Universidade Federal de Minas Gerais
Instituto de Ciências Exatas
Departamento de Ciências da Computação

MILTON FERREIRA LIMA FILHO

PLATAFORMA PARA CONSULTORES INDEPENDENTES SEGUINDO A ESPECIFICAÇÃO JAVA ENTERPRISE EDITION - JEE

Belo Horizonte
2018

Universidade Federal de Minas Gerais
Instituto de Ciências Exatas
Departamento de Ciências da Computação
Especialização em Informática: Ênfase: Engenharia de Software

PLATAFORMA PARA CONSULTORES INDEPENDENTES SEGUINDO A ESPECIFICAÇÃO JAVA ENTERPRISE EDITION - JEE

por

MILTON FERREIRA LIMA FILHO

Monografia de Final de Curso
CEI-ES-0xxx-T20-2016-01

Prof. ... *[Ítalo Fernando Scota Cunha]*
Orientador(a)

Belo Horizonte
2018

MILTON FERREIRA LIMA FILHO

PLATAFORMA PARA CONSULTORES INDEPENDENTES SEGUINDO A ESPECIFICAÇÃO JAVA ENTERPRISE EDITION - JEE

Monografia apresentada ao Curso de Especialização em Informática do Departamento de Ciências Exatas da Universidade Federal de Minas Gerais, como requisito parcial para a obtenção do grau de Especialista em Informática.

Área de concentração: Desenvolvimento e especificação de Software

Orientador(a): Prof. Italo Fernando Scota Cunha

Belo Horizonte
2018

UNIVERSIDADE FEDERAL DE MINAS GERAIS

INSTITUTO DE CIÊNCIAS EXATAS
DEPARTAMENTO DE CIÊNCIA DA COMPUTAÇÃO
ESPECIALIZAÇÃO EM INFORMÁTICA: ÁREA DE CONCENTRAÇÃO ENGENHARIA DE SOFTWARE

PLATAFORMA PARA CONSULTORES INDEPENDENTES SEGUINDO A ESPECIFICAÇÃO JAVA ENTERPRISE EDITION - JEE

MILTON FERREIRA LIMA FILHO

Monografia apresentada aos Senhores:

Prof. Ítalo Fernando Scotá Cunha
Orientador
DCC - ICEx - UFMG

Prof. Roberto da Silva Bigonha
DCC - ICEx - UFMG

Belo Horizonte, 13 de setembro de 2018

Ficha catalográfica elaborada pela Biblioteca do ICEx - UFMG

Lima Filho, Milton Ferreira.

L732p Plataforma para consultores independentes seguindo a especificação java enterprise edition - jee / Milton Ferreira Lima Filho. Belo Horizonte, 2018.
51 f.: il.; 29 cm.

Monografia (especialização) - Universidade Federal de Minas Gerais – Departamento de Ciência da Computação.

Orientador: Italo Fernando Scotá Cunha.

1. Computação-Teses 2. Engenharia de software. 3. Java (Linguagem de programação de computador).
4. Consultores de investimentos. I. Orientador. II. Título.

CDU 519.6*32(043)

À Deus,
aos professores,
aos colegas de curso e
aos meus familiares e
especialmente minha noiva,
dedico este trabalho.

AGRADECIMENTOS

Inicialmente quero agradeço a Deus, pelos dons recebidos.

Agradeço aos meus pais, pelo amor incondicional.

Aos meus professores, pelos conhecimentos adquiridos.

A minha amada Michelle, pelas dicas e idéias ao longo do trabalho.

E finalmente aos colegas de curso pela convivência e trocas de conhecimentos.

"Viva como se fosse morrer amanhã.
Aprenda como se fosse viver para sempre”
Mahatma Gandhi

RESUMO

O principal objetivo do trabalho é especificar e desenvolver uma plataforma de auxílio aos consultores independentes de produtos, tais como: Natura, Rommanel, Jequiti, Avon, entre outros. O projeto é dividido em duas partes principais, uma parte sendo acessado pela Internet através da Web e a outra parte um aplicativo móvel Android.

A aplicação Web tem funcionalidades que auxilia as consultoras com as suas atividades de controle de vendas, controle dos clientes, controle dos pedidos dos seus produtos. O aplicativo Android permite aos clientes visualizarem os produtos de seu vendedor, realizar o pedido de produtos e acompanhar os seus pedidos de compra.

Na fase de levantamento de requisitos realizou-se reuniões com uma vendedora de produtos, que colaborou com informações e necessidades de funcionalidades que o sistema deveria contemplar.

A arquitetura do módulo Web segue o *padrão de projeto Model View Controller (MVC)* que estrutura o código fonte de forma a separar as camadas da aplicação. A camada *Model* é a camada com as entidades do domínio da aplicação, a *View* é a camada responsável pelos códigos de apresentação, e a camada *Controller* é responsável por intermediar a comunicação entre as camadas, *View* e *Model*.

O arcabouço Scrum e práticas de metodologias ágeis foram utilizados para auxiliar o processo de desenvolvimento incremental do software. As Sprints foram desenvolvidas a partir dos *backlogs* gerados pelas estórias de usuários levantadas com uma consultora de vendas, um dos *stakeholders* do projeto.

Para auxílio ao desenvolvimento das especificações foram utilizados cartões no formato de post-it para definir as tarefas e suas prioridades com a utilização de *Kanban* e também utilizou-se *Unified Modeling Language - UML* para facilitar a comunicação com os *stakeholders* e o desenvolvimento da documentação da arquitetura do software.

Palavras-chave: Software, Desenvolvimento de software, Processos de desenvolvimento de software.

ABSTRACT

The main objective of this work is to specify and develop a platform to assist independent product consultants, such as: Natura, Rommanel, Jequiti, Avon, among others. The project is divided into two main parts, one part being a Web service accessed through the Internet and the second part a mobile Android application.

The Web application has the functionalities that assist the consultants with their activities of sales control, customers management, and bookkeeping orders of their products. The Android app allows customers to view their vendor's products, place orders for products, and track their purchase orders.

During the requirements specification phase, meetings were held with a product saleswoman, who collaborated with information and functional needs that the system should contemplate.

The Web module architecture follows the Model View Controller (MVC) design pattern that structures to separate the application layers. The Model layer is the layer with the entities of the application domain, the View is the layer responsible for the presentation and user interface, and the Controller layer is responsible for mediating the communication between the View and Model layers.

The Scrum framework and practices of agile methodologies were used to support the incremental software development process. The Sprints were developed from the backlogs generated by the user stories raised with a sales consultant, one of the project stakeholders.

To assist in the development of the specifications, post-it cards were used to define the tasks and their priorities with the use of Kanban. The Unified Modeling Language (UML) was used to facilitate the communication with the stakeholders and the development of the documentation of the software architecture.

Keywords*:* Software, Development of software, Processes for software development.

LISTA DE FIGURAS

LISTA DE SIGLAS

MVC	Model-View-Controller
OO	Orientado(a) por Objetos
POO	Programação Orientada por Objetos
TAD	Tipo Abstrato de Dados
JSF	Java Server Faces
JPA	Java Persistence API
API	Application Programming Interface
UML	Unified Modeling Language
EAP	Estrutura Analítica do Projeto
JCP	Java Community Process
WBS	Work Breakdown Structure
EJB	Enterprise Java Beans
JAX-RS	Java API for RESTful Web Services
CDI	Context Dependency Injection
JAAS	Java Authentication and Authorization Service
JEE	Java Enterprise Edition
IDE	Integrated Development Environment
JDK	Java Developer Kit
SDK	Software Developer Kit
DER	Diagrama Entidade Relacionamento
REST	Representational State Transfer
HTTP	Hypertext Transfer Protocol
JSON	JavaScript Object Notation

SUMÁRIO

1 INTRODUÇÃO

É indiscutível a importância atribuída aos computadores e softwares como ferramentas indispensáveis para a gestão de atividades econômicas e comerciais. Desta forma, podemos observar este fato com alguns exemplos do nosso cotidiano, tais como: em supermercados, com a utilização dos softwares nos *caixas*, em hospitais com os terminais de atendimento e nos aeroportos e cinemas temos os terminais de autoatendimento.

Vale ressaltar, que no primeiro semestre de 2018 no Brasil houve várias pesquisas por softwares como mostra a FIG. 1 (Google, 2018)

Figura 1 - Pesquisa por Softwares - (https://trends.google.com.br, 2018)

No Brasil as pesquisas por softwares, estão em todo o território nacional. De acordo, com o Google Trends todos os estados brasileiros pesquisaram por software em algum momento este ano, como mostra a FIG. 2.

Figura 2 - Distribuição de pesquisas por software entre estados do Brasil - (https://trends.google.com.br, 2018)

Somente no Brasil no ano de 2018 foram realizadas várias pesquisas com relação ao trabalho autônomo. Como pode ser visualizado no gráfico da FIG. 3.

Figura 3 - Pesquisa por Trabalhar Autonomo - (https://trends.google.com.br, 2018)

Os estados onde o maior número de pesquisas ocorreram foram os seguintes, conforme demonstrado na FIG. 4.

Figura 4 - Estados com Pesquisa por Trabalho Autônomo - (https://trends.google.com.br, 2018)

O sudeste do Brasil é onde ocorreram a maioria das pesquisas por Trabalhar Autônomo no primeiro semestre de 2018.

Logo, é possível observar indicativos que existe um mercado para a utilização de um software, que visa auxiliar os profissionais autônomos que busquem complementarem sua renda com vendas, além, operacionalizar a gestão do próprio negócio.

Com uma pesquisa realizada com uma amostragem de consultoras, consegui alguns dados interessantes na aplicação de um questionário. O questionário foi construído pensando no perfil de pessoas que trabalham realizando venda de produtos. Foram criadas perguntas tendo o foco gestão e a tecnologia da informação para auxiliar o négocio.

A FIG. 5 mostra que 13,3% não realizam nenhum gerenciamento dos produtos que eles revendem.

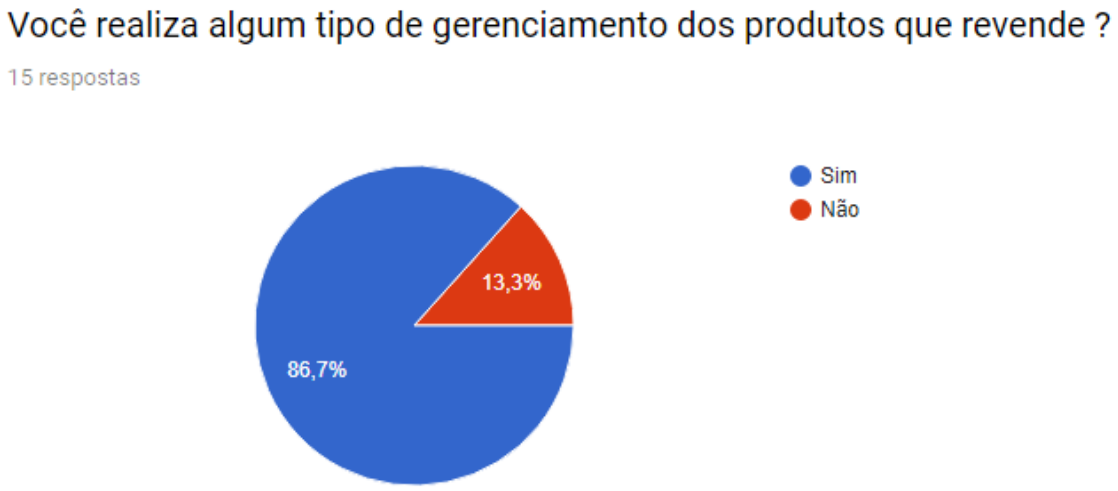


Figura 5 - Pesquisa de Mercado

A maioria das revendedoras de produtos estão preocupadas em gerenciar seus produto e suas vendas.
Uma amostragem de 80% dos revendedores realizam algum tipo de gerenciamento do produtos que eles revendem.

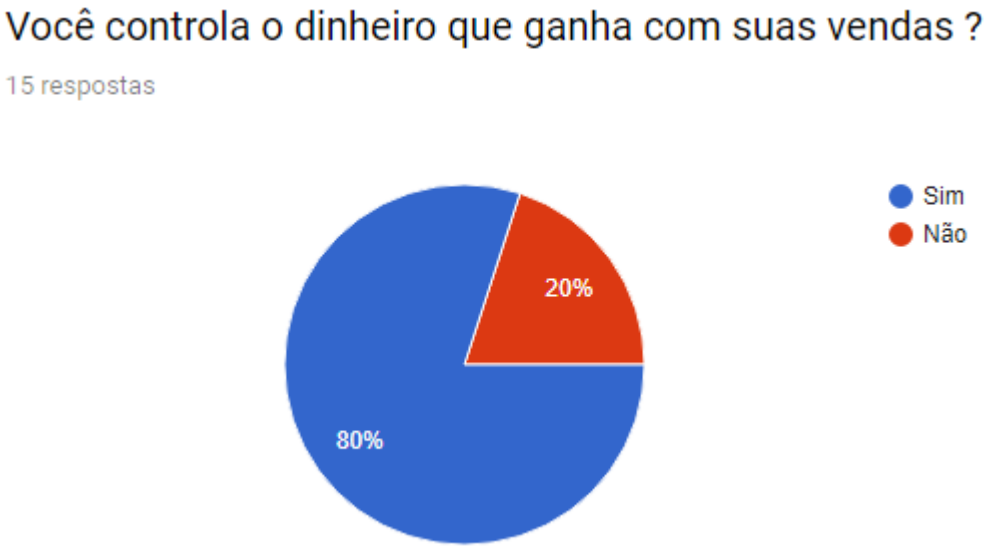


Figura 6 - Pesquisa de Mercado

Mais de 50% das consultoras de vendas estão preocupadas com o gerenciamento do seu dinheiro. Controlando o dinheiro que ganha com as vendas realizadas.

