UNIVERSIDADE POLITÉCNICA

A POLITÉCNICA

Escola Superior de Gestão, Ciências e Tecnologias

LICENCIATURA EM ENGENHARIA INFORMÁTICA E DE TELECOMUNICAÇÕES

PROPOSTA DE IMPLEMENTAÇÃO DE UM SISTEMA DE GESTÃO DE VENDAS DE MEDICAMENTOS

CASO DE ESTUDO: FARMÁCIA SAUL

EDILSON DOMINGOS ARAÚJO DA SILVA

MAPUTO

2020

UNIVERSIDADE POLITÉCNICA

A POLITÉCNICA

Escola Superior de Gestão, Ciências e Tecnologias

LICENCIATURA EM ENGENHARIA INFORMÁTICA E DE TELECOMUNICAÇÕES

PROPOSTA DE IMPLEMENTAÇÃO DE UM SISTEMA DE GESTÃO DE VENDAS DE MEDICAMENTOS

CASO DE ESTUDO: FARMÁCIA SAUL

EDILSON DOMINGOS ARAÚJO DA SILVA

MAPUTO

2020

PROPOSTA DE IMPLEMENTAÇÃO DE UM SISTEMA DE GESTÃO DE VENDAS DE MEDICAMENTOS

CASO DE ESTUDO: FARMÁCIA SAUL

Trabalho de Projecto apresentado à Universidade Politécnica – A Politécnica, como parte dos requisitos para a obtenção do grau de Licenciatura em Engenharia Informática e de Telecomunicações.

O candidato Edilson Domingos Araújo da Silva, estudante do curso de Licenciatura em Engenharia Informática e de Telecomunicações, nesta Universidade, realizou o seu Trabalho final na área de desenvolvimento de *software* com o tema "Proposta de implementação de um sistema de Gestão de vendas de medicamentos", tendo desta forma, aplicado os conhecimentos adquiridos ao longo da sua formação.

O trabalho desenvolvido cumpre com as normas de escrita e apresentação de trabalhos desta Universidade, bem como com o grau pelo qual se candidata, pelo que eu, Carlos Gaide Manhique, recomendo a submissão do trabalho para defesa pública conforme as normas da Universidade Politécnica, abaixo, subscrevo.

_____

Tutor: Carlos Gaide Manhique, Msc.

Maputo Julho de 2020

Declaração de honra

Declaro por minha honra que este trabalho é resultado da minha pesquisa pessoal e das orientações do meu tutor, feito segundo os critérios em vigor na Universidade Politécnica. O seu conteúdo é original e todas as fontes consultadas estão devidamente mencionadas no texto e na Bibliografia.

Declaro também que este trabalho não foi apresentado em nenhuma Instituição para obtenção de qualquer Grau Académico.

Maputo, Julho de 2020

_____

(*Edilson Domingos Araújo da Silva*)

Dedicatória

Dedico o presente trabalho aos meus pais que tornaram real o sonho de formar me na presente área, dedico também em especial destaque ao meu filho Thiago. Dedico também a minha família e minha namorada pelo apoio incondicional durante este todo tempo.

Agradecimentos

Em primeiro lugar agradeço a DEUS por conceder a força de vontade, saúde e perseverança que me conduziu até a conclusão desta parte da jornada. Agradeço também aos meus Pais por tudo que têm feito por mim apesar das constantes dores de cabeça durante a jornada. Agradeço também a minha namorada pelas diversas horas de contínuas conversas e discursos motivadores e por sempre estar ao meu lado nas horas mais difíceis. Finalmente gostaria de agradecer aos meus colegas de turma Ivo Sancho e Euclides Moiane pelo suporte constante e ajuda entre nós.

Por fim quero agradecer ao meu tutor, Eng. Carlos Manhique por ter me acolhido em suas asas e transmitir os seus conhecimentos na supervisão, avaliação e auxilio da reprodução do presente trabalho.

Resumo

Este trabalho propõe a implementação de um sistema de informação para ajudar na administração dos negócios de uma farmácia. Devido a crescente evolução das tecnologias ligadas a área de informática, empresas buscam cada vez mais ferramentas tecnológicas, sendo que estas ferramentas por sua vez contribuem principalmente para à tomada de decisões e por outro lado afectam o cotidiano do negócio.

Para o desenvolvimento do *software*, foi utilizada a linguagem de programação Java com o auxilio de *frameworks* como o *hibernate*, PrimeFaces e JavaServer Faces (JSF), juntamente com uma arquitetura de base de dados relacionais utilizando o sistema de gestão de base de dados MySQL. Primeiramente foram analisados o ambiente interno e o actuais processos de compra e venda para modelar o sistema. Em todo o processo de desenvolvimento do programa computacional, foram utilizados métodos, técnicas e ferramentas da engenharia de *software*. Conclusão tirada é que o objectivo fundamental deste trabalho foi alcançado, como resultado temos o sistema de gestão de vendas proposto.

Palavras-chave: Desenvolvimento de *Software*, Sistemas de Informação, desmaterialização.

Abstract

This work proposes the implementation of an information system to help in the administration of a pharmacy business. Due to the growing evolution of technologies related to the IT area, companies are increasingly looking for technological tools, and these tools in turn contribute mainly to decision making and, on the other hand, affect the daily business.

For the development of the software, the Java programming language was used with the help of frameworks such as hibernate, PrimeFaces and JavaServer Faces (JSF), together with a relational database architecture using the MySQL database management system. Firstly, the internal environment and the current buying and selling processes were analyzed to model the system. Throughout the computer program development process, software engineering methods, techniques and tools were used. Conclusion drawn is that the fundamental objective of this work has been reached, as a result we have the proposed sales management system.

Keywords: Software Development, Information Systems, dematerialization.

Lista de Acrônimos

API……...*Application Programming Interface*
BD…….Base de dados
CRUD……..*Create – Read – Update - Delete*
CSS……...*Cascade Style Sheet*
DAO……..*Data Access Object*
GUI……...*Graphical User Interface*
HQL…….*Hibernate Query Language*
IDE……*Integrated Development Environment*
JSF……...*Java Server Faces*
JSP…….*Java Server Pages*
LDAP……...*Lightweight Directory Access Protocol*
MVC…….*Model-View-Controller*
ORM……*Object Relational Mapping*
PF…….*Prime Faces*
SGBD……..Sistema de Gestão de Base de Dados
SI……..Sistema de Informação
SQL……..*Structured Query Language*
UML……*Unified Modeling Language*
XHTML……..*eXtensible Hypertext Markup Language*
XML……*eXtensible Markup Language*

Lista de Figura

Figura 1- Compontentes de um Sistema de Informação. 4
Figura 2 - Diagrama do modelo MVC. 10
Figura 3 - Estrutura do padrão DAO. 12
Figura 4 - Arquitectura do Hibernate 14
Figura 5- Configuração do Servlet do JSF no projecto. 16
Figura 6 - Ciclo de Vida do JSF. 17
Figura 7 - A Linguagem UML 18
Figura 8 - Modelagem da Base de Dados do Sistema 22
Figura 9 - Diagrama de Casos de Uso do Sistema 23
Figura 10 - Diagrama de Sequência do Caso de Uso Efectuar Venda 26
Figura 11 - Ficheiro de configuração do Hibernate no projecto 33
Figura 12 - Camada DAO do funcionário contendo o método Salvar. 33
Figura 13 - Tela de Login do Sistema. 34
Figura 14 - Tela de Cadastro do Fornecedor 34
Figura 15 - Tela de Cadastro do Funcionário 35
Figura 16 - Tela de Finalização da Venda 35

Glossário

***Software* –** Instruções (programas de computador) que, quando executados, fornecem características, funções e desempenho desejados (Pressman, 2011).

Base de Dados – consiste num conjunto integrado de dados, utilizável para múltiplos objectivos e acessível por múltiplos tipos de utilizadores de uma forma concorrente, reflectindo os propósitos fundamentais de integração, partilha, concorrência, múltiplos objectivos, múltiplos tipos de utilizadores (Neves e Ruas, 2005).

Sistema de Gestão de Base da Dados (SGBD) – consiste no *software* que gere todo acesso a uma ou mais base de dados, permitindo a definição, acesso concorrente, manipulação e controlo dos dados, assegurando a integridade, segurança e recuperação das bases de dados (Neves e Ruas, 2005)**.**

Hibernate – Solução para mapeamento de objecto relacional (ORM) e solução de gestão de persistência ou camada persistente (Hennebrueder, 2013).

PrimeFaces – é uma biblioteca de componentes de interface usuário de código-fonte aberto para aplicativos baseados em JavaServer Faces (Baeldung, 2018).

JavaServer Faces (JSF) – Tecnologia padrão Java para construção de componentes baseados em interfaces web orientados a eventos. Como Java Server Pages (JSP), JSF permite o acesso à dados do lado do servidor e lógica (Tyson, 2018).

