.** Disponível em: <https://www.statista.com/statistics/266136/global-market-share-held-by-smartphone-operating-systems/>. Acesso em: 28 jun. 2017.

_____.**App stores: number of apps in leading app stores 2017.** Disponível em: <https://www.statista.com/statistics/276623/number-of-apps-available-in-leading-app-stores//>. Acesso em: 28 jun. 2017.

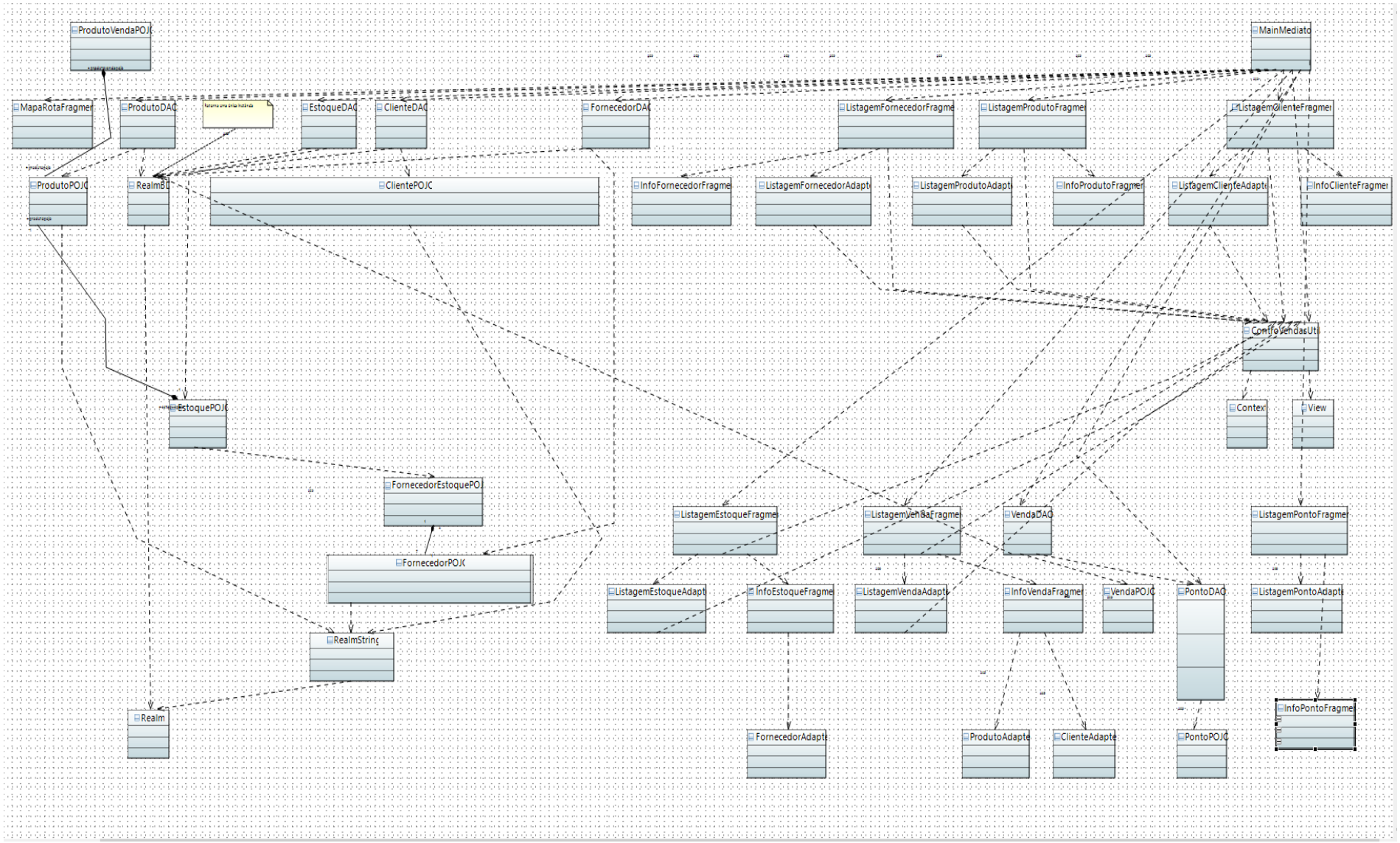
UML. **Welcome to UML WebSite.** Disponível em: < http://www.uml.org/>. Acesso em: 27 jun. 2017.