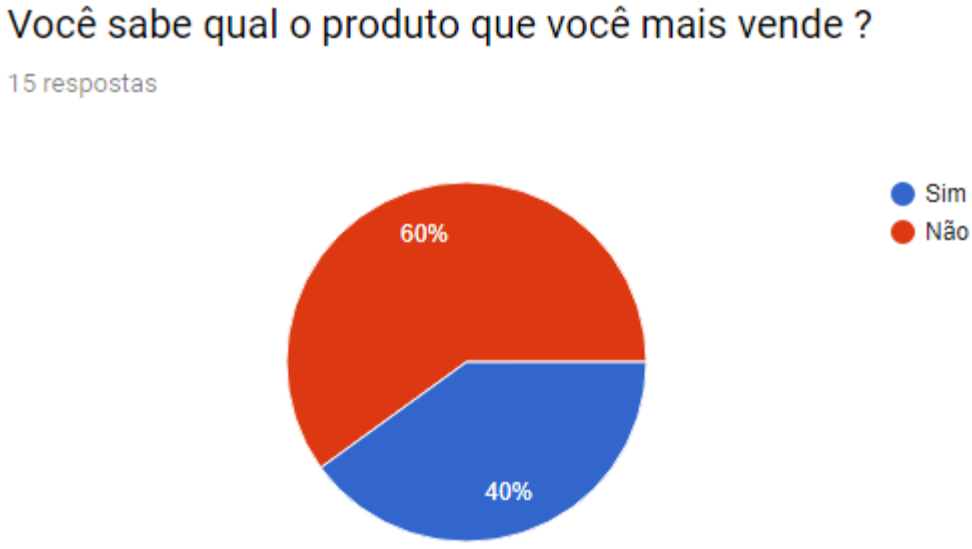


Figura 7 - Pesquisa de Mercado

Mais de 50% das consultoras não sabem quais os produtos que elas mais vendem.

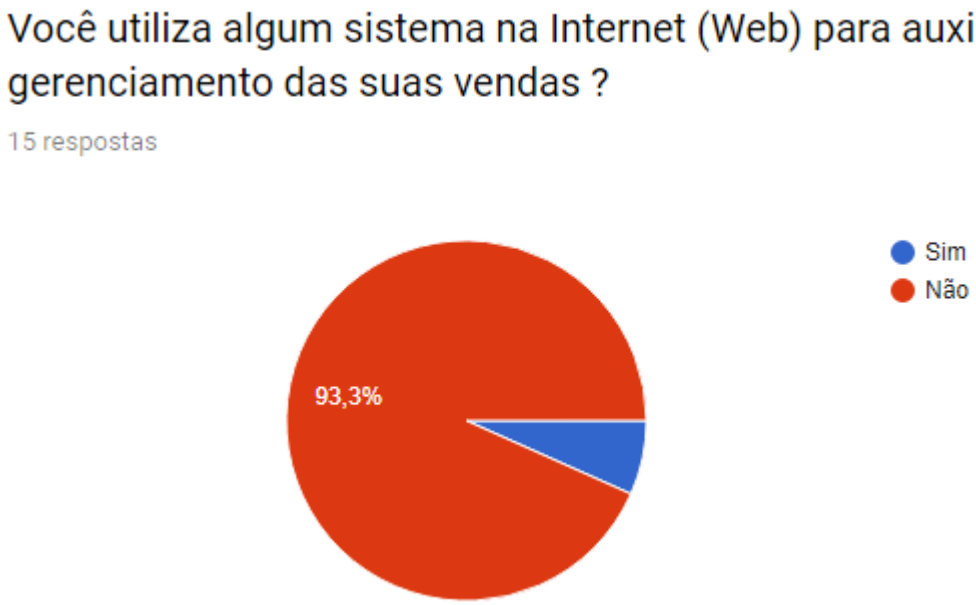


Figura 8 - Pesquisa de Mercado

Mais de 80% dos consultores não utilizam algum software para auxiliá-los nos gerenciamentos dos seus clientes, produtos e vendas.

A pesquisa completa pode ser acessada no seguinte link (https://goo.gl/forms/e8kltBBpBWfXkVf83).

Neste contexto, identificou-se a necessidade de desenvolver um sistema (software) para auxiliar os consultores de produtos.

O projeto contemplou a construção de módulos para auxiliar no controle e gestão dos clientes, produtos e vendas.

Alguns pacotes de trabalho foram definidos para o desenvolvimento e implementação do projeto como pode-se visualizar na Estrutura Analítica do Projeto - EAP na FIG. 9.

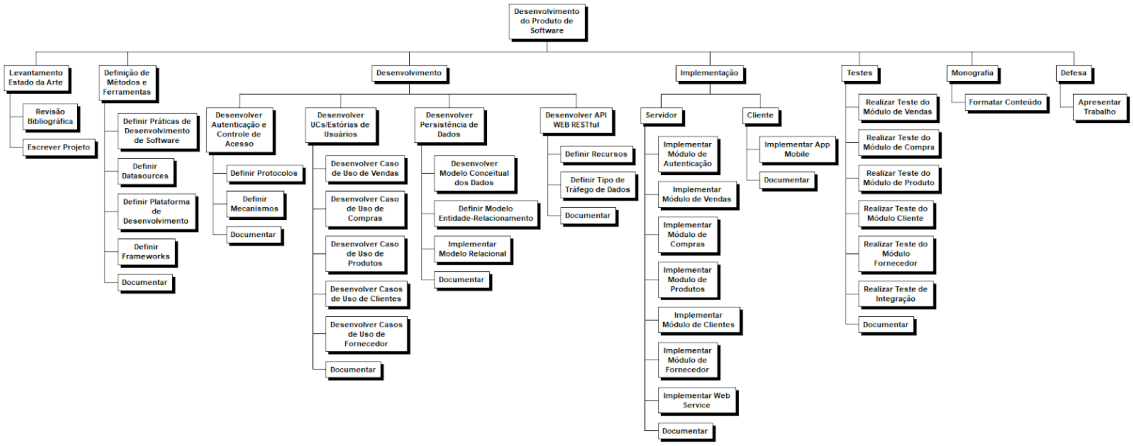


Figura 9 - Estrutura Analítica do Projeto - EAP

1.1 Objetivo

A proposta deste trabalho é implementar uma plataforma para consultores independentes com as seguintes funcionalidades: Controle de produtos; Controle de clientes; Controle de revendedores; Autenticação e Controle de Acesso. As demais funcionalidades serão incrementalmente desenvolvidas posteriormente, estendendo algumas classes do projeto. A implementação desta plataforma segue as especificações Java publicadas na JCP - Java Community Process (https://jcp.org, 2018).

1.2 Justificativa

O mercado de profissionais trabalhando para compor a renda cresce a cada dia no Brasil. Na pesquisa que realizei constatei que 100% dos consultores utilizam um caderno de anotações para auxiliá-los no gerenciamento dos produtos vendidos. Como mostra a FIG. 10.

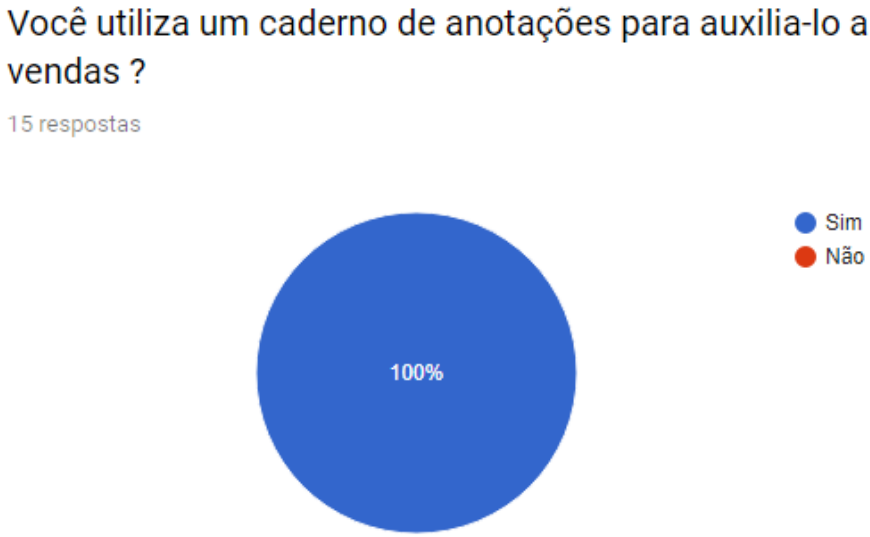


Figura 10 - Pesquisa de Mercado

Com a pesquisa realizada com uma amostragem de consultores podemos perceber de acordo com a FIG. 10, 100% gostariam de utilizar um software que os auxiliasse na gestão dos produtos, vendas e clientes.

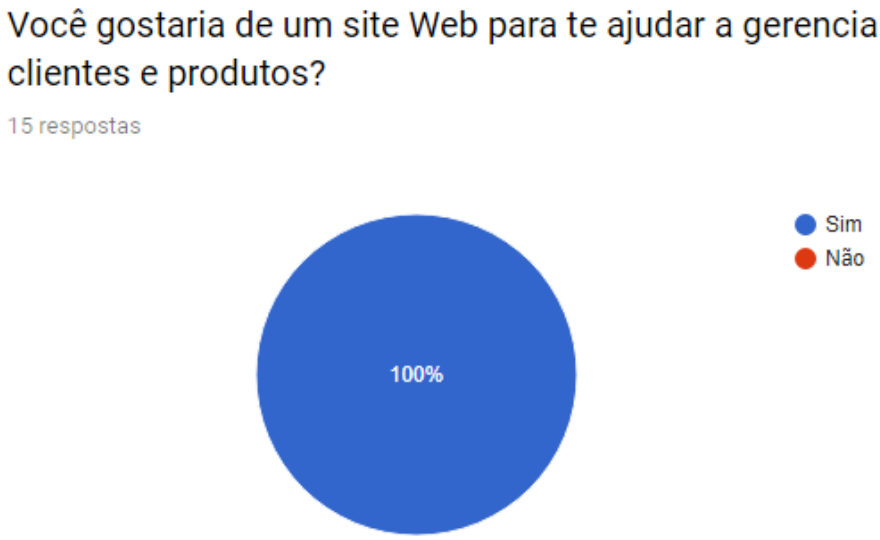


Figura 11 - Pesquisa de Mercado

Porém, apenas 66,7% estariam dispostos a pagarem por esse serviço. Como mostrado na FIG. 12.

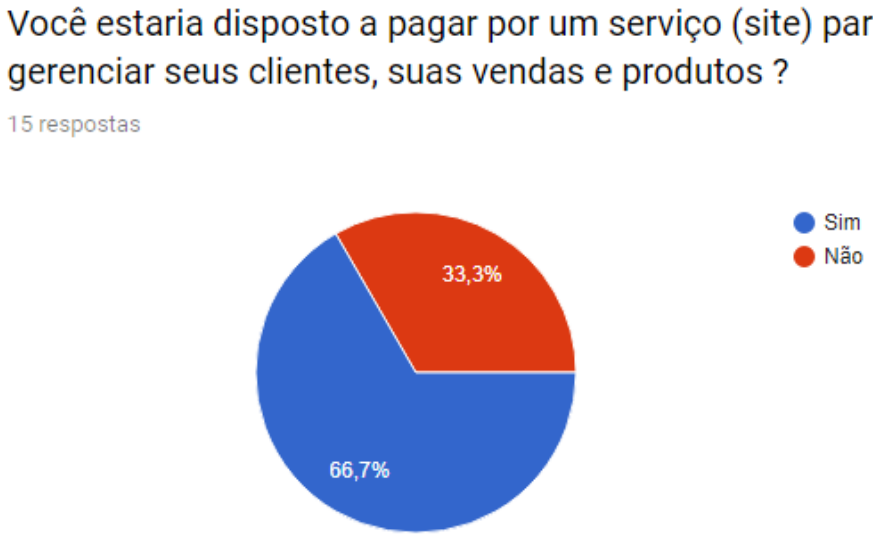


Figura 12 - Pesquisa de Mercado

Esta pesquisa pode ser acessada no seguinte endereço (https://goo.gl/forms/LMMCd4cm7Ks6Bmak2).

Com a realização da pesquisa levantamos alguns dados que podem servir de indicadores para a aceitação de uma plataforma que os auxiliem com a gestão do seu negócio.

De acordo com a amostragem observada na pesquisa, cerca de 66,7% dos consultores de vendas de produtos estão dispos a utilizarem um serviço para auxiliá-los com o gerenciamento do seu negócio.

2 MÉTODOS E FERRAMENTAS

2.1 Ferramentas Utilizadas

Esta seção demonstra as ferramentas e tecnologias utilizadas para o desenvolvimento do projeto.

2.2.1 Softwares

Foram instaladas e utilizadas no processo de desenvolvimento deste trabalho, várias ferramentas (softwares) e sistemas. Realizei um agrupamento dos softwares utilizados com sua função de utilização no desenvolvimento deste trabalho.

1. Astah Community:
    a. De acordo com os mantenedores (www.astah.net/editions/community, 2018) o software é uma ferramenta fácil e prática para desenvolvimento dos modelos UML e para o refinando seu processo de desenvolvimento.

2. Dia Diagram Editor:
    a. De acordo os desenvolvedores (https://wiki.gnome.org/action/show/Apps/Dia, 2013) o software Dia é inspirado no programa comercial "MS Visio", embora mais voltada para diagramas informais para uso casual. Ele pode ser usado para desenhar diversos tipos diferentes de diagramas. Atualmente tem objetos especiais para ajudar a desenhar diagramas entidade relacionamento, diagramas UML, fluxogramas, diagramas de rede, e muitos outros diagramas.

Os softwares Astah Community e o Dia Diagram Editor são softwares que auxiliam no desenvolvimento de diagramas para facilitar a comunicação com os stakeholders.

3. Git:

    a. De acordo com o site oficial (https://git-scm.com/, 2018), Git é um sistema de controle de versão distribuído gratuito e de código aberto projetado para lidar com qualquer tipo de projetos, de projetos pequenos a muito grandes, com velocidade e eficiência.

4. GitHub:
    a. De acordo com site oficial (https://github.com/, 2018), GitHub é uma plataforma de desenvolvimento inspirada na maneira como você trabalha. De código-fonte aberto a negócios, você pode hospedar e revisar código, gerenciar projetos e construir software ao lado de milhões de outros desenvolvedores.

As ferramentas Git e GitHub foram utilizadas no desenvolvimento do projeto para auxiliar no controle de versionamento do código fonte e sua rastreabilidade com relação as sprints desenvolvidas.

5. PGAdmin:
    a. O pgAdmin é a plataforma de administração e desenvolvimento de código aberto mais popular e rica em recursos para o PostgreSQL, o banco de dados de código aberto mais avançado do mundo (https://www.pgadmin.org/, 2018).

6. PostgreSQL:
    a. O PostgreSQL é um poderoso sistema de banco de dados objeto-relacional de código aberto com mais de 30 anos de desenvolvimento ativo que lhe garantiu uma forte reputação de confiabilidade, robustez de recursos e desempenho (https://www.postgresql.org/, 2018).

O banco de dados PostgreSQL utilizado para implementar o domínio da aplicação e o PGAdmin uma ferramenta de administração do banco de dados foram utilizadas para auxiliarem o desenvolvimento e implementação do modelo relacional dos dados.