Índice

Dedicatória i

Agradecimentos ii

Resumo iii

Abstract iv

Lista de Acrônimos v

Lista de Figura vi

Glossário vii

Capítulo I – Introdução 1

1.2 Objectivos 2

1.2.1 Objectivo Geral 2

1.2.2 Objectivos Específicos 2

1.3 Problema da Pesquisa 2

1.4 Perguntas a investigar e as Hipóteses a considerar 3

1.4.1 Formulação da pergunta a investigar 3

1.4.2 As Hipóteses H0 e H1 3

1.5 Justificativa 3

1.6 Delimitação do tema 3

Capítulo II – Revisão Bibliográfica 4

2.1 Sistemas de Informação (SI) 4

2.1.1 Componentes de um SI 4

2.1.2 Projecto e Implantação 5

2.2 Sistemas de Informação e as organizações 6

2.2.1 Ganhos provenientes do uso dos SI 6

2.3 Técnicas para o desenvolvimento de projectos de Software 8

2.3.1 MVC (Model, View, Controller) 9

2.3.2 DAO 11

2.4 Ferramentas e Tecnologias Usadas para a implementação do projecto 12

2.4.1 Java 12

2.4.2 Eclipse 13

2.4.3 MySQL 13

2.4.4 JDBC 13

2.4.5 Hibernate 14

2.4.6 JSF (Java Server Faces) 15

2.4.9 Astah Community 18

2.4.10 Testes Unitários (*JUnit*) 19

Capítulo III – Metodologia 20

Capítulo IV – Desenvolvimento do Modelo 21

4.1 Descrição do processo actual de Compra e venda de produtos na farmácia 21

4.1.1 Modelagem da Base de dados 21

4.2 Diagrama de Casos de Uso e Lista de Casos de Uso 23

4.2.1 Descrição de Caso de Uso e Diagrama de Sequência 24

4.2.2 Diagrama de Sequênica do Caso de Uso Efectuar Vendas 25

Capítulo V – Conclusões e Recomendações 27

5.1 Conclusões 27

5.2 Recomendações 28

5.3 Referências bibliográficas 29

5.4 Anexos 33

Anexo.A – Configuração do Hibernate no projecto 33

Anexo.B – Camada DAO do Funcionário com o método salvar 33

Anexo.C – Tela de Login do Sistema 34

Anexo.D – Tela de Cadastro do Fornecedor 34

Anexo.E – Tela de Cadastro do Funcionário 35

Anexo.F – Tela de Finalização da Venda 35

Capítulo I – Introdução

As tecnologias de informação tem sido um grande aliado no aperfeiçoamento e dinamização de tarefas nos dias de hoje, na maioria dos casos, elas provem uma secção de benefícios como mais habilidade na realização de tarefas quotidianas assim como uma facilidade de organização de arquivos, produtos, documentação bem como dos próprios funcionários das empresas/organizações principalmente aos clientes, pois o tempo de espera e a demora entre os processos da empresa se tornam menores.

Segundo Pressman (1995) uma empresa com esse tipo de *software* tem vantagens sobre as empresas concorrentes. A farmácia Saúl é uma empresa de pequeno porte que não possui nenhum aparato tecnológico que o auxilie nas suas funções diárias

Para a criação de um *software* é necessário que o desenvolvedor saiba quais são os requisitos específicos de cada empresa. O presente trabalho de pesquisa relata o desenvolvimento de um sistema para a gestão de vendas de produtos de uma farmácia, tendo como objectivo maximizar a sua eficiência e qualidade no serviço diante dos seus clientes e fornecedores. Este *software* será um diferencial para a empresa, pois ela ainda não tem nenhum aparato tecnológico que auxilie em suas tarefas diárias.

Em linhas gerais, o presente trabalho tem como objectivo, o desenvolvimento de um softwar*e*, que visa aumentar a competitividade da empresa, assim como a agilidade e facilidade em suas actividades do dia-a-dia. Numa abordagem inicial, o *software* de gestão em questão deverá abranger cadastro, edição, exclusão, busca e filtragem de produto, fornecedor, utilizador, vendas e cobranças por data, contendo também a entrada e saída de estoque.

Este trabalho está estruturado em cinco capítulos. No primeiro capítulo consta a introdução, a definição do problema e o objectivo geral e específicos do trabalho; No segundo capítulo é apresentada a revisão bibliográfica, abrangendo aspectos importantes relacionados ao assunto do trabalho; No capítulo 3 é apresentada a metodologia definida para este trabalho; O capítulo 4 inclui o desenvolvimento da pesquisa, a análise actual dos processos de compras e a proposta de um sistema de informação; Por fim, o quinto capítulo relata as conclusões gerais da proposta, posteriormente as recomendações para trabalhos futuros, a revisão bibliográfica e os anexos.

1.2 Objectivos

1.2.1 Objectivo Geral

- Propor a implementação de um sistema de gestão de venda de Medicamentos

1.2.2 Objectivos Específicos

- Analisar o actual processo de compra assim como da venda da organização alvo;
- Elaborar o modelo de entidade e relacionamento das entidades identificadas;
- Propor um possível emprego de *software* nas operações de compras e vendas na farmácia escolhida.

1.3 Problema da Pesquisa

A Farmácia Saul é uma empresa localizada na cidade de Maputo – Maxaquene C. Actualmente a empresa não tem controle de medicamentos vendidos, fazendo com que o estoque fique prejudicado, pois só é notada a quantidade reduzida de produtos (ex: medicamentos) quando é necessário realizar uma venda.

A cada venda é necessário um novo cadastro através de planilhas físicas em que são anotados os produtos vendidos e suas respectivas quantidades, desta forma impulsionando a perda de tempo e agilidade da farmácia em suas funções corriqueiras. Sendo que cada vez que é necessário fazer uma consulta o processo é manual.

Outro problema da não informatização da empresa é a falta de controle dos produtos vendidos e seus respectivos vendedores, ou seja, com a informatização da empresa será possível ter consciência que produtos foram vendidos assim como o agente responsável pela sua venda, podendo assim ter um registro para possíveis eventualidades (como fraudes) bem como o responsável do caso.

Posto isto, a farmácia dispondo de um sistema de gestão de vendas que auxilie em suas tarefas diárias, como por exemplo a gestão de productos, poderá aumentar a eficiência e qualidade no serviço diante de seus clientes, pois será facilitada a forma de consulta quanto à disponibilidade de produtos. E também será facilitado o trabalho realizado na empresa, uma vez que para a consulta não será necessário ser realizada a procura em várias planilhas, pois o *software* tratará todas as informações necessárias em um menu de buscas, podendo assim o funcionário agir rapidamente e com maior precisão.

1.4 Perguntas a investigar e as Hipóteses a considerar

1.4.1 Formulação da pergunta a investigar

Até que ponto o Sistema de Gestão de Vendas proposto pode ajudar a aumentar a productividade e qualidade de serviço na Farmácia Saúl?

1.4.2 As Hipóteses H0 e H1

-H (0): com a adoção de um sistema de gestão de vendas por parte da Farmácia em questão não se pode alcançar uma boa qualidade de serviço.

-H (1): com o uso de um sistema de gestão de vendas é possível maximizar a eficiência e qualidade no serviço diante dos clientes e fornecedores da Farmácia Saul.

1.5 Justificativa

Segundo Pressman (1995), uma empresa que possui um *software* tem uma diferenciação dos seus concorrentes, pois pode ao mesmo tempo gerir o negócio, ter o controle dos seus produtos e contar com todas as informações oferecidas pelo *software*.

Este foi o factor motivador para que se planejasse um sistema atendendo aos requisitos que irá suprir as necessidades da Farmácia Saul.

Importante também referir que com a digitalização dos sistemas contribuímos também com um dos grandes problemas que muitas vezes é ignorado, concretamente falando do ecossistema. Diversas organizações ao redor do mundo têm aderido a práticas ecológicas para salvaguardar o meio ambiente.

1.6 Delimitação do tema

Este trabalho foca-se no desenvolvimento de um sistema de gestão de vendas, onde o caso de estudo tem como empresa alvo uma farmácia denominada Farmácia Saul.

Pretende se neste âmbito ver os ganhos que um sistema de gestão de vendas poderá trazer para a farmácia supracitada em termos de maximização da eficiência assim como da rentabilidade e agilidade na realização de todas funções diárias.

Capítulo II – Revisão Bibliográfica

2.1 Sistemas de Informação (SI)

Sistema de Informação (SI) é um conjunto organizado de pessoas, *hardware*, *software*, redes de comunicações e recursos de dados que colectam, transformam e disseminam informações em uma organização (O'brien, 2004).

O SI envolve tecnologias, conceitos comportamentais e aplicações complexas e abstratas. É importante ressaltar que nem todos os sistemas de informação são computadorizados, mas como de facto a maioria é, o termo SI normalmente é utilizado como sistema baseado em computador (O'brien, 2004).