7 APÊNDICES

7.1 Diagrama de classes sprint dez



7.2 Diagrama de classes *sprint* treze



7.3 Teste de Usabilidade 2

Avaliador	Everson
Idade	35
Afinidade com tecnologia	Alta
Dispositivo	Motorola Moto G 5
Tarefas	
Acessar menu o sistema	Na tela inicial “mapa” ao tentar sair pelo teclado de retornar, aplicativo não é encerrado. Campo “settings” sem função
Acessar pacote de gerenciamento de fornecedores	Ok – sem observações
Realizar as quatro atividades básicas envolvendo esta entidade	Cadastro de fornecedores: Tela não corre durante o preenchimento. Sugestão: endereço buscar pelo localizador do google maps. Habilitar a função discar a partir do contato.
Acessar informações da agenda de contato para cadastrar um fornecedor	Ok – sem observações
Acessar pacote de gerenciamento de clientes	Ok – sem observações
Realizar as quatro atividades básicas envolvendo esta entidade	Cadastro de cliente: 1 - Tela não corre durante o preenchimento. Sugestão: cadastro de cpf e cnpf salvamento dos dados permitir caracteres especiais (. - /) Habilitar a função discar a partir do cliente.
Acessar informações da agenda de contato para cadastrar um cliente	Ok – sem observações
Acessar pacote de gerenciamento de produtos	Ok – sem observações
Realizar as quatro atividades básicas envolvendo esta entidade	Cadastro de produto: Sugestão: definir uma unidade padrão de peso. Permitir adição de fotos a partir de registros já presentes no aparelho Na tela geral do produto mostra ao lado do nome do produto a unidade relativa ao cadastro. (ex. posso vender um vidro de palmito ou uma caixa. Como gerenciar isso?) Ou no cadastro permitir somente itens unitários.
Acessar pacote de gerenciamento de estoques	Ok – sem observações
Realizar as quatro atividades básicas envolvendo esta entidade	Entrada do estoque: ao incluir um fornecedor, ao clicar na tecla de “enter” ao invés de seguir para o próximo campo o cursor cria mais uma linha. Tela não corre durante preenchimento Tela apresenta dois campos qtde, pq? Cadastro não muito claro. Ao clicar em adicionar com campo sem preenchimento, o aplicativo é fechado. Estando em quantidade, ao clicar em “enter” o curso segue direto para fornecedor, pula o campo “data compra” Ao fazer uma atualização de um cadastro já existente, aplicativo fecha. Para um mesmo produto adquirido os valores não são somados. Campo de valor não aceita valor de

	centavos. Sugestão: Campo hora pode ser removido
Cadastrar nova adição de itens a determinado estoque	Para um mesmo produto adquirido os valores não são somados ao estoque existente.
Realizar a remoção de itens de determinado estoque via interface de estoque	Sistema travou
Acessar pacote de gerenciamento de vendas	Ok – sem observações
Realizar as quatro atividades básicas envolvendo esta entidade	Vendas: ao clicar na tecla de "enter" ao invés de seguir para o próximo campo o cursor cria mais uma linha. Sistema permite realizar venda com quantidades maiores que as existentes em estoque. Se realizar venda de produto não cadastrado aplicação trava. Sugestão: ao sair da tela salvar de forma automática para não perder o registro da venda.
Cadastrar nova adição de itens a determinada venda	Ok – sem observações
Realizar a remoção de itens de determinada venda	Ok – sem observações
Compartilhar relatório da venda	Sugestão. No ícone compartilhar deveria abrir somete a tela de envio da "nf" para o referido cliente. No relatório incluir demais dados do cliente telefone, cpf etc.
Acessar pacote de gerenciamento de pontos	Ok – sem observações
Realizar as quatro atividades básicas envolvendo esta entidade	Não é clara a aplicação desta tela/função
Verificar pontos adicionados no mapa da tela inicial	Ok – sem observações
Teste SUS	5
1.Eu acho que gostaria de usar esse sistema com frequência	4
2.Eu acho o sistema desnecessariamente complexo	2
3.Eu achei o sistema fácil de usar	4
4.Eu acho que precisaria de ajuda de uma pessoa com conhecimentos técnicos para usar o sistema	2
5.Eu acho que as várias funções do sistema estão muito bem integradas	2
6.Eu acho que o sistema apresenta muita inconsistência	3
7.Eu imagino que as pessoas aprenderão como usar esse sistema rapidamente	5
8.Eu achei o sistema atrapalhado de usar	4
9.Eu me senti confiante ao usar o sistema	5
10.Eu precisei aprender várias coisas novas antes de conseguir usar o sistema	1