7. JDK 8:
    a. JDK - Java Developer Kit é um ambiente de desenvolvimento para construção de aplicações (http://www.oracle.com/technetwork/pt/java/javase/downloads/jdk8-downloads-2133151.html, 2018).

8. Eclipse Oxygen:
    a. O Eclipse Oxygen é um ambiente integrado de desenvolvimento para a plataforma Java EE. De acordo com os mantenedores (https://www.eclipse.org/, 2018).

9. Wildfly:
    a. WildFly é um servidor de aplicações, flexível, leve e que ajuda a criar aplicações incríveis (http://wildfly.org/about/, 2018).

10. Android SDK
    a. É o kit de desenvolvimento para a plataforma android (https://developer.android.com/, 2018).

11. Android Studio:
    a. O Android Studio oferece as ferramentas mais rápidas para criar aplicativos em todos os tipos de dispositivos Android (https://developer.android.com/studio/, 2018).

Os softwares enumerados do 7 ao 11 são softwares utilizados para auxiliar na programação do sistema, o Android Studio e o Eclipse são IDEs que foram

utilizadas juntamentes com a JDK e o Android SDK Kits de desenvolvimentos para a linguagem Java. O wildfly é o servidor de aplicação que implementa as especificações JEE que foi utilizado como web container e o EJB container.

12. WBS Chart Pro
    a. O WBS é um software de planejamento e controle de projetos. Ele combina uma estrutura de trabalho com gráficos, (sem os gráficos de rede do antigo WBS Schedule Pro) à folha de tarefas, e outras várias funcionalidades (http://www.criticaltools.com.br/, 2018).

13. Trello:
    a. Infinitamente flexível. Incrivelmente fácil de usar. Ótimos aplicativos móveis. É grátis. O Trello acompanha tudo, desde a foto grande até os detalhes minuciosos (https://trello.com, 2018).

Os softwares WBS Chart Pro e o Trello são softwares para auxiliarem no gerenciamento e formalização dos pacotes de trabalhos e atividades a serem desenvolvidas pelo programador.

14. Postman:
    a. O Postman é o único ambiente de desenvolvimento de API completo, para desenvolvedores de API, usado por mais de 5 milhões de desenvolvedores e 100.000 empresas em todo o mundo (https://www.getpostman.com/, 2018).

O Postman foi utilizado com um cliente HTTP para testes no desenvolvimento do API Web desenvolvida no projeto.

Todos os softwares são disponibilizados em versões gratuitas, e podem ser instalados pelo terminal ou pelo gerenciador de pacotes synaptic no Linux Ubuntu.

2.2.2 Configurações

Algumas configurações mínimas são necessárias, por exemplo: Deve-se alterar o usuário e senha do administrador do SGBD PostgreSQL para garantir a segurança dos dados persistidos.

A IDE Eclipse, pode ser customizada para atender ao desenvolvedor de forma mais fácil, por exemplo: Instalando *plug-ins* que facilitam a utilização de alguns *frameworks*. No nosso caso, foi realizado o download do Eclipse EE para desenvolvedores java para aplicações corporativas, este eclipse tem todas as

funcionalidades para se trabalhar com servidores de aplicações como o Wildfly. Foi instalado o plugin JBoss tools pelo marketplace do eclipse para facilitar a integração com o servidor.

Foi configurado o servidor para ser executado com comandos integrados na IDE para facilitar o desenvolvimento do sistema. Essa configuração é feita acessando a aba servers e clicando em new servers e apontado o local de instalação do servidor.

Para o acesso ao banco de dados foi configurado o arquivo persistence.xml para utilização do JPA. Todas as configurações de acesso ao banco de dados relacionais foram realizadas neste arquivo, como por exemplo, a string de conexão com o banco de dados.

Os datasources foram configurados no modo standalone do Wildfly, os datasources foram incluídos no arquivo standalone.xml. Neste arquivo também foi configurado o drive de acesso ao banco de dados.

2.2.3 Arcabouços e Especificações Utilizadas

Todos os frameworks utilizados no processo de desenvolvimento da aplicação são disponibilizados gratuitamente sob a licença de software livre.

O frameworks e especificações utilizados neste trabalho foram os seguintes:

1. Java Server Faces - JSF
    a. JavaServer Faces (JSF) é uma especificação desenvolvida na Java Community Process - JCP para o desenvolvimento de interfaces gráficas baseadas em componentes para aplicações web. Possui uma árvore de componentes guiados a eventos abstraindo o desenvolvimento de interfaces ricas para as aplicações.
2. Primefaces

   a. De acordo com os mantenedores (https://www.primefaces.org/#primefaces, 2018). PrimeFaces é um framework de software livre popular para JavaServer Faces com mais de 100 componentes, kit mobile otimizado, validação no lado do cliente, mecanismo de tema e muito mais.

3. Java Persistence API - JPA
   a. Java Persistence API (JPA) é uma especificação desenvolvida na Java Community Process – JCP para mapeamento objeto-relacional. Diversos frameworks implementam está especificação, um exemplo é o EclipseLink a implementação de referência da especificação.

4. Hibernate
   a. O Hibernate ORM permite aos desenvolvedores escrever mais facilmente aplicativos cujos dados superam o processo de aplicação. Como um framework para o mapeamento objeto-relacional, o Hibernate está preocupado com a persistência de dados, uma vez que se aplica a bancos de dados relacionais (via JDBC) (http://hibernate.org/orm/, 2018).

5. Enterprise Java Beans - EJB
   a. Enterprise Java Beans (EJB) é uma especificação desenvolvida na Java Community Process – JCP para desenvolvimento rápido e simplificado de componentes distribuídos, transacionais, seguros e portáveis.

6. JAX-RS
   a. Java API for RESTful Web Services - JAX-RS é uma especificação desenvolvida na Java Community Process - JCP. De acordo com os mantenedores, JAX-RS 2.0 traz diversas características úteis que tornam ainda mais simples o desenvolvimento e permitem a criação de aplicações com a arquitetura RESTful para Java SE/EE ainda mais sofisticadas mas, ao mesmo tempo, leves.

7. CDI

a. Context Dependency Injection - CDI é uma especificação desenvolvida na Java Community Process - JCP para injeção de dependências na plataforma java.

8. Bean Validation

    a. Bean Validation é uma especificação desenvolvida na Java Community Process - JCP para expressar restrições em modelos de objetos por anotações, permite escrever restrições (validações) personalizadas de maneira extensível.

9. JAAS
    a. O Java Authentication and Authorization Service - JAAS, é a implementação Java do framework de segurança de informações Pluggable Authentication Module - PAM. O JAAS foi introduzido como uma biblioteca de extensão para a Plataforma Java, Standard Edition 1.3 e foi integrado na versão 1.4 (https://docs.oracle.com/javase/8/docs/technotes/guides/security/jaas/JAASRefGuide.html, 2018).

2.2.4 Ambiente de Desenvolvimento

A plataforma *Java Enterprise Edition* foi escolhida para realizar a implementação dos artefatos funcionais do software por possuir um grande ecossistema de ferramentas que auxiliam o programador.

A implementação do código fonte da aplicação foi realizada com auxílio da IDE Java EE Eclipse versão Oxygen como mostras as FIG. 13.

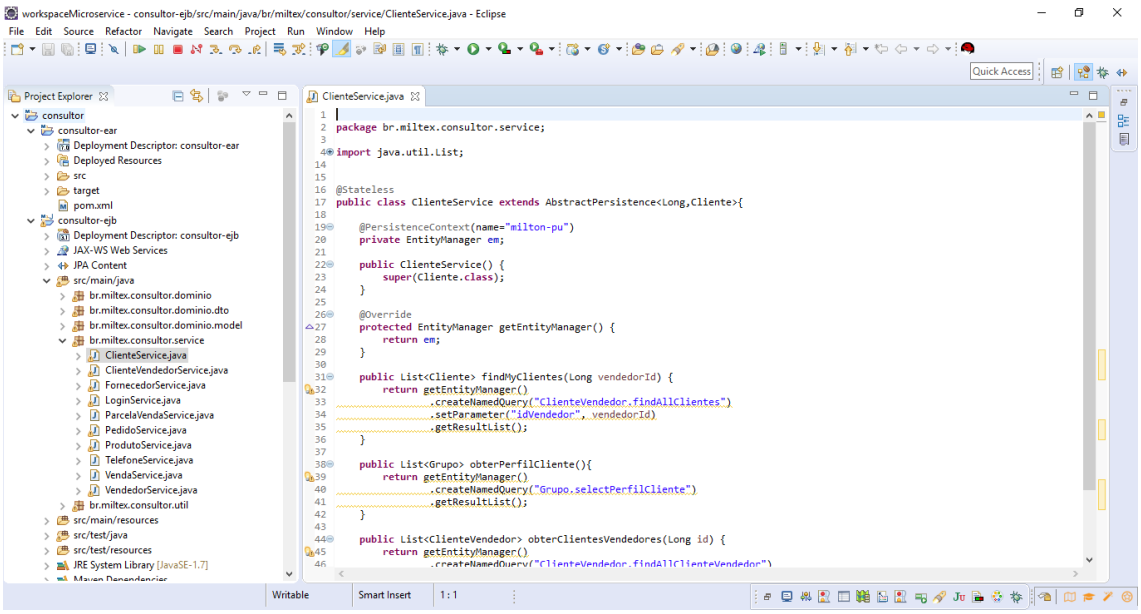


Figura 13 - IDE Eclipse

O Apache Maven que de acordo com seus mantenedores (https://maven.apache.org/, 2018) é uma ferramenta de gerenciamento e compreensão de projetos de software. Com base no conceito de um Project Object Model - POM, o Maven pode gerenciar a criação, o relatório e a documentação de um projeto a partir de uma informação central. Foi integrado ao Eclipse para auxílio ao desenvolvimento do projeto.
O servidor de aplicação Wildfly foi integrado ao Eclipse para o desenvolvimento do projeto. O servidor fornece implementações das especificações Java EE citadas anteriormente. De acordo com os mantenedores (http://wildfly.org/about/, 2018) o WildFly implementa os mais recentes padrões corporativos de Java. O Java EE melhora a produtividade do desenvolvedor, fornecendo recursos empresariais sofisticados em frameworks fáceis de utilizar que e reduzem o muito a necessidade de um especialista técnico. Isso permite que sua equipe se concentre nas principais necessidades comerciais do seu aplicativo.

Para controle e versionamento do código fonte foi utilizado o controle de versão Git que de acordo com os desenvolvedores (https://git-scm.com/, 2018) é um sistema de controle de versão distribuído gratuito e de código aberto projetado para lidar com tudo, de projetos pequenos a muito grandes, com velocidade e eficiência. O Git é fácil de aprender e com desempenho extremamente rápido. Ele supera as ferramentas de SCM, como Subversion, CVS, Perforce e ClearCase, com recursos como ramificação local barata, áreas de preparação convenientes e vários fluxos de trabalho.

Foi também utilizado um repositório remoto criado no GitHub para hospedar o código fonte do projeto. O GitHub é uma plataforma de desenvolvimento inspirada na maneira como você trabalha. De código-fonte aberto a negócios, você pode hospedar e revisar código, gerenciar projetos e construir software ao lado de milhões de outros desenvolvedores (https://github.com/, 2018).

O Sistema de Gerenciamento de Banco de Dados - SGBD utilizado foi o PostgreSQL, juntamente com a ferramenta administrativa PgAdmin como mostrado na FIG. 14.

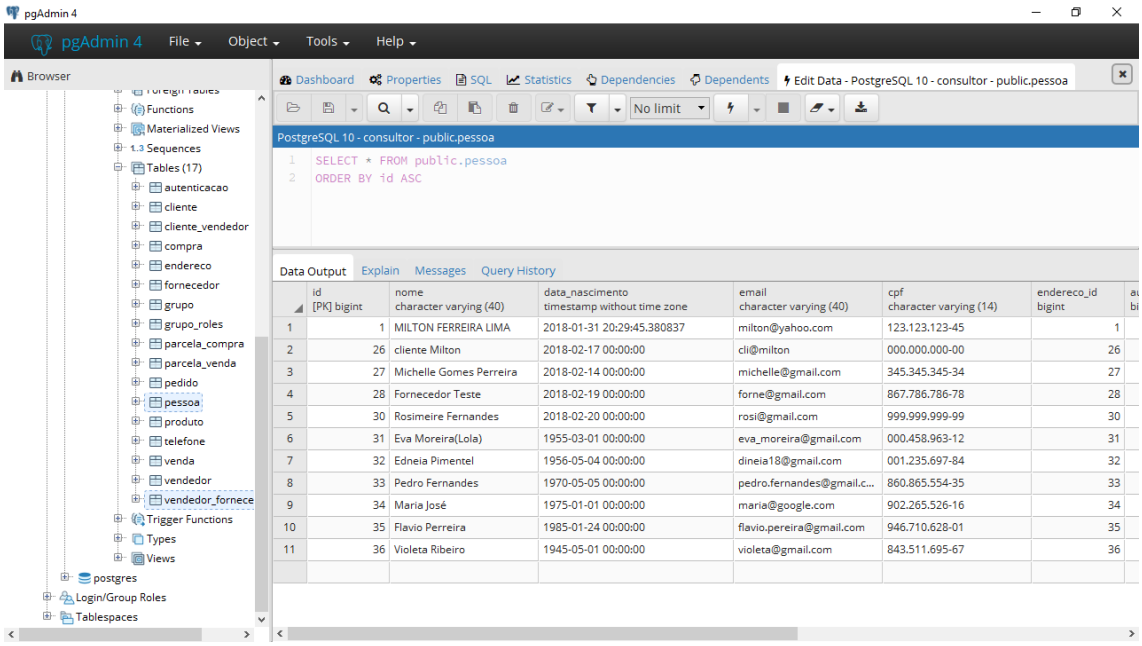


Figura 14 - PgAdmin

A Java Developer Kit - JDK 8 foi utilizada para o desenvolvimento do projeto.

O Android Studio, como mostrado na FIG. 15, juntamente com a SDK Android foram utilizados para o desenvolvimento do projeto mobile.