2.1.1 Componentes de um SI

Podemos descrever um sistema como uma união e interação de um conjunto de componentes buscando a realização de um objectivo comum.

A figura 1 sintetiza os componentes básicos dos sistemas de informação necessários para os profissionais de uma organização.

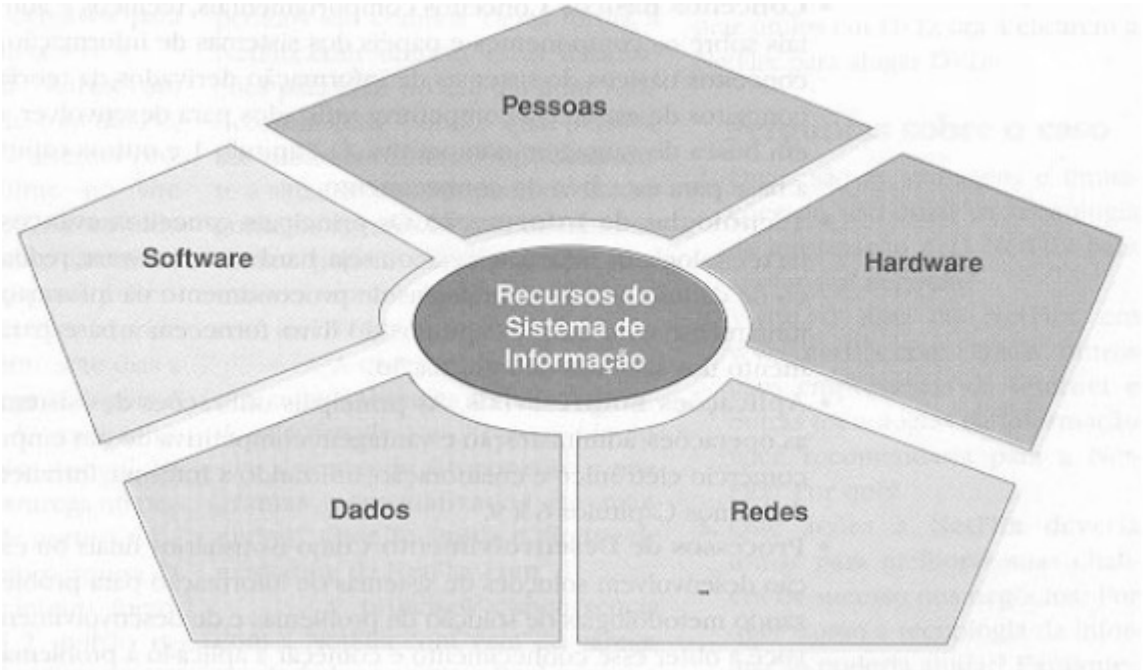

Figura 1- Componentes de um Sistema de Informação (Fonte: O'Brien, 2004).

De acordo com O'Brien (2004), um SI depende basicamente das pessoas, hardware, *software*, uma base da de dados e redes para converter itens de dados em produtos e informação.

a) Pessoas – são todos os indivíduos que utilizam um SI ou a informação que ele produz. Podemos citar: os clientes, vendedores, gerentes, engenheiros, ou seja, quase todos nós estamos em interação com algum SI.
b) *Hardware* – a parte física responsável por colocar o SI em operação, como: monitores impressoras, e computadores. São os dispositivos que recebem dados e informações, os processam e os exibem para as pessoas.
c) *Software* – é a parte intangível que permite que o *hardware* processe os dados, como: planilhas electrónicas, programas de folha de pagamento, programas de processamento de textos entre outros.
d) Base de dados – é uma coleção de arquivos ou tabelas relacionadas que guardam dados já processados. Estes devem ser bem administrados visando o beneficiamento de todos os utilizadores finais da organização.
e) Rede – Sistema fundamental em todos SI para o compartilhamento dos dados, informações e conhecimentos interpretados pelo *software*, entre as pessoas e diferentes computadores.

2.1.2 Projecto e Implantação

Implantar um novo sistema de informação ou efectuar a troca do interior na empresa é uma tarefa complexa, porque enquanto as mudanças ocorrem, a organização continua funcionando normalmente (Mattos, 2005).

As organizações precisam analisar a necessidade da implantação e justificar em termos de custos e benefícios. Nesta análise é preciso avaliar as vantagens do investimento em um SI em relação a investimentos em projectos alternativos.

Outro factor de suma importância é a medição dos riscos envolvidos, alguns estudos mostram que poucos projctos conseguem chegar ao fim, porque a maioria são abandonados ou precisam ser desenvolvidos novamente.

Segundo Mattos (2005), os projectos concluídos também podem gerar falhas e prejuízos. Na análise em busca destas falhas, Mattos afirma que ela pode ocorrer pelos seguintes motivos:

- Redução de custos e prazos para que o projecto seja aprovado mais rapidamente;
- Por problemas de comunicação no levantamento das necessidades do sistema;'
- O utilizador não saber de facto o que deseja;

- Por alterações demasiadas na estrutura da empresa pelo motivo da implantação do SI;
- Elaboração do SI sem uma ampla análise dos dados necessários para a geração da informação objectiva.

2.2 Sistemas de Informação e as organizações

Uma organização tem como característica ser um sistema aberto, que sofre influências do ambiente ao qual está inserida. Milhares de transações comerciais ocorrem diariamente em cada empresa, e são dos sistemas de informação a função de monitorar, colectar, armazenar e processar cada um destes dados sempre de modo eficiente, evitando erros e tempo ocioso, sem nunca perder resultados (Mendes, 2009).

Investir em sistemas de informação é a maneria que as empresas têm para administrar suas funções de produção internas, bem como lidar com as demandas dos actores-chave presentes em seu entorno.

Segundo Laudon & Laudon (2004), as empresas investem em sistemas de informação para atender aos seguintes objectivos organizacionais:

- Atingir a excelência operacional (productividade, eficiência e agilidade);
- Desenvolver novos produtos e serviços;
- Estreitar o relacionamento com o cliente e atendê-lo melhor (marketing contínuo, vendas e serviços; customização e personalização);
- Melhorar a tomada de decisões (em termos de precisão e rapidez);
- Promover a vantagem competitiva;
- Assegurar a sobrevivência.

2.2.1 Ganhos provenientes do uso dos SI

A mensuração das vantagens dos sistemas de informação é dividida na teoria em duas categorias, em ganhos mensuráveis e não mensuráveis. O primeiro deles é o responsável pelo créscimo de algo de melhor na organização, onde o principal exemplo o lucro decorrente de sua implantação, ou seja, se o investimento em sistemas para informatizar as informações for menor que o lucro alcançado os sistemas se tornam vantajosos, caso contrário não. O segundo deles e também os mais comuns são os ganhos não mensuráveis que são particularmente surpreendentes, pois avaliá-los é um dilema decorrente de se

tratarem de benefícios como precisão, oportunidades, actualizações, entre outros (Moresi, 2000).

Para Gouveia (1997), outros exemplos são ganhos tangíveis: redução de despesas, redução de taxas de erros e aumento da capacidade de produção; ganhos intangíveis: melhor gestão da informação, melhor serviço aos clientes e melhor tempo de resposta a pedidos de encomendas.

Os activos não mensuráveis são fontes de benefícios futuros para a organização, geram inovação que no decorrer do tempo fará da organização competitiva (Silva, 2005).

A maneira de identificar o real valor dos SI para as organizações é fazer com que os ganhos mensuráveis e os ganhos não mensuráveis sejam visíveis para o negócio. É fazer dos benefícios intangíveis serem traduzidos em termos quantitativos e para que isso ocorra o valor dos sistemas de informação dependerá muito da forma como se inserem no contexto organizacional, sendo em ambiente interno ou externo e ainda em termos estratégicos ou competitivos (Silva, 2005).

2.2.1.1 Benefícios internos

É clara a vantagem competitiva tanto da tecnologia quanto da informatização das informações no interior das organizações. Os sistemas de informação conseguem alcançar objectivos como agilidades nos processos quando partilham as informações instantaneamente e também criando a ponte entre as tecnologias e as metas da organização e estrutura evolui, as competências dos colaboradores recebem mudanças e sua importância para o negócio cresce e continuará crescendo no futuro (Araújo, 2010).

Os clientes internos (funcionários) necessitam de informações rápidas e precisas para o desenvolvimento de suas tarefas e tomada de decisões (Souza, 2008).

Os sistemas de informação são importantes porque monitoram as actividades e também as pessoas dentro da organização, fornecendo ajuda na fabricação de mercadorias no desenvolvimento dos serviços, controlando peças, estoques, etc. Com isso os processos ficam menos burocráticos com eliminação de resistências e diminuição de pensamentos negativos, permitindo um uso melhor do tempo em benefício para que a empresa se torne uma organização de facto (Souza, 2008).