7.4 Teste de Usabilidade 3

Avaliador	Irlete
Idade	61
Afinidade com tecnologia	Média-Baixa
Dispositivo	LG K8 LTE
Tarefas	
Acessar menu o sistema	Ok
Acessar pacote de gerenciamento de fornecedores	Ok
Realizar as quatro atividades básicas envolvendo esta entidade	Ok
Acessar informações da agenda de contato para cadastrar um fornecedor	Ok
Acessar pacote de gerenciamento de clientes	Ok
Realizar as quatro atividades básicas envolvendo esta entidade	Ok
Acessar informações da agenda de contato para cadastrar um cliente	Ok
Acessar pacote de gerenciamento de produtos	Ok
Realizar as quatro atividades básicas envolvendo esta entidade	Cadastro realizado, porém buscar foto da galeria. Fazer verificação de preenchimento dos campos
Acessar pacote de gerenciamento de estoques	Ok
Realizar as quatro atividades básicas envolvendo esta entidade	Ok
Cadastrar nova adição de itens a determinado estoque	Ok
Realizar a remoção de itens de determinado estoque via interface de estoque	Ok
Acessar pacote de gerenciamento de vendas	Ok
Realizar as quatro atividades básicas envolvendo esta entidade	Ok
Cadastrar nova adição de itens a determinada venda	Ok
Realizar a remoção de itens de determinada venda	Ok
Compartilhar relatório da venda	Venda compartilhada, porém no relatório poderia aparecer data da entrega
Acessar pacote de gerenciamento de pontos	Ok
Realizar as quatro atividades básicas envolvendo esta entidade	Ok
Verificar pontos adicionados no mapa da tela inicial	Ok
Teste SUS	
1.Eu acho que gostaria de usar esse sistema com frequência	5
2.Eu acho o sistema desnecessariamente complexo	2
3.Eu achei o sistema fácil de usar	3
4.Eu acho que precisaria de ajuda de uma pessoa com conhecimentos técnicos para usar o sistema	4
5.Eu acho que as várias funções do sistema estão muito bem integradas	5
6.Eu acho que o sistema apresenta muita inconsistência	1
7.Eu imagino que as pessoas aprenderão como usar esse sistema rapidamente	5
8.Eu achei o sistema atrapalhado de usar	2
9.Eu me senti confiante ao usar o sistema	5
10.Eu precisei aprender várias coisas novas antes de conseguir usar o sistema	1

7.5 Teste de Usabilidade 4

Avaliador	Tatiana
Idade	34
Afinidade com tecnologia	Média
Dispositivo	Motorola Moto G 5
Tarefas	
Acessar menu o sistema	Ok
Acessar pacote de gerenciamento de fornecedores	Ok
Realizar as quatro atividades básicas envolvendo esta entidade	Ok
Acessar informações da agenda de contato para cadastrar um fornecedor	Ok
Acessar pacote de gerenciamento de clientes	Ok
Realizar as quatro atividades básicas envolvendo esta entidade	Ok
Acessar informações da agenda de contato para cadastrar um cliente	Ok
Acessar pacote de gerenciamento de produtos	Ok
Realizar as quatro atividades básicas envolvendo esta entidade	Ok
Acessar pacote de gerenciamento de estoques	Dificuldade em cadastrar produto, fiquei na duvida se era quantidade ou valor
Realizar as quatro atividades básicas envolvendo esta entidade	Ok
Cadastrar nova adição de itens a determinado estoque	Ok
Realizar a remoção de itens de determinado estoque via interface de estoque	Ok
Acessar pacote de gerenciamento de vendas	Ok
Realizar as quatro atividades básicas envolvendo esta entidade	Ok
Cadastrar nova adição de itens a determinada venda	Ok
Realizar a remoção de itens de determinada venda	Ok
Compartilhar relatório da venda	Ok
Acessar pacote de gerenciamento de pontos	Ok
Realizar as quatro atividades básicas envolvendo esta entidade	Ok
Verificar pontos adicionados no mapa da tela inicial	Ok
Teste SUS	
1.Eu acho que gostaria de usar esse sistema com frequência	5
2.Eu acho o sistema desnecessariamente complexo	3
3.Eu achei o sistema fácil de usar	3
4.Eu acho que precisaria de ajuda de uma pessoa com conhecimentos técnicos para usar o sistema	3
5.Eu acho que as várias funções do sistema estão muito bem integradas	5
6.Eu acho que o sistema apresenta muita inconsistência	3
7.Eu imagino que as pessoas aprenderão como usar esse sistema rapidamente	5
8.Eu achei o sistema atrapalhado de usar	3
9.Eu me senti confiante ao usar o sistema	5
10.Eu precisei aprender várias coisas novas antes de conseguir usar o sistema	1