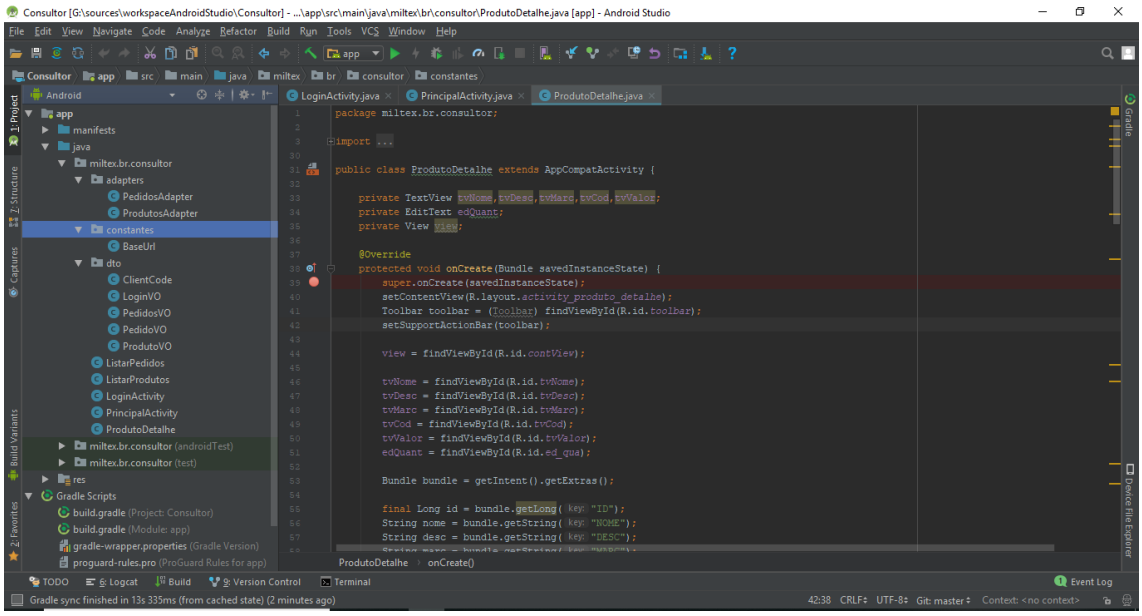


Figura 15 - Android Studio

O Gradle foi utilizado para auxiliar no desenvolvimento do projeto mobile juntamente com o Android Studio.

De aplicativos móveis a microsserviços, de pequenas startups a grandes empresas, a Gradle ajuda as equipes a criar, automatizar e entregar softwares melhores, mais rapidamente (https://gradle.org/, 2018).

3. DEFINIÇÃO DE REQUISITOS

3.1 Levantamento e Estruturação dos Requisitos

Os requisitos para o desenvolvimento do sistema foram levantados com a realização de conversas com vendedores de produtos, tais como; Rommanel, Natura, Jequiti, Avon etc.

Com as primeiras conversas surgiram as necessidades básicas, de persistir informações referentes aos produtos e clientes. Uma indagação levantada em conversa foi; “qual as informações dos produtos e clientes são importantes ?”.

Essas informações foram sendo categorizadas em: Pouco, Médio e Alto valor para o negócio. Com isso foi surgindo alguns diagramas de casos de usos. Esses diagramas foram guiando a documentação das estórias de usuários, que foram escritas posteriormente.

Os diagramas de casos de usos foram sendo criados à medida que eram entregues algumas releases para testes de aceitação com os stakeholders. Esses diagramas podem ser visualizados nas FIG. 16 e FIG. 17.

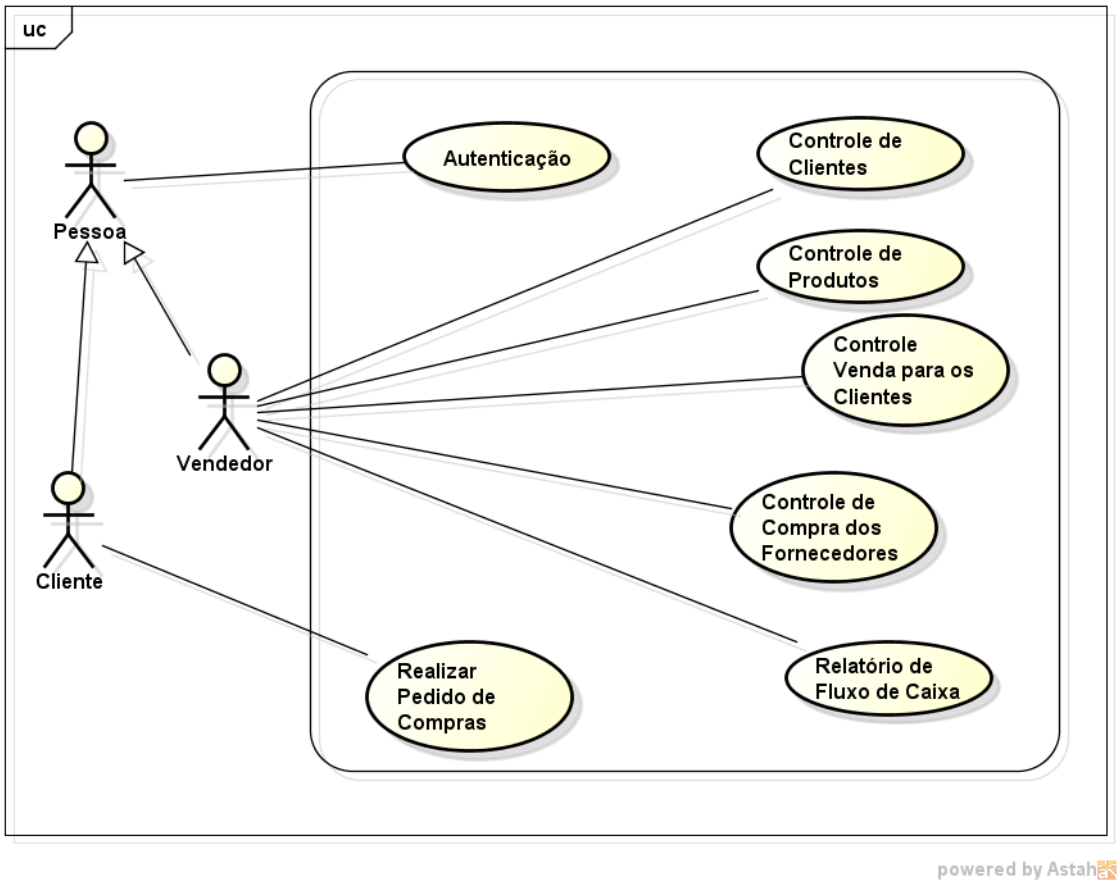


Figura 16 - Diagrama de Casos de Usos

Os Casos de Uso foram sendo desenvolvidos para geração das estórias de usuários. Algumas estórias podem ser vistas a seguir.

- Como vendedor de produtos eu posso visualizar todos meus produtos que estão cadastrados no sistema, o vendedor também pode adicionar novos produtos, alterar os valores, tais como; preço, nome, código, etc.

- O cliente pode realizar o login no sistema e visualizar todos os produtos dos seus vendedores. Pode também realizar o pedido de novos produtos, adicionando a quantidade de produtos que ele quer pedir. Quando ele informa a quantidade de item que ele deseja comprar o sistema te informa o valor total e unitário dos itens pedidos.

- Os usuários devem realizar login no sistema para acessar informações do negócio. O usuário Cliente terá acesso aos produtos dos seus vendedores, visualizando informações, tais como; valor do produto, código do produto, entre outras. Como Cliente, será possível realizar o pedido de produtos pelo sistema Web e também pelo sistema Android. O Cliente não terá acesso às funcionalidades do Vendedor.

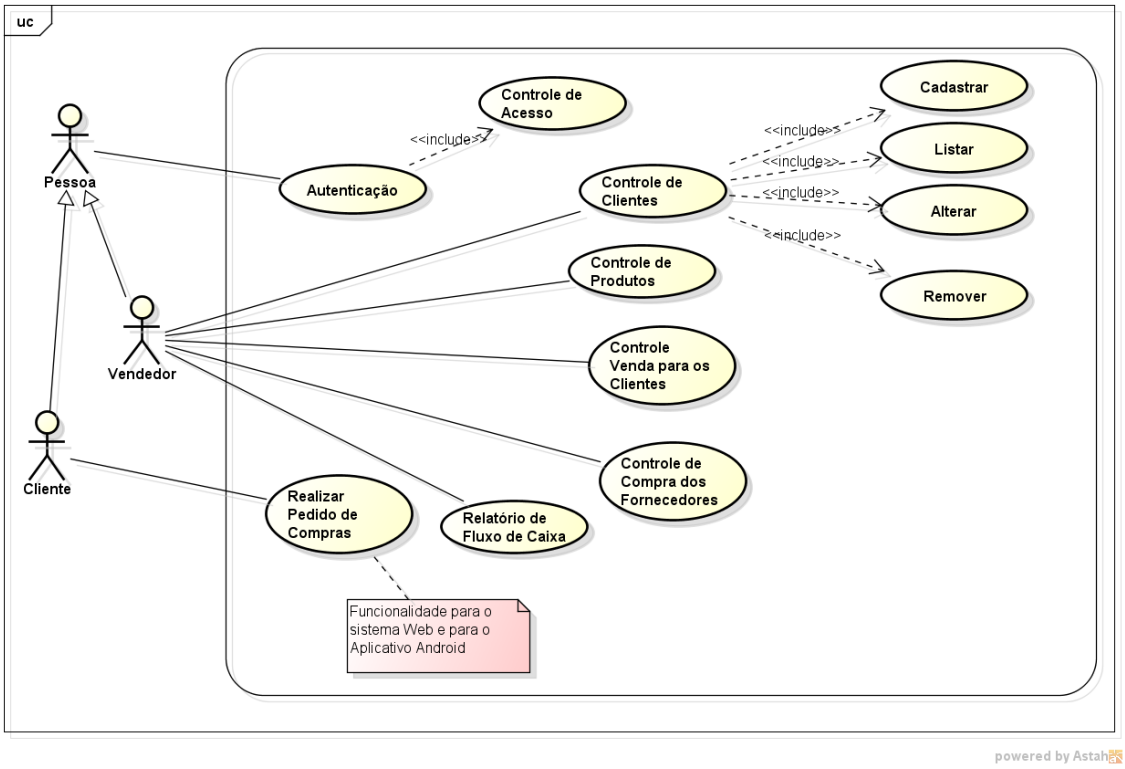


Figura 17 - Diagrama de Casos de Usos

O diagrama citado acima, tem as funções macro do sistema. Os Casos de Usos, Cadastrar, Listar, Alterar e Remover estão associados a todos os casos de usos, menos o Autenticação e Controle de Acesso.

Os requisitos foram estruturados em funcionais e não-funcionais, os requisitos funcionais deram origem a algumas dessas estórias de usuários e foram documentadas utilizando a ferramenta Trello como representada na FIG. 18 e pode ser acessado (https://trello.com/b/OvSJNX58/projeto-tcc, 2018).

Os requisitos funcionais foram sendo estruturados nos post-its como estórias de usuários a serem implementadas pelo desenvolvedor.

Utilizou-se o sistema Kanban para auxiliar no controle da produção do software com informações sobre, o quê precisa ser feito, o quê está sendo feito e o quê foi concluído.

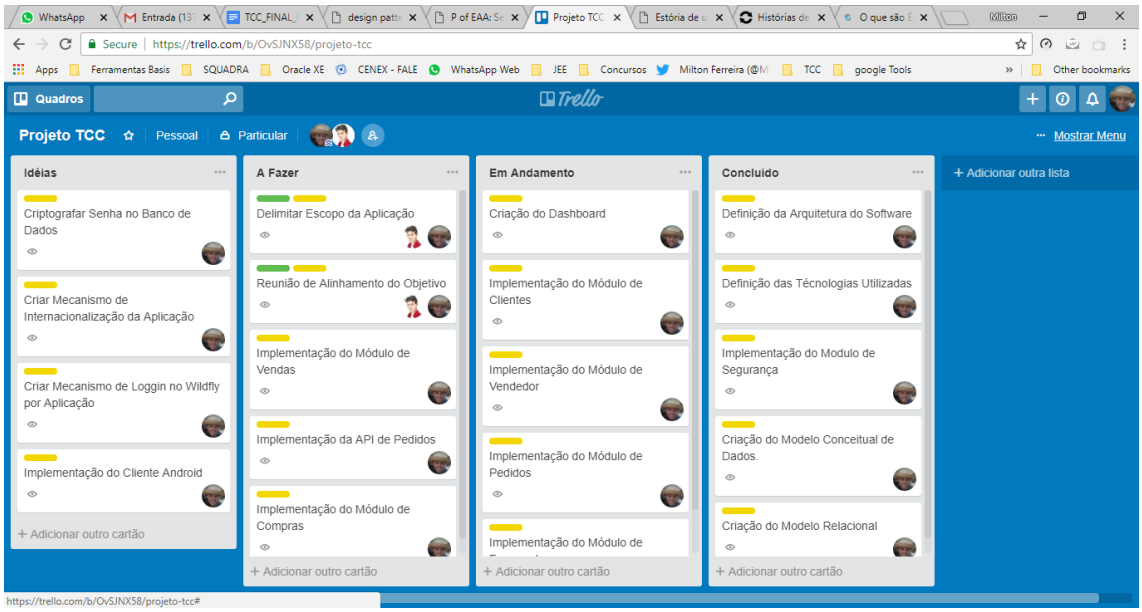


Figura 18 - Kanban Board

Estas estórias foram gerando o Product Backlog para guiar o desenvolvedor na criação do produto de software.

Em um segundo momento foi realizada priorização dos Backlogs que mais geram valor para o negócio e também fornecem as funcionalidades básicas para o funcionamento dos sistema na sua primeira versão.

Após essa fase de priorização dos backlogs, foram definidas sprints semanais para que o desenvolvedor pudesse trabalhar na implementação das estórias de usuários.

4. MODELO DE DADOS

4.1 Modelo Conceitual

Foi necessária a elaboração do modelo conceitual dos dados, levando em consideração os requisitos estabelecidos pela elicitação dos requisitos realizados com a vendedora de produtos.

Várias versões do modelo conceitual foram criadas até atingir sua versão final. Estas versões podem ser observadas entre as figuras, FIG.19 a FIG.22. A partir do modelo conceitual final, converteu-se para o modelo relacional.

Segundo Elmasri e Navathe (2005, p. 89) “O modelo relacional foi introduzido por Ted Codd, da IBM *Research*, em 1970, em um artigo clássico (Codd, 1970) que imediatamente atraiu a atenção em virtude de sua simplicidade e base

matemática. O modelo usa o conceito de uma relação matemática, algo como uma tabela de valores, como seu bloco de construção básica e tem sua base teórica na teoria dos conjuntos e na lógica de predicados de primeira ordem".