2.2.1.2 Benefícios externos

A vantagem competitiva almejada pela organização vem demonstrar a necessidade da satisfação principalmente de seus clientes, fornecedores e um destaque diante dos seus concorrentes, com esse enfoque na competitividade os sistemas de informação possibilitam as empresas que o utilizem como estratégia competitiva, ganhando principalmente em eficiência comparativa e poder de barganha no ambiente externo (Pires, 1994).

Os sistemas de informação quando bem estructurados ajudam a organização em ambiente externo por um lado a detectar novas oportunidades por outro lado a auxiliam a se defender das ameaças provenientes da concorrência (Braga, 2000).

Para os SI auxiliar a organização em meio interno precisam fazer uma ponte entre as tecnologias aplicadas e os objectivos ou metas da organização, os sistemas de informação para actuarem em ambientes externos também devem fazer uma ligação entre suas estratégias e o ambiente onde está inserida, realizando uma completa análise da estratégia global da organização (Braga, 2000).

Assim temos que os benefícios ou ganhos dos sistemas de informação tanto internos quanto externos serão de facto obtidos se antes sofrerem uma adaptação ao objectivo da organização a o ambiente em que está inserida, apontando as vantagens que desejáveis e o caminho a traçar.

2.3 Técnicas para o desenvolvimento de projectos de *Software*

Os padrões e técnicas são utilizados tanto para auxiliar na construção de sistemas simples como complexos. Eles visão desenvolver *software* livre de defeitos e de fácil manutenção. Um *software* livre de defeitos é aquele que atende exactamente suas especificações e sempre se comportará conforme esperado pelo cliente.

Construir *software* livre de defeitos não é tarefa fácil para um programador. Além de saber programar satisfatoriamente em uma determinada linguagem de programação, terá que utilizar melhores técnicas de engenharia de *software* e também boas práticas de programação.

É fundamental para se construir qualquer aplicação, adoptar *designer patterns* ou arquitecturas, por ser padrões de alto nível que possibilitam expandir o *software* e fazer reutilização de código fonte.

Padrões de projectos (design *patterns*) são arquitecturas testadas para construir *software* orientado a objectos flexíveis e que podem ser mantidos. O campo dos padrões de projectos tenta aumentar aqueles padrões recorrentes, encorajando os designers de *software* a reutilizá-los para desenvolver *softwares* de melhor qualidade em menos tempo, dinheiro e esforço (Deitel, 2010).

Algumas vantagens do uso de padrões de projectos:

- Alimentam a productividade;
- Representam boas práticas de projecto;
- Permitem obter soluções comprovadas;
- Evita duplicação de código;
- Ajudam a detectar e resolver problemas recorrentes;
- Melhoram e facilitam a manutenção do sistema;
- Facilita a documentação.

Quando não se utiliza um padrão de projecto na construção de um *software* pode-se dizer que o sistema foi desenvolvido sem padrão ou simplesmente é anti-padrão. Em termo simplório, reforçando a ideia de anti-padrão é realmente quando uma aplicação não se detecta fielmente nenhuma base de padrão de projecto apropriado no desenvolvimento de um *software* específico, definitivamente o sistema é desenvolvido ou fabricado a moda antiga.
Um sistema anti-padrão lhe informa com ir de um problema até uma solução ruim (Freeman & Freeman, 2007).

2.3.1 MVC

O MVC (*Model-View-Controller*) é um padrão de arquitetura muitas vezes utilizados em projectos de *software* devido a natureza que o compõem, ou seja, a principal ideologia do MVC é que cada secção de código possui o seu propósito (e cada um diferente do outro).

Segundo Caetano (2013), o grande objectivo do padrão MVC é isolar ao máximo a camada de apresentação. Em essência, o padrão MVC indica que os componentes (*servlets*, por exemplo) de um sistema devem ser divididos em três categorias distintas:

- **Modelo**: composto pelos componentes de entidade e persistência, esta categoria representa os componentes que representam a modelagem de dados.

- **Visão**: composto pelos componentes de apresentação (janelas, formulários etc.), esta categoria representa os componentes que interagem com o utilizador, seja para receber informações, seja para fornecer informações.
- **Controle**: composto pelos componentes de processamento, esta categoria representa os componentes que controlam os processos de negócio, coordenando os outros componentes do sistema para que o resultado final seja aquele esperado pelo utilizador.

Um diagrama simples exemplificando a relação entre *Model, View* e *Controller* é representado na figura 2. As linhas sólidas indicam associação directa e as tracejadas indicam associação indirecta.

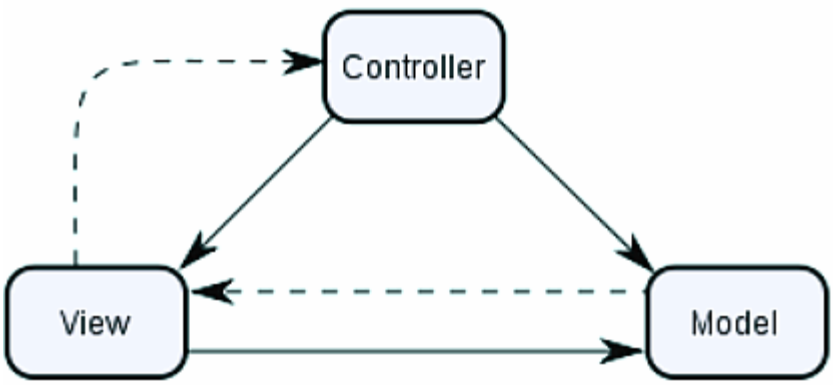

Figura 2 - Diagrama do modelo MVC (Fonte: Caetano, 2013).

Pelo diagrama, se percebe que os componentes do tipo Controle fazem requisições tanto para componentes do tipo Modelo quanto do tipo Visão. Os componentes do tipo Visão, por sua vez, fazem requisições apenas aos componentes do tipo Modelo. Por fim, os componentes do tipo Modelo são passivos, não dando início a requisições para componentes do tipo Controle e Visão.

Majeed e Rauf (2018) listam alguns aspectos que ajudam a decidir quando adoptar ou não a arquitectura MVC na nossa aplicação:

- ✓ A aplicação precisa de uma comunicação assíncrona no *backend*;
- ✓ A aplicação precisa conter uma funcionalidade que resulta em não carregar a página completa por exemplo comentar uma publicação enquanto estiver a usar o *facebook* ou rolara a página infinitamente etc;

- ✓ Manipulação dos dados é na maioria das vezes do lado do cliente (navegador) em vez do lado do servidor;
- ✓ Mesmo tipo de dados estão sendo transferidos de maneiras diferentes numa mesma página;
- ✓ Quando a nossa aplicação tem tantas conexões insignificantes que são usadas para alterar os dados (botão, *switches*).

Majeed e Rauf (2018) apresentam as vantagens do uso da arquitectura MVC:

- ✓ MVC ajuda-nos a controlar a complexidade da aplicação dividindo a em três componentes (Modelo, Visão e Controle);
- ✓ MVC não usa servidor baseado em formulários, por isso ela é ideal para desenvolvedores que queriam controle total sobre o comportamento da sua aplicação;
- ✓ MVC usa o padrão de controlador frontal.

2.3.2 DAO

O DAO (*Data Access Object*) é um padrão para persistência de dados em base de dados. Utiliza a linguagem SQL e busca melhorar as condições das classes separando-as de forma organizada e responsável. Dessa forma torna-se mais viável manter a manutenção do código nesta classe pois é centralizada as operações CRUD (*Creat – Read – Update - Delete*).

DAO é utilizado para abstrair o acesso de dados da camada de negócios (*Business Layer*), que por sua vez, acessa a Base de dados através de *Data Acess Objects*, tornando assim transparente o acesso. A aplicação do padrão DAO é importante porque permite a abstração e encapsulamento de expressões SQL, a camada de negócio consequentemente não sabe nada a respeito da base de dados. Isso leva a uma productividade muito grande e facilidades em manutenções futuras (Aéce, 2005).

Segundo Sun (2007), o DAO esconde completamente os detalhes da execução da origem dos dados de seus clientes, conforme a figura 3.

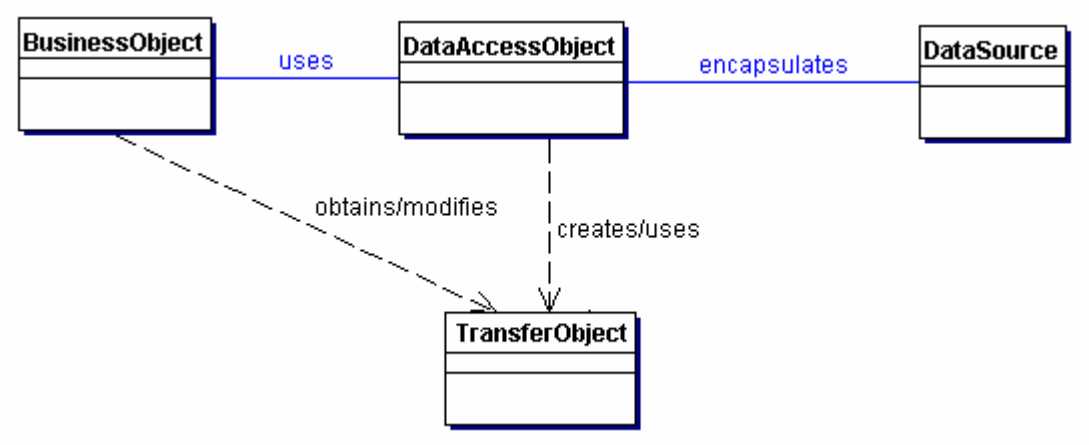

Figura 3 - Estrutura do padrão DAO (Fonte: Sun, 2007).