O modelo '*entidade-relacionamento*' será convertido no modelo relacional que propiciará a sua implementação num SGBD – Sistema Gerenciador de Banco de Dados, no nosso caso, *PostgreSQL*.

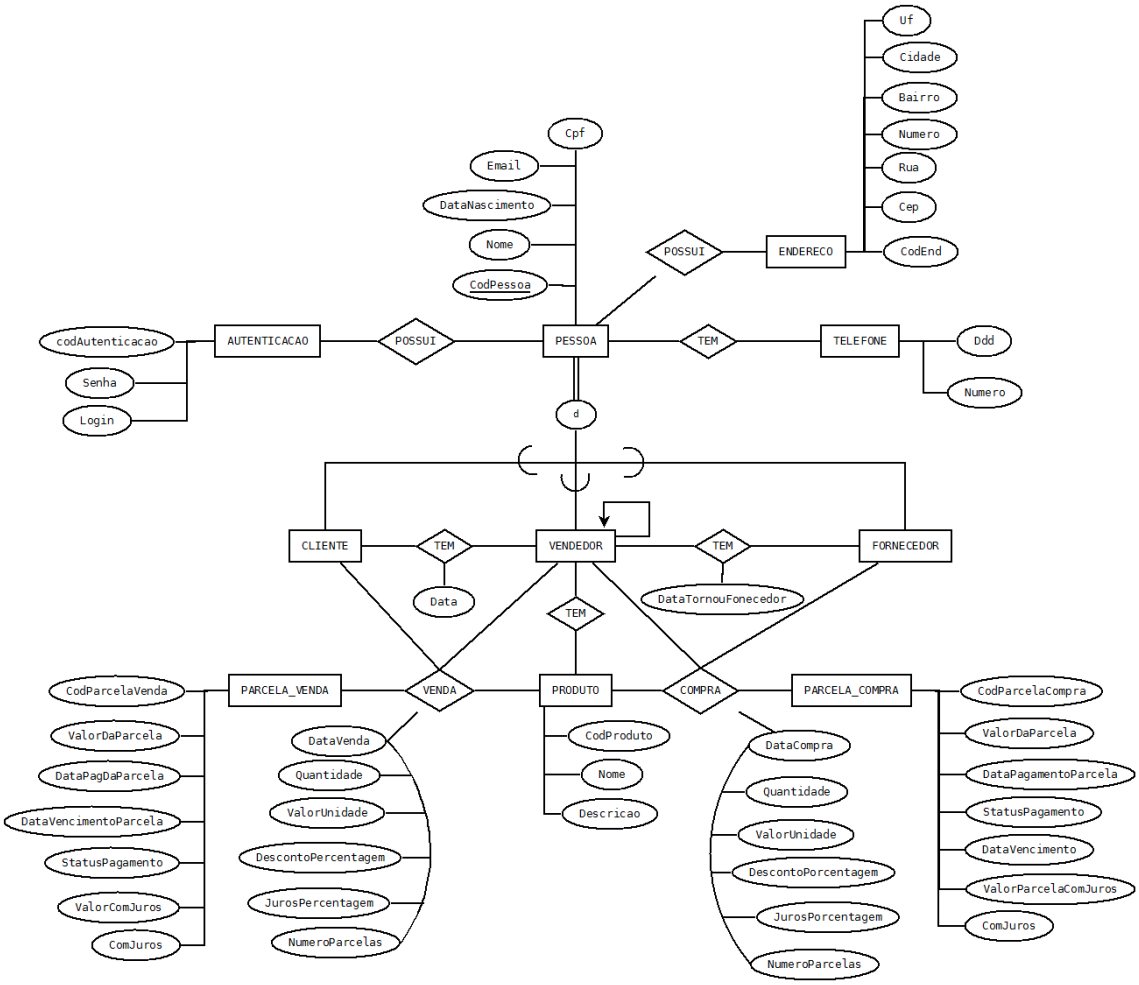


Figura 19 - Modelo Conceitual Versão 1

O modelo demonstrado acima foi o primeiro esboço do modelo conceitual dos dados. As principais entidades foram levantadas nesta fase.

Inicialmente foram incluídos alguns atributos nas entidades do modelo, somente a identificação das entidades e alguns atributos foram levados em consideração nesta fase.

No modelo de representação escolhido as entidades são representadas como retângulos e os relacionamentos como losangos, os atributos como elipses.

Fez-se necessário uma análise, para identificação dos dados que são relevantes e precisam ser armazenados no banco de dados.

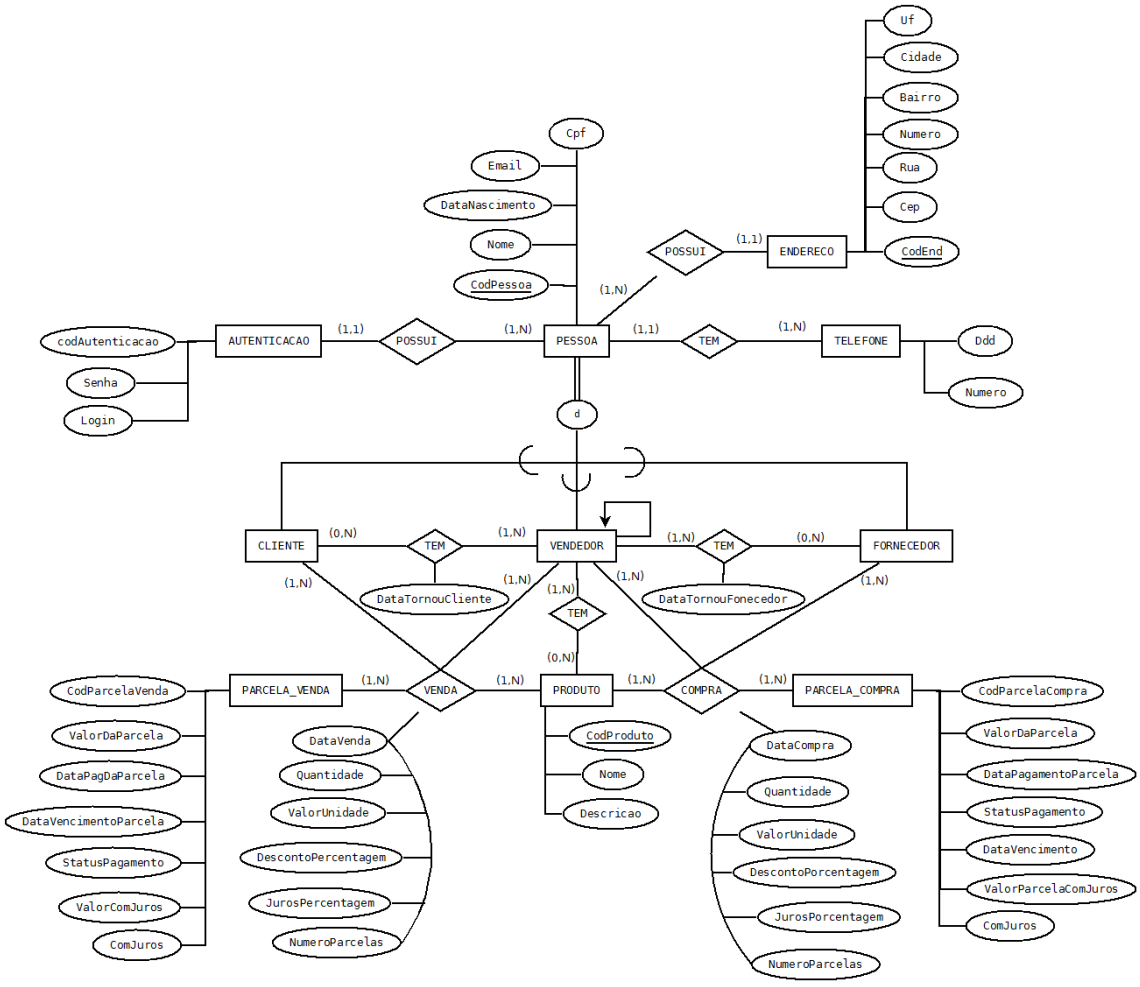


Figura 20 - Modelo Conceitual Versão 2

Nesta etapa iniciou-se a inserção das cardinalidade dos relacionamentos entre as entidades do modelo.

Foi utilizado uma derivação da relação PESSOA para as relações CLIENTE, VENDEDOR e FORNECEDOR, visto que há relacionamentos e atributos comuns as relações.

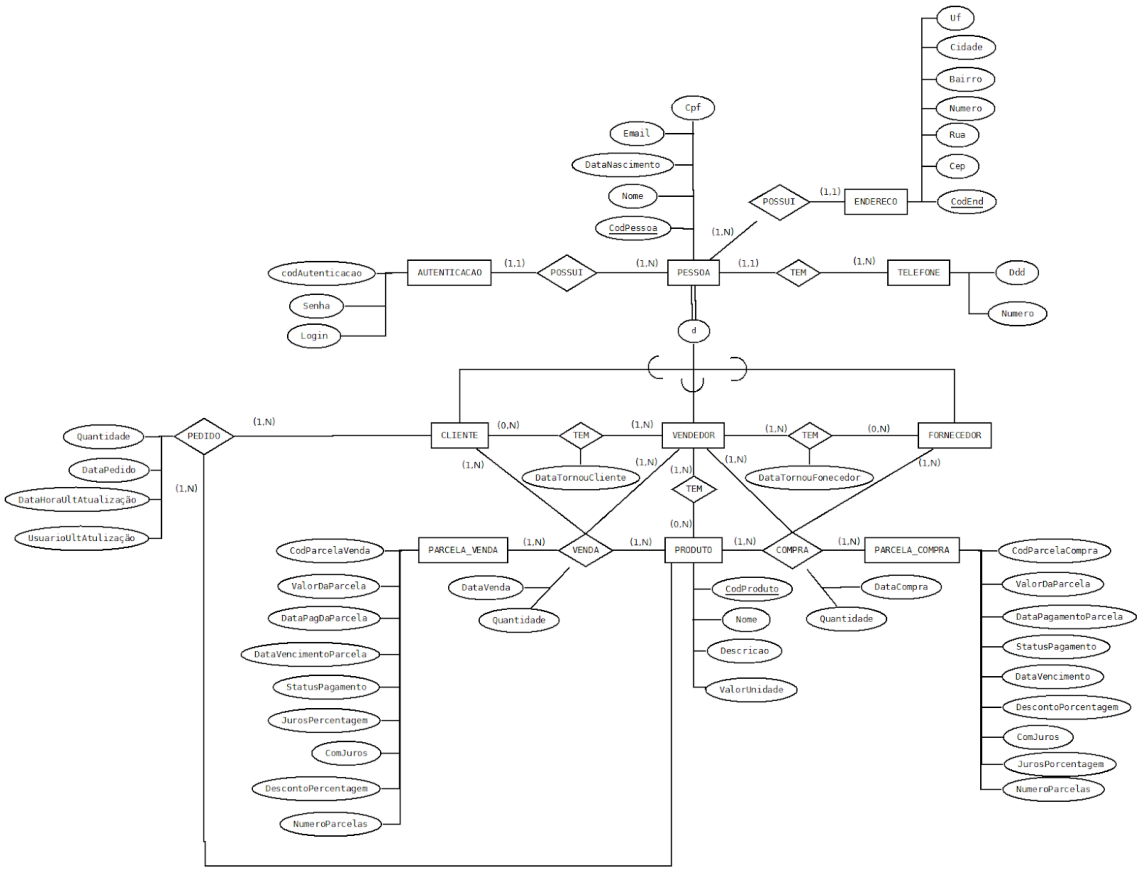


Figura 21 - Modelo Conceitual Versão 3

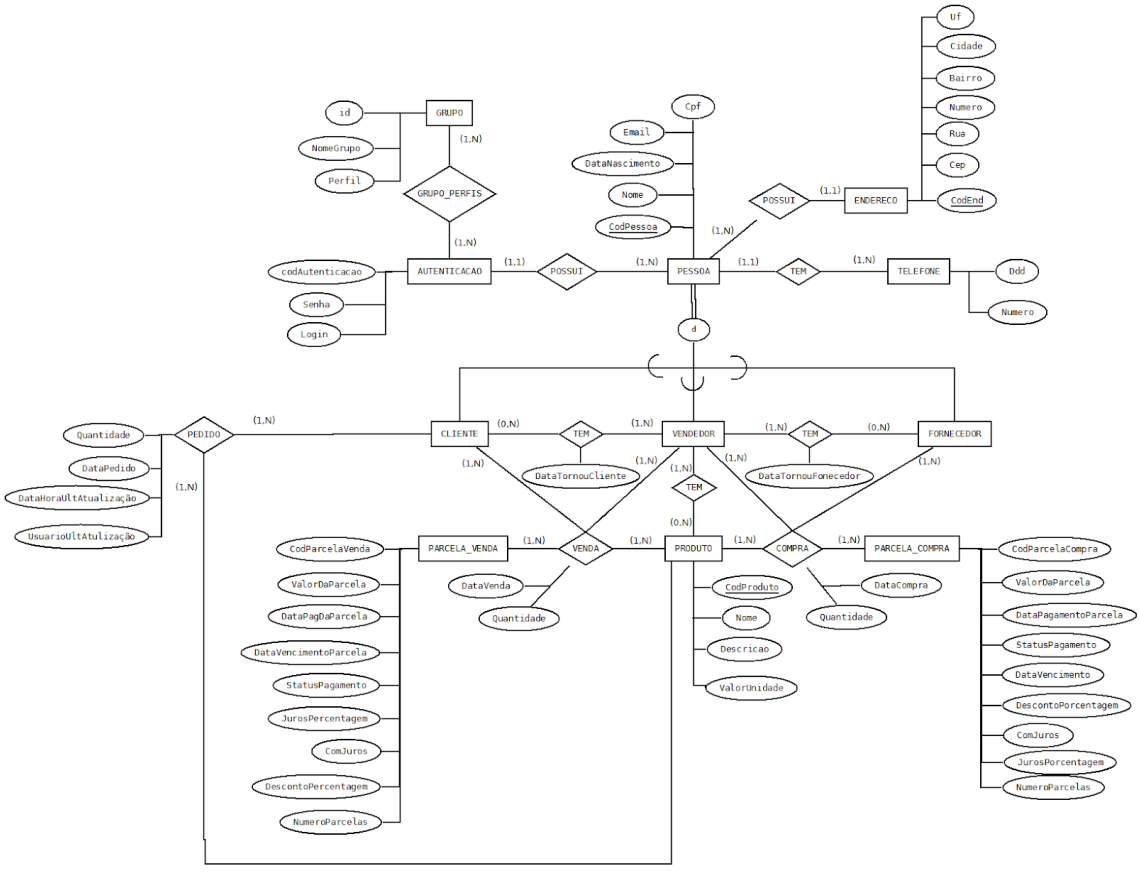


Figura 22 - Modelo Conceitual Versão 4

O modelo conceitual foi sendo desenvolvido incrementalmente a medida que os requisitos foram sendo estruturados e dando origem as relações representadas nas figuras acima.

A última intervenção no modelo foi para inserir informações dos grupos e perfis de segurança. Foi adicionado a relação AUTENTICACAO outra relação denominada GRUPO para controle de autenticação e autorização dos usuários.

As relações que compõem o modelo conceitual final, são as seguintes:

1. GRUPO
    a. Representa os grupos de usuários e perfis que poderão acessar o sistema.

2. AUTENTICACAO
    a. Representa as informações necessárias para se autenticar no sistema.

3. PESSOA
    a. Uma representação dos usuários principais do sistema.

4. ENDERECO
    a. Representa informações de endereços dos usuários.

5. TELEFONE
    a. Uma representação das informações de contato dos usuários.

6. CLIENTE
    a. Uma representação dos cliente que acessarão o sistema.

7. VENDEDOR
    a. Representação dos vendedores que terão acessos ao sistema.

8. PARCELA_VENDA
    a. Uma representação das informações de venda dos produtos.

9. PRODUTO
    a. Representação dos produtos.

10. PARCELA_COMPRA
    a. Uma representação das informações de compra dos produtos.

11. VENDA
    a. Uma representação da venda de um produto.

12. COMPRA
    a. Representação da compra de um produto.

13. PEDIDO

a. Uma representação do pedido dos produtos.

4.2 Modelo Relacional

A partir do modelo conceitual obtivemos o modelo relacional dos dados aplicando o algoritmo de conversão do mapeamento do modelo ER para o relacional. O algoritmo de mapeamento do modelo ER – Entidade Relacionamento para o modelo Relacional de Elmasri e Navathe (2005, p. 139). A partir da aplicação deste algoritmo obteve-se o modelo relacional. Com o modelo relacional em mãos foi possível implementar os scripts Data Definition Language - DDL para o banco de dados PostgreSQL.

O mapeamento do modelo entidade-relacionamento para o modelo relacional, a ser implementado, garante a propriedade de consistência e integridade dos dados a serem persistidos.

A partir do modelo relacional, surgiu o diagrama DER, que pode ser visualizado na FIG. 23, este diagrama foi gerado por ferramentas auxiliares, utilizadas após o banco de dados ser implementado no SGBD PostgreSQL.

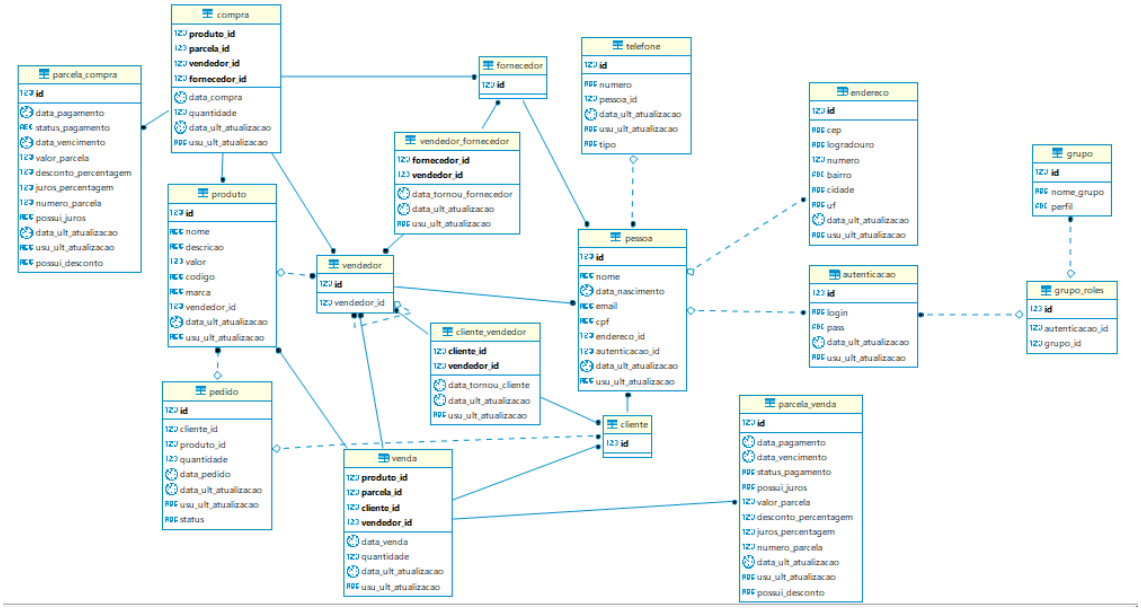


Figura 23 - Diagrama Entidade-Relacionamento - DER

5. MODELO ARQUITETURAL

5.1 Descrição da Arquitetura

O projeto foi empacotado com a utilização de um Enterprise Application aRchive - EAR, este tipo de arquivo é utilizado por aplicações JEE para empacotar os módulos em um arquivo simples para a realização de publicação em um servidor de aplicação, como mostra a FIG. 24.

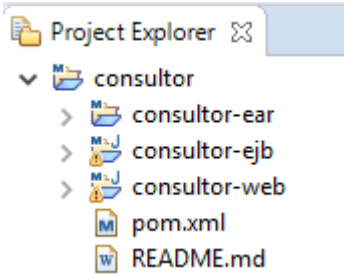


Figura 24 - Projeto no Eclipse

O sistema como mostra a FIG. 24 é constituído pelos módulos, consultor-web e consultor-ejb.

O módulo Enterprise Java Beans - EJB implementa o domínio da aplicação com as classes de entidades persistentes e os serviços que disponibilizam operações das regras de negócio da aplicação.

Todas entidades que são classes que realizam o mapeamento objeto-relacional implementam a interface AbstractEntity. A classe abstrata AbstractPersistence é uma classe genérica tipada com seus limites definidos pelo tipo Number e pelo tipo AbstractEntity, essa classe define o método abstrato chamado getEntityManager que será implementado pelas classes de serviço que estenderem esta classe e será responsável por devolver o EntityManager injetado na classe de serviço específica.

As operações CRUD - Create, Read, Update e Delete são implementadas na classe AbstractPersistence que serão herdadas por todos serviços que implementarem essa classe e que realizam operações nas entidades que implementam a interface AbstractEntity, como mostrado na FIG. 25 (diagrama de classes)

Os serviços específicos são implementados em cada serviço conforme a necessidade da regra de negócio associado ao serviço.

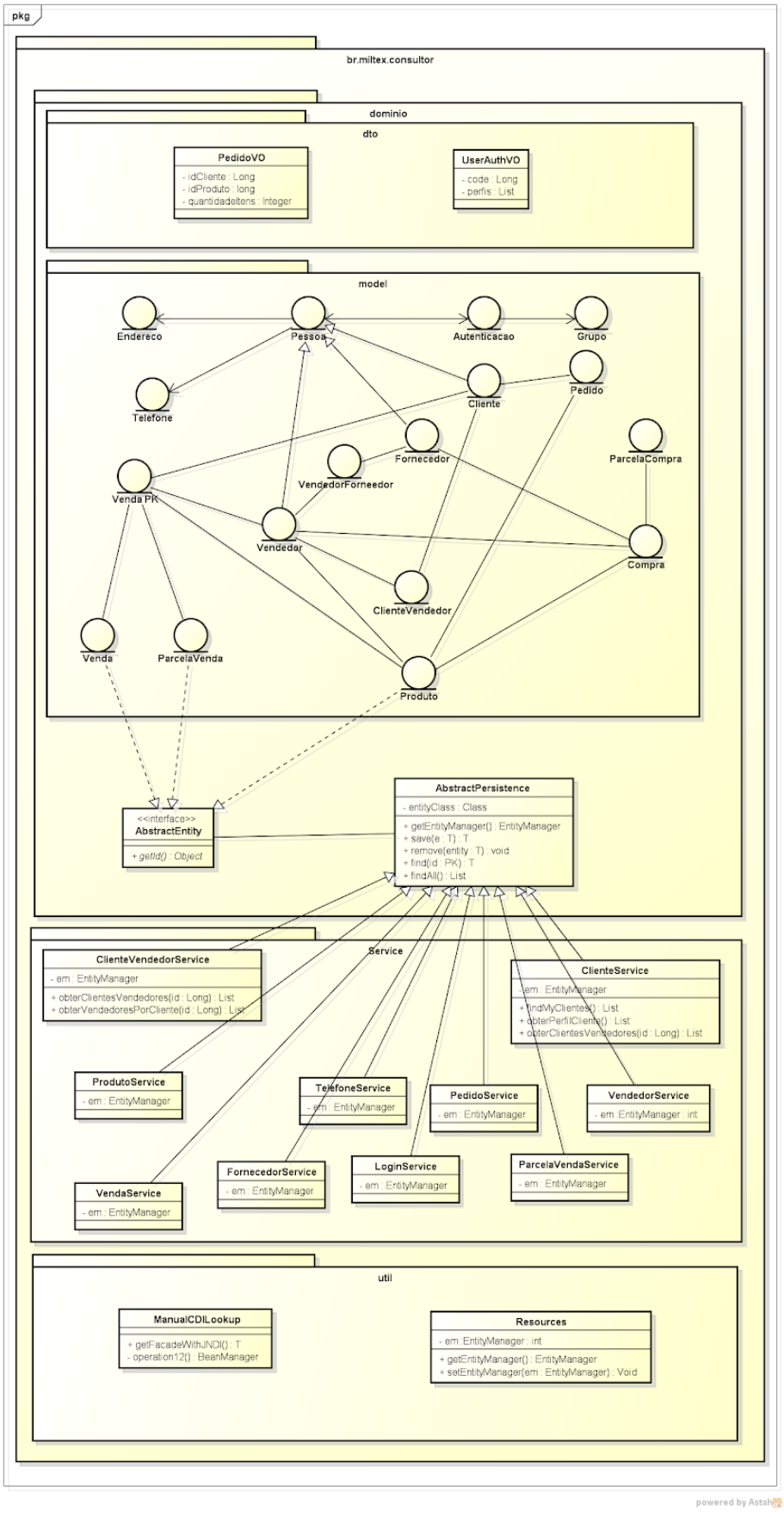


Figura 25 - Diagrama de Classes

O diagrama de classe da FIG. 25, demonstra a estrutura dos pacotes do módulo EJB. O pacote model representa as entidades mapeadas os objetos do domínio da aplicação, no pacote service estão representadas as classes que implementam os serviços da aplicação. O pacote dto estão classes auxiliares que são utilizadas para trafegar informações sobre os serviços.

O módulo web é o responsável pela aplicação web que pode ser acessada pelo navegador.

O módulo web é implementado utilizando a especificação Java Server Faces e foi projetado para possuir as camadas de controle e de visão da aplicação, o modelo ou domínio da aplicação está no módulo EJB.

A FIG. 26 representando uma abstração do diagrama de classe, como mostrada, divide as camadas da aplicação web.

O ManagedBeans utilizam os serviços do módulo EJB que são injetados pela utilização do CDI.

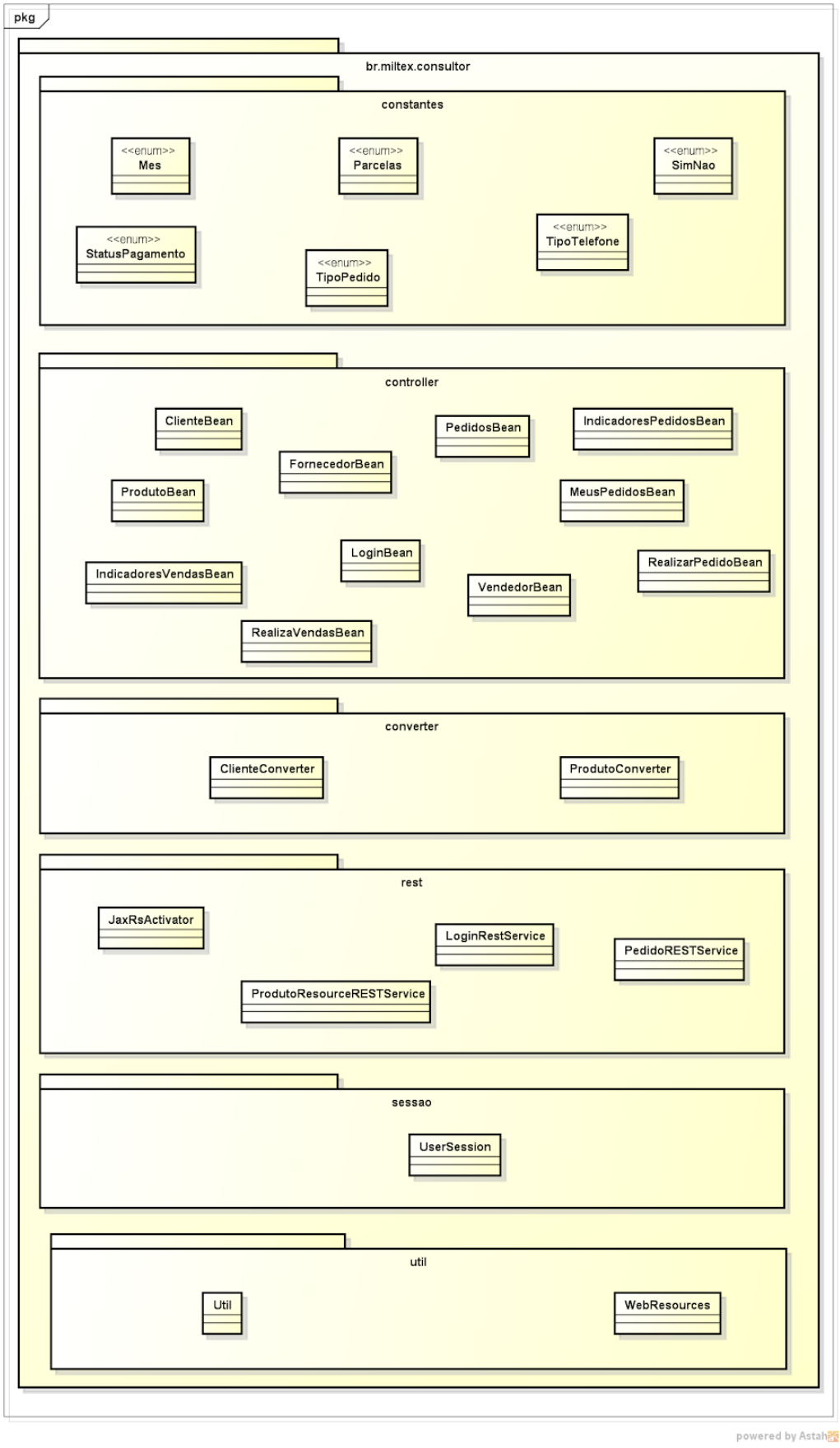


Figura 26 - Diagramas de Classes

Os Diagramas de classe da FIG. 26 representa uma abstração do módulo Web do projeto da plataforma para consultoras independentes.