Conforme Sun (2007), no padrão DAO existem quatro Classes:

a) *BusinessObject*: representa o cliente dos dados. É o objecto que requer à origem dos dados de obter e armazenar dados. Um exemplo pode ser uma tela de cadastro que solicita a busca de uma informação ou armazenamento de um dado na base a qual será conectada.
b) *DataAccessObject*: é objecto principal deste padrão. Ele abstrai a execução do acesso dos dados para que o *BusinessObject* libere o acesso transparente à origem dos dados;
c) *DataSource*: representa a origem dos dados. Uma origem dos dados pode ser uma base de dados, um repositório de XML, outro sistema (legacy/mainframe), um serviço (business-to-business – B2B), ou algum tipo do repositório LDAP.
d) *TransferObject*: representa um objecto de transeferência usado como um portador de dados. O *Data AccessdObject* pode usar esse objecto como retorno dos dados ao cliente. O *DataAccessObject* pode também receber os dados do cliente nesse objecto para actualizar os dados na origem dos dados.

2.4 Ferramentas e Tecnologias Usadas para a implementação do projecto

2.4.1 Java

Java é uma linguagem de programação orientada a objectos amplamente utilizada no mundo. Ela é uma poderosa linguagem de programação (Deitel, 2010).

A linguagem Java foi originalmente projectada para programar dispositivos de consumo popular, mais utilizada pela primeira vez com sucesso para escrever *applets* da Internet (Horstmann, 2004).

Uma razão para a popularidade do Java é a sua independência de plataforma. Isto significa que o mesmo programa compilado pode executar em qualquer computador. Esta independência torna o Java diferente da maioria das outras linguagens de programação, que existem diferentes compiladores para diferentes sistemas operativos (Hubbard, 2006).

2.4.2 Eclipse

O Eclipse é um dos principais ambientes de desenvolvimento integrado para desenvolvimento orientado a objectos utilizando a linguagem de programação Java. Uma ferramenta poderosa para a construção de aplicações Desktop e Web.

A IDE pode ter uma coleção impressionante de ferramentas que podem ser facilmente instaladas em seu ambiente de trabalho, incluindo construtores GUI, ferramentas de modelização, gráficos, relatórios, testes e muito mais (IDE, 2016).

2.4.3 MySQL

O MySQL, é um servidor e gerenciador de base de dados (SGBD) relacional, de licença dupla (sendo uma delas de *software* livre), projectado inicialmente para trabalhar com aplicações de pequeno e médio portes, mas hoje atendendo a aplicações de grande porte e com mais vantagens do que seus concorrentes (Milani, 2006).

O MySQL é usado para armazenar, consultar, recuperar dados dentre outras de forma eficiente e para isso utiliza a linguagem SQL. Ela está disponível para vários sistemas operacionais.

2.4.4 JDBC

O JDBC (*Java Database Conncectivity*) é uma API (*Application Programming Interface*) escrita na linguagem Java para execução e manipulação das instruções SQL.

A tecnologia da biblioteca é bastante utilizada para facilitar o processo de persistência das informações em banco de dados relacionais. Esta API ou driver JDBC é de extrema importância, sem ele não haverá a conectividade da base de dados com a aplicação por isso é muito utilizado pelos programadores que querem manipular dados.

2.4.5 Hibernate

O *framework Hibernate* foi desenvolvido por uma equipe de programadores Java por Gavin e teve sua primeira versão divulgada em 2004. Segundo King (2007), um dos objectivos ao criar o projecto era resolver seus problemas referentes à persistência causados pelo EJB (*Enterprise Java Beans*) 2.0, o qual considerava mais complexo.

Como um escopo muito vasto, o projecto tornou-se inviável de ser mantido apenas nos tempos livres, assim King aceitou entrar para o *JBoss Group*, passando a ser remunerado para continuar a desenvolver o projecto, o que lhe permitiu dedicação completa ao *Hibernate*.

A figura 4 representa o altíssimo nível da arquitectura do *Hibernate* usando a base de dados e a configuração de dados para prover persistência de serviços e persistência de objectos para o aplicativo.

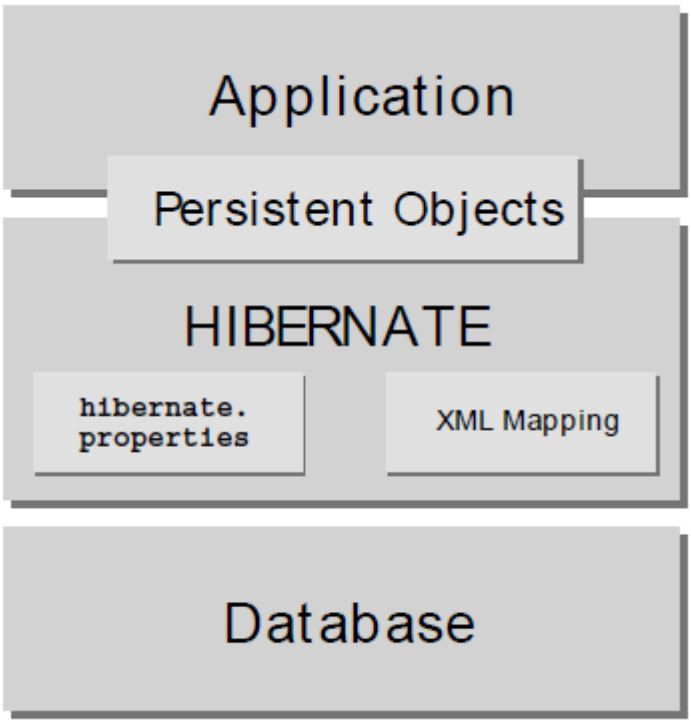

Figura 4 - Arquitectura do *Hibernate* (Fonte: King et al., 2007.)

O *Hibernate* é um dos mais populares *frameworks* de ORM (*Object/Relational Mapping*) no mundo Java. Ele permite aos desenvolvedores mapear estruturas dos objectos e classes *Java* normal para uma estrutura relacional de base de dados. Com a ajuda de um *framework* ORM o trabalho de armazenar dados de instâncias de objectos na memória para um repositório de dados persistente e carrega-los de volta na mesma estrutura de objectos torna se fácil (Mois, 2015).

O *Hibernate* provém JPA, isto significa que ele implementa o *Java Persistence* API (JPA). JPA é uma especificação para mapeamento de objectos Java para tabelas de uma base de dados relacional (Mois, 2015).

Segundo Mois (2015) o *Hibernate* consiste em três componentes diferentes:

- Entidades: as classes que são mapeadas pelo *Hibernate* para as tabelas de um sistema de base de dados relacional são classes Java simples.
- Objecto relacional Meta data: a informação sobre o mapeamento de entidades para uma base de dados relacional é também fornecida por anotações ou através dos ficheiros de configuração XML.
- *Hibernate Query Language* (HQL): ao usar o *Hibernate*, *queries* enviadas para a base de dados não têm de formuladas no SQL nativo, mas podem ser especificadas usando o HQL. Todas essas *queries* são transladadas no tempo de execução no dialeto em uso, *queries* formuladas em HQL são independentes do dialeto SQL.

2.4.6 JSF (Java Server Faces)

JSF é uma tecnologia que nos permite criar aplicações Java para Web utilizando componentes visuais pré-prontos, de forma que o desenvolvedor não se preocupe com *Javascript* e HTML. Basta adicionarmos os componentes (calendários, tabelas, formulários) e eles serão renderizados e exibidos em formato html.

O JSF segue o padrão arquitectural MVC e faz o papel de *Controller* da aplicação. Para começar a usá-lo, é preciso configurar a *servlet* do JSF no web.xml da aplicação. Esse *servlet* é responsável por receber as requisições e delegá-las ao JSF. A figura 5 ilustra a configuração do JSF no projecto da pesquisa.

```
7     <!-- Nome da Aplicação -->
8     <display-name>Farmacia</display-name>
9
10    <!-- Arquivo Principal da Aplicação -->
11    <welcome-file-list>
12        <welcome-file>paginas/autenticacao.xhtml</welcome-file>
13
14    </welcome-file-list>
15    <servlet>
16
17        <!-- Configuração do Servlet do JSF -->
18        <servlet-name>Faces Servlet</servlet-name>
19        <servlet-class>javax.faces.webapp.FacesServlet</servlet-class>
20        <load-on-startup>1</load-on-startup>
21    </servlet>
22    <servlet-mapping>
23        <servlet-name>Faces Servlet</servlet-name>
24        <url-pattern>*.xhtml</url-pattern>
25    </servlet-mapping>
26
```

Figura 5- Configuração xml do Servlet do JSF no projecto (Fonte: Autor, 2019).