Os ManagedBeans representados no pacote controller utilizam através da especificação Context Dependency Injection - CDI os serviços que são injetados para realizar as regras de negócios da aplicação, esses serviços seguem o *Service Layer Pattern*, fornecendo transações distribuídas gerenciadas pelo container EJB.

De acordo com Martin Fowler (2003): "*Os aplicativos corporativos normalmente exigem tipos diferentes de interfaces para os dados que armazenam e a lógica que implementam: carregadores de dados, interfaces com o usuário, gateways de integração e outros. Apesar de seus propósitos diferentes, essas interfaces geralmente precisam de interações comuns com o aplicativo para acessar e manipular seus dados e invocar sua lógica de negócios. As interações podem ser complexas, envolvendo transações através de múltiplos recursos e a coordenação de várias respostas a uma ação. Codificar a lógica das interações separadamente em cada interface causa muita duplicação.*
Uma camada de serviço define o limite de um aplicativo e seu conjunto de operações disponíveis na perspectiva da interface das camadas do cliente. Ele encapsula a lógica de negócios do aplicativo, controlando transações e coordenando respostas na implementação de suas operações."

Nossa camada de serviços foi implementada no módulo EJB do projeto sendo utilizada pelo módulo WEB na maioria dos casos no pacote de controle, onde estão os ManagedBeans que gerenciam a árvore dos componentes do JSF.

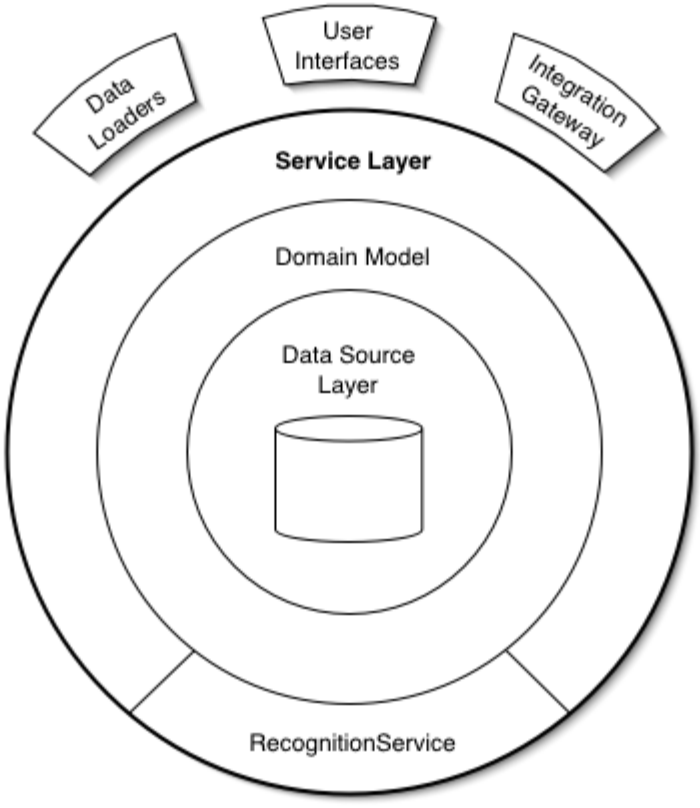


Figura 27 - Service Layer
(https://www.martinfowler.com/eaaCatalog/serviceLayer.html, 2018)

O mapeamento objeto relacional utilizando a especificação Java Persistence API - JPA foi utilizado com a implementação do Hibernate como Provider para o mecanismo de acesso a dados através do design pattern repository.

Repository faz a mediação entre o domínio e as camadas de mapeamento de dados usando uma coleção como interface para acessar objetos de domínio (Martin Flowler, 2003).

Um exemplo de aplicação do design pattern repository pode ser verificado na FIG. 28.

Segundo Martin Fowler (2003), Conceitualmente, um Repositório encapsula o conjunto de objetos persistidos em um datasource e as operações executadas sobre eles, fornecendo uma visão mais orientada a objeto da camada de persistência. O Repository também suporta o objetivo de obter uma separação limpa e uma dependência unidirecional entre o domínio e as camadas de mapeamento de dados.

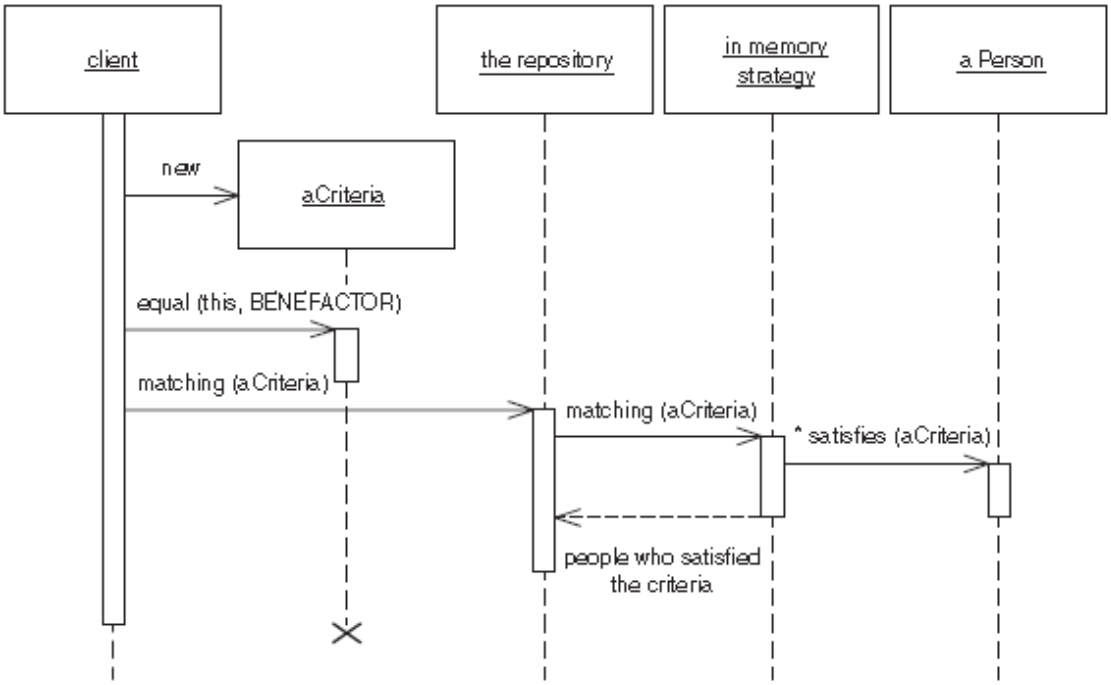


Figura 28 - Exemplo de uso do padrão repositório (Patterns of Enterprise Application Architecture, p. 322)

Uma forma de utilização de computação distribuída é a utilização do estilo arquitetural Representa Representational State Transfer - REST. Este estilo é embasado sobre o protocolo Hypertext Transfer Protocol - HTTP e o tráfego de textos sobre a Internet. A API Web RESTful consumida pela aplicação android foi construída utilizando este estilo arquitetural.

De acordo com (Bill Burke, 2013), no mundo REST, a capacidade de endereçamento é gerenciada por meio do uso de URIs. Quando você faz uma solicitação de informações em seu navegador, você está digitando em um URI. Cada solicitação HTTP deve conter o URI do objeto para o qual você está solicitando informações ou enviando informações. O formato de um URI é padronizado da seguinte maneira:

scheme://host:port/path?queryString#fragment

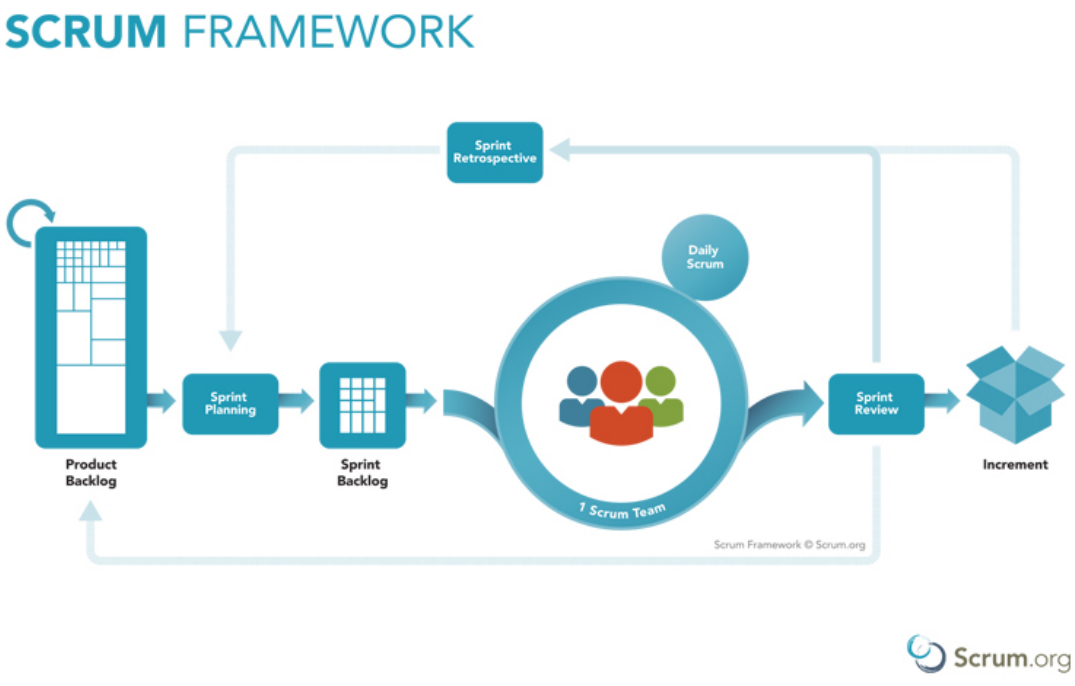
O protocolo HTTP define um grupo de operações que são comumente utilizadas para implementar as requisições ao servidor. As mais utilizadas na implementação de serviços RESTful são as seguintes: GET, PUT, DELETE, POST, comumente estas operações são chamadas de verbos HTTP.

A Web API RESTful desenvolvida neste trabalho, utiliza JSON para o tráfego dos dados entre as aplicações cliente-servidor. A implementação RESTeasy do servidor de aplicação JBoss foi utilizada para a construção da Application Programming Interface - API.

6. IMPLEMENTAÇÃO

A abordagem para o desenvolvimento do sistema e do aplicativo android proposto neste trabalho foi com auxílio do scrum e do kanban.

De acordo com (Jeff Sutherland , 2013), Scrum é um framework para desenvolver e manter produtos complexos. A definição consiste em papéis, eventos, artefatos que o compõem.



Figrua 29 - Scrum Framework (https://www.scrum.org/)

Como pode ser observado na FIG. 29 (https://www.scrum.org/, 2018) foi definido o Product Backlog a partir dos requisitos e estórias de usuários levantada com os stakeholders. No Sprint Planning foi realizada a estruturação/priorização dos Product Backlog dando origem ao conteúdo a ser desenvolvido na realização da Sprint. O Sprint Backlog foi organizado utilizando os post-its na ferramenta Trello que funcionou basicamente como um kanban board das estórias de usuários e atividades para a realização dos ciclos das sprints. Na fase da Sprint Review eram revistos quais atividades eram concluídas, quais não eram concluídas, qual o impacto na publicação de uma nova release e era reestruturado o novo Sprint Backlog para a realização da nova Sprint.

No kanban board utilizando o Trello, foi estruturado as atividades e as estórias de usuários a serem implementadas, nas raias; idéias, a fazer, em andamento e concluído.

6.1 Módulo Web

O módulo web foi construído utilizando-se a especificação Java Server Faces - JSF. Toda interação com a camada de visão foi utilizando a árvore de componentes do JSF e os componentes do Primefaces para melhorar a usabilidade e experiência do usuário com a aplicação. Todo acesso via Web é realizado por esse módulo, as páginas criadas com os componentes do JSF são renderizadas em HTML no browser do usuário.

A autenticação e autorização é controlada pela utilização do Java Authentication and Authorization Service - JAAS como pode ser observado na FIG. 30 a tela de acesso ao sistema.

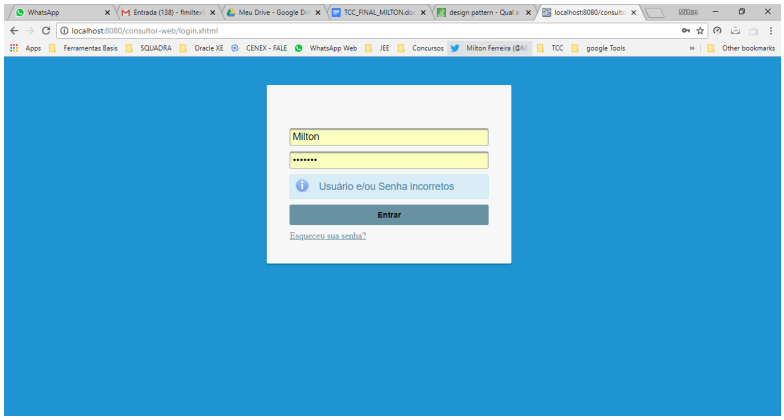


Figura 30 - Logon do Sistema

Após a realização do login o usuário é direcionado para o dashboard com informações sobre os pedidos realizados e vendas realizadas dos produtos. Como pode ser observado na FIG. 31.

Figura 31 - Tela Inicial do Sistema

O controle de autorização e acesso é realizado pelo JAAS com base nos perfis cadastrados para acesso ao sistema. Foram definidos três perfis para acessar o sistema são eles: Administrador, vendedor, fornecedor e cliente.

Quando o usuário autentica-se com o perfil de cliente e tenta acessar recurso que estão disponíveis para o perfil de vendedor o sistema bloqueia o acesso para o usuário.

Todas as funcionalidades de acesso ao sistema através da web estão disponíveis neste módulo. Um exemplo é o cadastro de produtos que pode ser visualizado na FIG. 32.