Segundo Rocha (2013) o JSF possui as seguintes características:

- Transparência na gestão do estado nas requisições;
- Encapsulamento de diferenças entre navegadores;
- Suporte a processamento multi-página de formulários
- Plataforma extensível (através de bibliotecas de componentes criadas por terceiros);
- Suporte nativo e extensível a validação, eventos e conversão de tipos;
- Controle declarativo e condicional de navegação;

Segundo Egidio (2013), o ciclo de vida do JSF se divide em seis fases:

- Fase 1: Restore View (Restauração da Visão);
- Fase 2: Apply Request Values (Aplicar valores da requisição);
- Fase 3: Process Validation (Processar as validações);
- Fase 4: Update Model Values (Actualizar valores de modelo);
- Fase 5: Invoke Application (Invocar aplicação);
- Fase 6: Render Response (Renderizar a resposta);

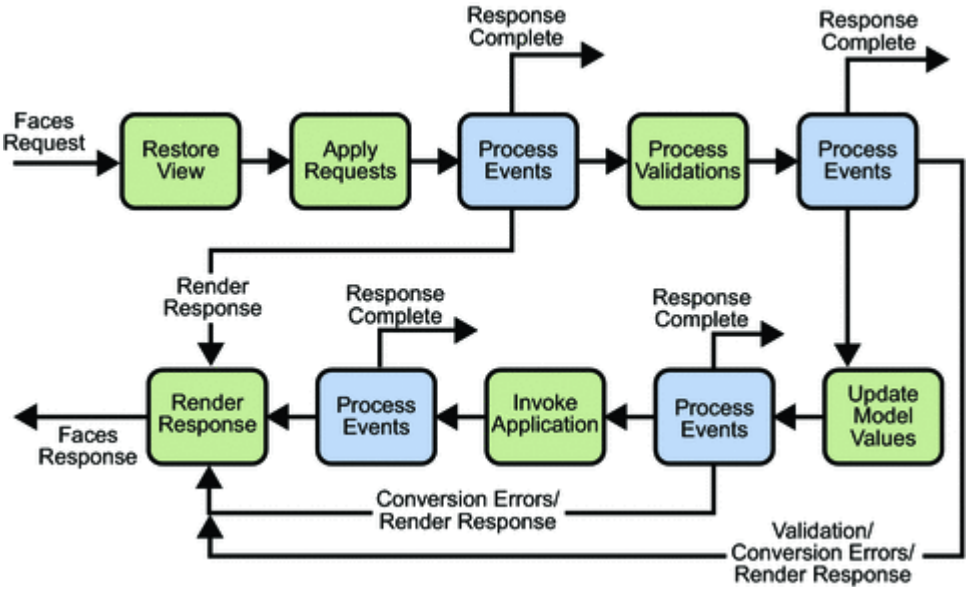

Figura 6 - Ciclo de Vida do JSF (Fonte: Egidio, 2013).

2.4.8 UML

UML (*Unified Modeling Language*) é uma linguagem que se presta à modelagem de estruturas que irão compor uma aplicação, estando fortemente amparada em conceitos de Orientação a Objctos.

É bastante comum que se encontrem literaturas a respeito de design *patterns* que fazem uso da UML, empregando esta última como um meio para a representação esquemática das ideias que estão sendo discutidas. Além disso, metodologias de desenvolvimento como RUP (*Rational Unified Process*) têm nesta linguagem um importante pilar, já que os diversos diagramas existentes acabam por servir como parte da documentação de um projeto (DevMedia, 2019).

A linguagem modelada unificada é usada para criar modelos de diagramas aplicada a um programa orientado a objecto. Os diagramas podem representar partes dos sistemas ou o sistema completo tendo em vista a facilidade do manuseio do *software* por meio da documentação gerada. Os principais diagramas da UML são:

- Diagrama de caso de uso;
- Diagrama de classes;
- Diagrama de objectos;
- Diagrama de colaboração Diagrama de sequencial;
- Diagrama de actividades;
- Diagrama de estado;

- Diagrama de componentes;
- Diagrama de pacotes.

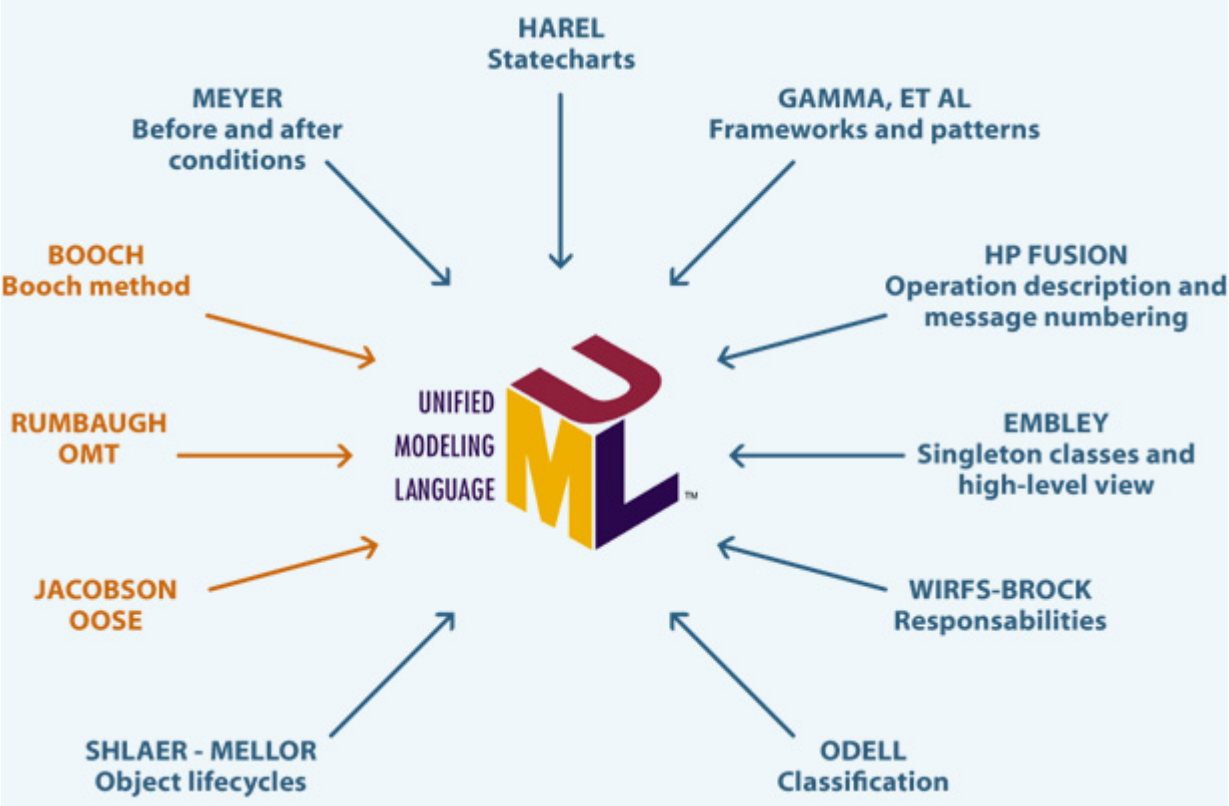

Figura 7 - A Linguagem UML (Fonte: DevMedia, 2019).

2.4.9 Astah Community

É uma ferramenta voltada a desenvolver diferentes tipos de diagramas UML. A versão do programa utilizada é gratuita, além desta, existem outras versões como: Astah UML, Astah Professional e Astah Share que disponibilizam outras funcionalidades além da modelagem UML, porém sua licença é comercial. A modelagem no *Astah Community* é prática e simples.

Dentre os principais diagramas utilizados durante o processo de elaboração do projecto foram digramas de caso de uso e diagrama de sequência.

Além dos citados acima existem outros como: classe, actividade, comunicação, máquina de tempo, componentes, implantação, estrutura de composição, objectos, pacotes, e outros que o próprio programa consegue gerar. Uma funcionalidade muito peculiar do *software* é transformar o diagrama construído em uma imagem para ser utilizada em outro programa.

2.4.10 Testes Unitários (*JUnit*)

JUnit pode ser uma função, uma classe, um pacote, ou um subsistema, portanto, os testes unitários referem-se a prática de testes de pequenas unidades do seu código, para ter a certeza de que que ele funciona como esperado (Dimtsa, 2014).

Está prática ajuda os desenvolvedores a descobrirem falhas na sua lógia por detrás do seu código e melhorar a qualidade do seu código. Também, testes unitários podem ser usados para assegurar que o código irá funcionar como esperado em caso de futura mudanças.

JUnit é um *framework* de testes de código aberto que é usado para escrever e executar testes automatizados repetidamente, para que nos assegurarmos de que o nosso código funciona como esperado.

Capítulo III – Metodologia

Para fazer a revisão geral dos aspectos envolvidos na importância do uso de um sistema de gestão de vendas, foi feita a consulta bibliográfica que consistiu na leitura e síntese do material de interesse para o trabalho, como forma de determinar os principais pontos de análise e a integração destes no caso de estudo em questão.

Para a avaliação do funcionamento da farmácia, foi elaborado um questionário destinado aos responsáveis da farmácia, como forma de compreender os processos actuais de compra e venda, como é que feito o controle de estoque, e dentre outros aspectos.

Para o desenvolvimento do protótipo do sistema será usado o Processo de Modelação Unificado que sugere a UML como modelo de desenvolvimento adequado.

Capítulo IV – Desenvolvimento do Modelo

4.1 Descrição do processo actual de Compra e venda de produtos na farmácia

O processo de venda dos produtos na farmácia é feito da seguinte forma: A cada venda é necessário um novo cadastro através de planilhas em que é anotado o nome do cliente, e os respectivos produtos vendidos, desta forma impulsionando a perda de tempo e agilidade da farmácia em suas funções corriqueiras. Sendo que cada vez que é necessário fazer uma consulta o processo é manual, assim como as datas de cobranças.

4.2 Proposta do Sistema

Com base nas necessidades dos inquiridos foram detectados os requisitos principais para a elaboração do sistema e listados abaixo:

- Gestão dos dados dos funcionários;
- Gestão dos dados dos usuários que manusearão o sistema;
- Gestão dos produtos da farmácia;
- Gestão dos dados das vendas;
- Gestão dos dados dos fornecedores;
- Gestão do estoque.

4.1.1 Modelagem da Base de dados

Nesta parte do projecto foi criada a base de dados que age como chefe de qualquer aplicação que manipula dados consistentemente. Caso ela não esteja bem formatada o *software* pode sofrer com problemas logo no início. O projecto de base de dados foi elaborado a partir dos requisitos firmados junto aos colaboradores da farmácia.

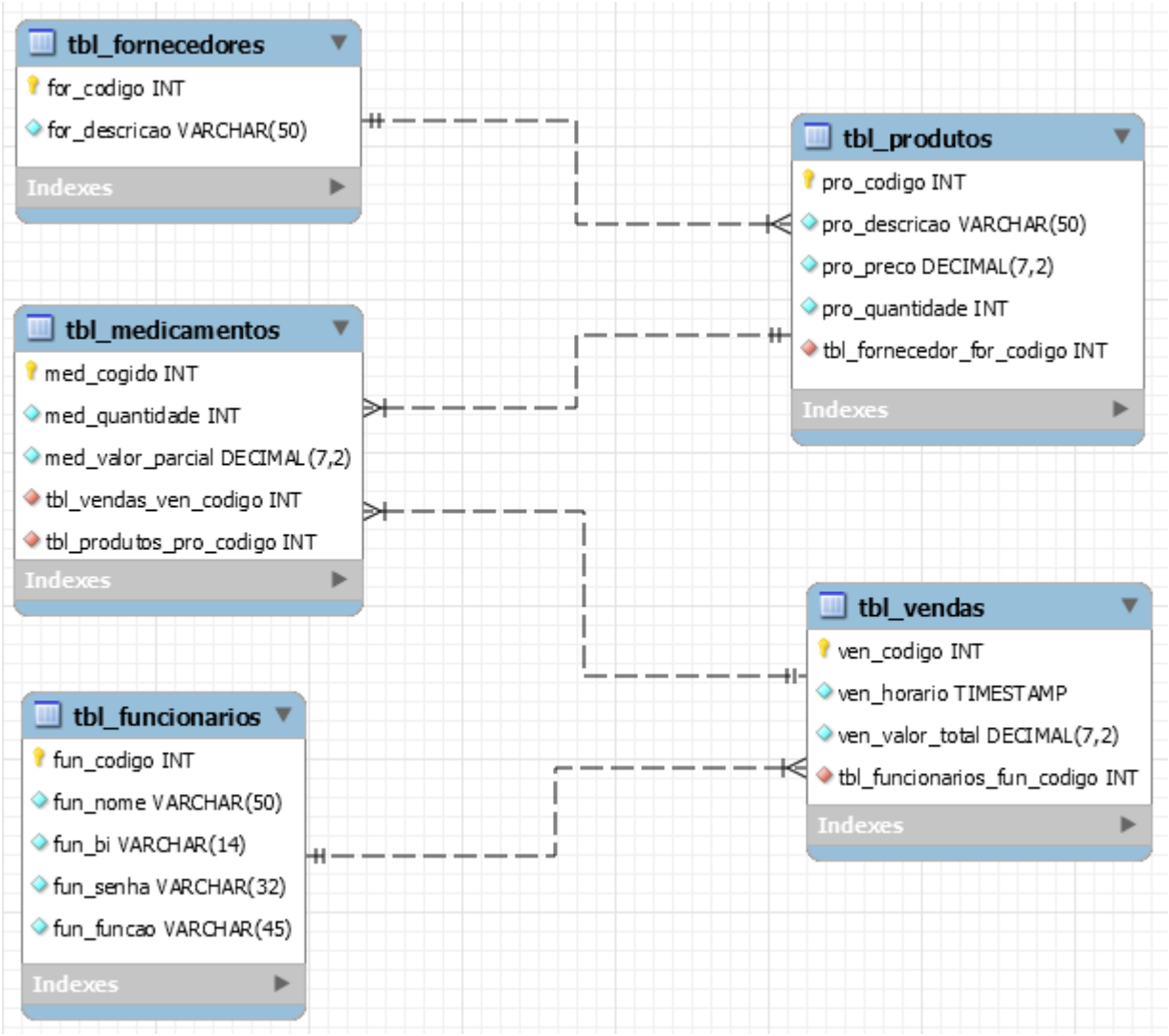

Figura 8 - Modelagem da Base de Dados do Sistema (Fonte: Autor, 2019).

A figura 8 representa a modelagem da base de dados da aplicação. Para isso foi utilizado o *software* para a criação e manipulação de dados, o MySQL. Além disso, mostra a ligação que as tabelas têm umas com as outras de forma eficiente gerando toda a estrutura de comunicação que vai resultar na base de dados do sistema desenvolvida para o TFC (Trabalho de Final de Curso). É no MySQL que o protótipo desenvolvido faz determinadas operações geradas através das telas.

Inicialmente era suposto ter uma tabela de login, no entanto, ela não teria nenhuma ligação com as demais tabelas da BD, nesse caso optou em se usar dois atributos do funcionário (nome e senha), para questões de autenticação dos utilizadores.

4.2 Diagrama de Casos de Uso e Lista de Casos de Uso

Um caso de uso e uma sequência de acções que um ou mais actores realizam num sistema de modo a obterem um resultado particular (OMG, 1998).

O diagrama de casos de uso permite capturar os requisitos de um sistema através do detalhe de todos os cenários que os utilizadores podem realizar. Os casos de uso, mais que iniciar a modelação de requisitos de um sistema, dirigem/conduzem todo o processo de desenvolvimento (Silva e Videira, 2001).

Um caso de uso deve descrever o que faz um sistema (ou parte deste), mas como é que tal é realizado. O foco é a visão externa do sistema, ou seja, na visão que os utilizadores têm dele. A figura 9 representa o digrama de casos de uso do projecto.