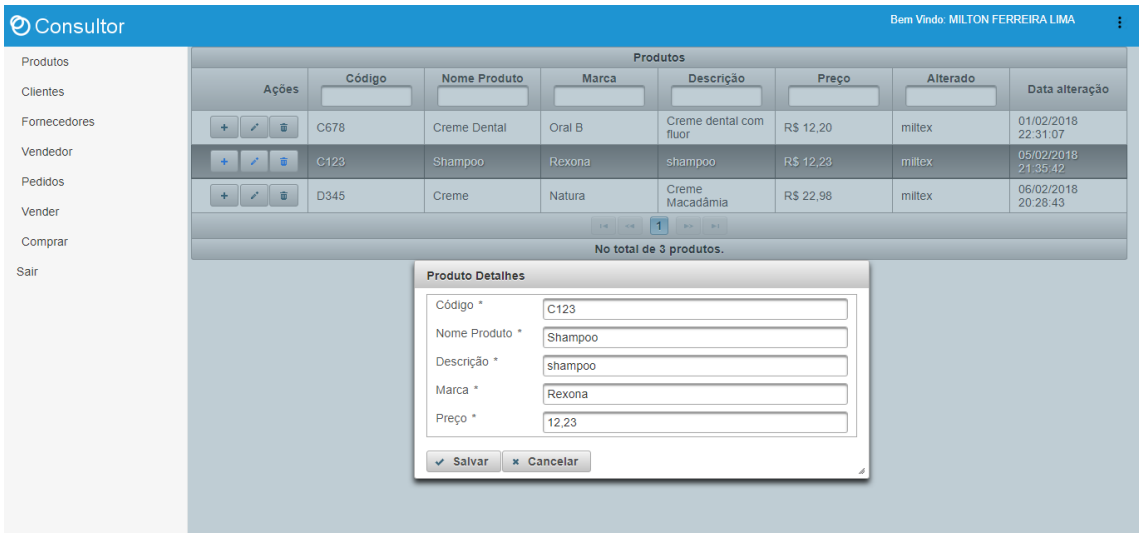


Figura 32 - Tela de Cadastro de Produtos

6.2 Módulo EJB

Este módulo foi criado seguindo a especificação Enterprise Java Beans - EJB. O contêiner EJB utilizado foi o JBoss Wildfly. Este módulo ao contrário do módulo web não possui uma interface gráfica associado a ele.

Este módulo foi responsável pela implementação da lógica de negócio da aplicação e pela camada de abstração de acesso aos dados.

Os serviços que implementam as regras de negócios da aplicação ficam no pacote service do módulo EJB.

Os serviços de integração com o aplicativo móvel que utilizam a especificação JSR 339: JAX-RS 2.0: The Java API for RESTful Web Services foi a escolhida para implementarmos esta integração com a plataforma de consultores independentes.

A API desenvolvida nesse trabalho utilizará os protocolo *HTTP* e o conceito de *Representational State Transfer* REST. Esse conhecimento é necessário para pensarmos em uma arquitetura baseada em práticas de utilização de serviços RESTful. O protocolo HTTP é amplamente utilizado pela sua simplicidade e para o tráfego de dados sob a Internet.

Utilizaremos o *JavaScript Object Notation* – JSON para o tráfego de dados dos serviços para os seus clientes. Utilizaremos como cliente dos serviços RESTful desenvolvidos, o *SDK Android* para consumir os serviços implementados utilizando a plataforma Java e o SGBD PostgreSQL. Com relação a implementação dos serviços será utilizado componentes da especificação jax-rs implementados pelo o *Wildfly* através do RESTeasy.

O RESTEasy é um projeto do JBoss que fornece vários frameworks para ajudá-lo a construir serviços Web RESTful e aplicativos Java RESTful. É uma implementação totalmente certificada e portátil da especificação JAX-RS 2.1, uma especificação JCP que fornece uma API Java para serviços Web RESTful sobre o protocolo HTTP (https://resteasy.github.io/, 2018).

O RESTEasy foi utilizado para construir a API Web, a qual o aplicativo android consumiu para realizar as operações que ele se propõe a realizar.

O JBoss foi utilizado por fornecer diversas implementações para as especificações que foram publicadas na JCP - Java Community Process.

6.3 Web API RESTful

A API Web RESTful foi implementada juntamente com o módulo web utilizando serviços da camada EJB.

Os recursos disponibilizados para acesso através desta API foram através dos endpoints disponibilizados.

Toda aplicação Android consumiu essa API para o seu funcionamento. Foi utilizado o POSTMAN para realizar testes nos serviços RESTful como pode ser visualizado na FIG. 33, o teste de autenticação de um usuário.

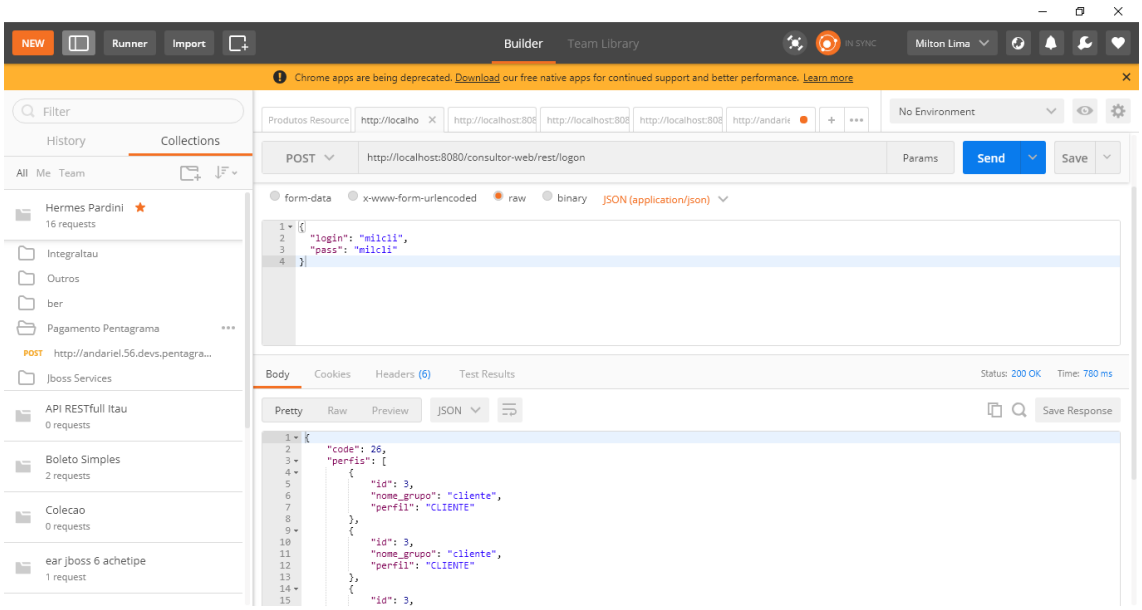


Figura 33 - Consumo da Web API RESTful

6.4 Persistência

Na persistência de dados utilizou-se Java Persistence API - JPA para realização do mapeamento objeto-relacional. O provider utilizado foi o Hibernate que foi configurado no arquivo persistence.xml.

O arquivo persistence.xml é o arquivo responsável pela configuração da utilização da API de persistência do java. Esse arquivo de configuração pode ser visualizado na FIG. 34.

```xml
<?xml version="1.0" encoding="UTF-8"?>

<persistence version="2.1"
    xmlns="http://xmlns.jcp.org/xml/ns/persistence" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="
        http://xmlns.jcp.org/xml/ns/persistence
        http://xmlns.jcp.org/xml/ns/persistence/persistence_2_1.xsd">
    <persistence-unit name="milton-pu">

        <jta-data-source>java:jboss/datasources/consultorDS</jta-data-source>

        <class>br.miltex.consultor.dominio.model.Autenticacao</class>
        <class>br.miltex.consultor.dominio.model.Cliente</class>
        <class>br.miltex.consultor.dominio.model.Endereco</class>
        <class>br.miltex.consultor.dominio.model.Fornecedor</class>
        <class>br.miltex.consultor.dominio.model.Grupo</class>
        <class>br.miltex.consultor.dominio.model.Pessoa</class>
        <class>br.miltex.consultor.dominio.model.Produto</class>
        <class>br.miltex.consultor.dominio.model.Telefone</class>
        <class>br.miltex.consultor.dominio.model.Vendedor</class>
        <class>br.miltex.consultor.dominio.model.Venda</class>
        <class>br.miltex.consultor.dominio.model.VendaPK</class>
        <class>br.miltex.consultor.dominio.model.ParcelaVenda</class>

        <properties>
            <!-- Properties for Hibernate -->
            <!-- <property name="hibernate.hbm2ddl.auto" value="create-drop" /> -->
            <property name="hibernate.show_sql" value="true" />
        </properties>
    </persistence-unit>
</persistence>

```

Figura 34 - Arquivo de Configuração do JPA

Todo o mapeamento objeto-relacional do modelo entidade-relacionamento foi realizado utilizando anotações do JPA, por exemplo, o @Entity para demarcar que uma classe é uma entidade que será persistida no banco de dados.

Um trecho da classe que representa a entidade telefone com as anotações do JPA pode ser visualizada na FIG. 35 , esta classe é a representação de uma entidade do domínio da aplicação mapeada.

```
21
22  @Entity
23  public class Telefone implements AbstractEntity {
24
25      private Long id;
26      private String numero;
27      //private Pessoa pessoa; //Talves pode sair
28      private Timestamp data_ult_atualizacao;
29      private String usu_ult_atualizacao;
30      private Long idPessoa;
31      private String tipo;
32
33      @Id
34      @GeneratedValue(strategy=GenerationType.IDENTITY)
35      public Long getId() {
36          return id;
37      }
38
39      public void setId(Long id) {
40          this.id = id;
41      }
42
43      public String getNumero() {
44          return numero;
45      }
46
47      public void setNumero(String numero) {
48          this.numero = numero;
49      }
```

Figura 35 - Trecho da Entidade Telefone

6.5 Aplicativo Android

O aplicativo Android foi desenvolvido utilizando-se a SDK Android e o Android Studio. O aplicativo consome os serviços disponibilizados na Web API RESTful do projeto. O serviço de autenticação é utilizado na tela de login do aplicativo como pode ser visualizado na FIG. 36. Essa tela utiliza o usuario e senha para realizar a autenticação do usuário para acessar as demais funcionalidades do App.

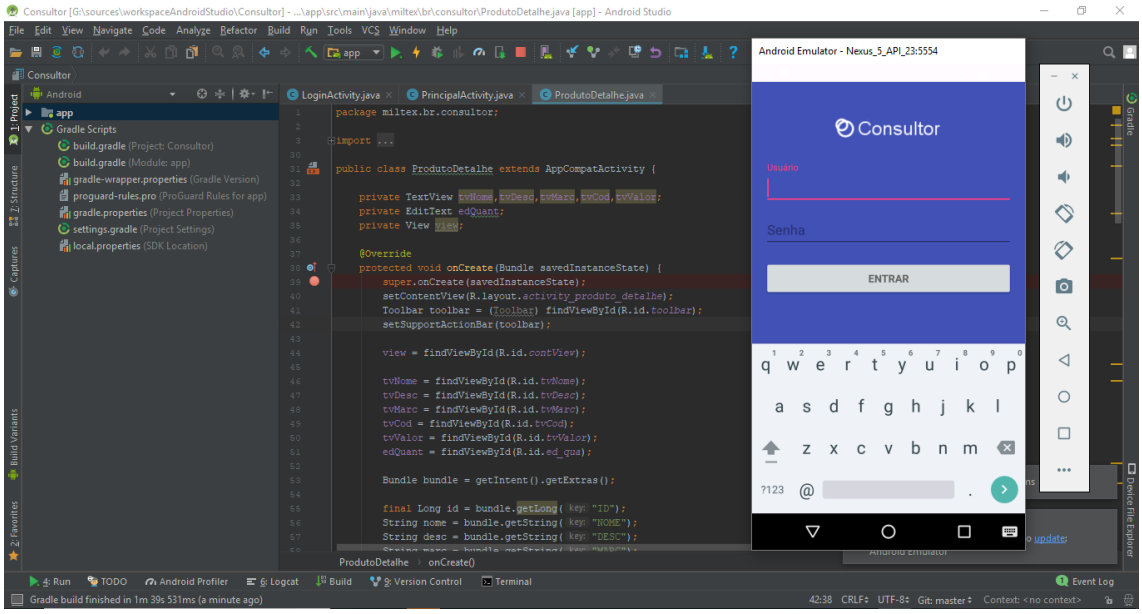


Figura 36 - Tela de Login do Aplicativo Móvel

No menu do App, pode se ter acessos a outras funcionalidades.

Basicamente o usuário do aplicativo pode visualizar todos os produtos que o seu vendedor tem no seu estoque. O usuário também pode realizar um pedido dos produtos que estão no estoque do seu vendedor. O App mostra para o seus usuários todos os seus pedidos para ele saber quais itens ele comprou.

7 CONCLUSÃO

Com o desenvolvimento deste trabalho foi possível colocar todos os conceitos abordados no curso de Engenharia de Software da UFMG em prática. Foi possível aplicar os conceitos de gerenciamento de projeto, engenharia de requisitos, processo de desenvolvimento de software com práticas ágeis, modelagem e implementação de banco de dados, arquitetura de software e programação de sistemas de software.

O trabalho realizado deu origem a um produto de software com funcionalidades que podem ser estendidas de forma bem simples seguindo a arquitetura na qual a aplicação foi construída.

O processo de desenvolvimento deste produto de software foi de muita importância para o meu desenvolvimento como profissional, foi possível passar por todas as grandes áreas do desenvolvimento de um software corporativo.

7. REFERÊNCIAS

Fowler, M.; Lewis, J. Link do artigo na web. *Microservices a definition of this new architectural term.* 25 March 2014.Disponivel em : < http://www.service-architecture.com/articles/web-services/service-oriented_architecture_soa_definition.html > . Acesso em: 10 de Agosto de 2017.

S M Sohan; *et al*. Automated Example Oriented REST API Documentation at Cisco, 2017

Barry, D. Link do artigo na web. *Service Architecture.* Disponivel em : < http://www.service-architecture.com/articles/web-services/service-oriented_architecture_soa_definition.html > Acessado em 12 de Agosto de 2017

Fowler, M. Patterns of Enterprise Application Architecture, 2003

Burke, B. RESTful Java with JAX-RS 2.0 - Designing and Developing Distributed Web Services, 2013

Schwaber, K.; Sutherland, J. The Scrum Guide - The Definitive Guide to Scrum: The Rules of The Game, 2017

Mehta, B. RESTful Java Patterns and Best Practices,