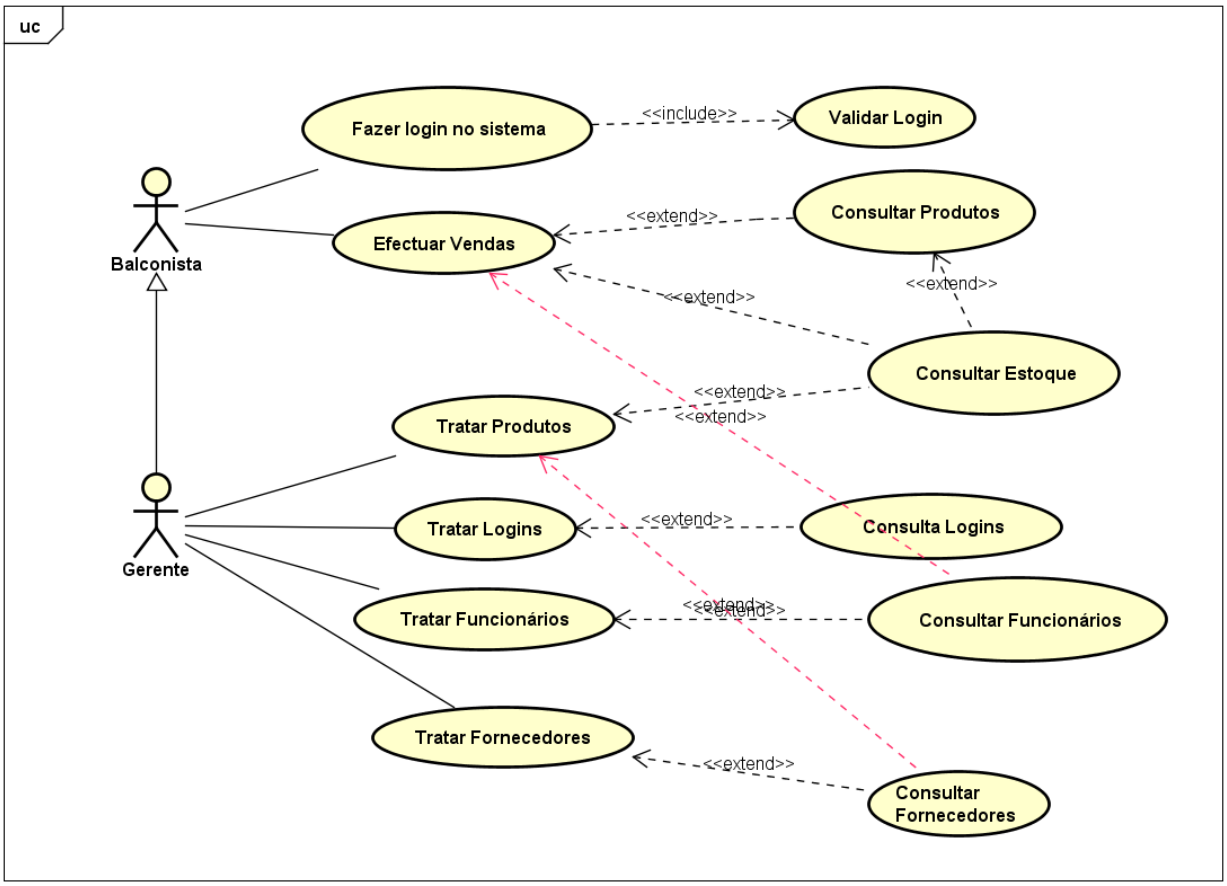

Figura 9 - Diagrama de Casos de Uso do Sistema (Fonte: Autor, 2019).

Lista de casos de uso:

1. Fazer login no Sistema;
2. Tratar as vendas;
3. Tratar Produtos;

4. Tratar Logins;
5. Tratar Funcionários;
6. Tratar Fornecedores.

Atores: Balconista (Farmacêutico) e Gerente.

4.2.1 Descrição de Caso de Uso e Diagrama de Sequência

Os casos de uso são uma ferramenta valiosa para ajudar no entendimento dos requisitos funcionais de um sistema. Um grande perigo dos casos de uso é que as pessoas normalmente os tornam complicados demais e não conseguem prosseguir. Normalmente, você terá menos problemas fazendo pouco o que fazendo demais, umas duas páginas por caso de uso estão bom para a maioria das situações. Se você tiver muito pouco, pelo menos terá um documento curto e legível, que terá um ponto de partida para perguntas. Se você tiver demais, dificilmente alguém o lerá e o entenderá (Fowler, 2005).

Para fins de documentação do *software* não serão elaborados todos os casos de uso listados, pois implicará em uma documentação muito extensa, por isso foi optado por dar uma sequência lógica do caso de uso efectuar vendas de forma parcial com suas principais operações e a relação dos diagramas para cada acção efectuada.

Caso de Uso 2: Efectuar vendas

Autor: Balconista (Farmacêutico)

Descrição: A possibilidade de realizar todas as seguintes operações no programa: salvar, listar, buscar, editar, excluir, cancelar, limpar.

Fluxo Normal:

1. O Balconista solicita a tela de vendas.
2. O sistema exibe a tabela com a lista dos produtos disponíveis e permite fazer as operações do programa.
3. O Balconista busca o produto que o cliente necessita e adiciona ao conjunto de itens para a venda.
4. O Balconista Finaliza e venda e salva o histórico.
5. O caso de uso encerra.

4.2.2 Diagrama de Sequênica do Caso de Uso Efectuar Vendas

Um diagrama de sequência ilustra uma interação segundo uma visão temporal. Um diagrama de sequência é representado através de duas dimensões: a dimensão horizontal, que representa o conjunto de objectos intervenientes; e a dimensão vertical que representa o tempo.

A representação destas dimensões pode ser invertida, se for conveniente. Não existe qualquer significado na ordenação horizontal dos objectos intervenientes, ou seja, na sua disposição relativa.

Nos diagramas de sequência as setas são desenhadas horizontalmente de forma a representar a indivisibilidade da operação necessária para enviar estímulo.

A figura 10 ilustra o diagrama de sequência do caso de uso efectuar venda em que são representados diferentes tipos de mensagens.

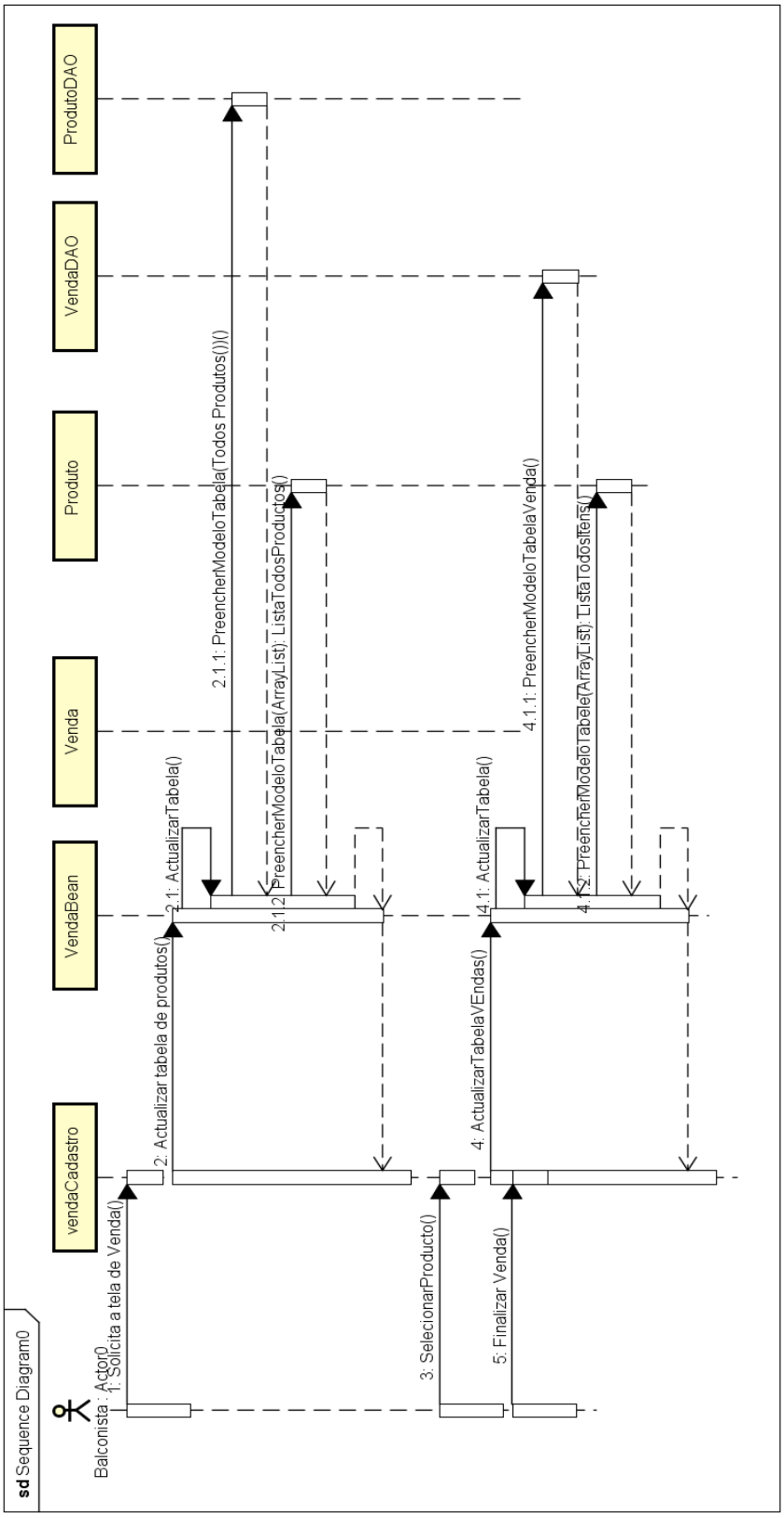

Figura 10 - Diagrama de Sequência do Caso de Uso Efectuar Venda (Fonte: Autor, 2019).

Capítulo V – Conclusões e Recomendações

5.1 Conclusões

Justificadamente, a concepção e o desenvolvimento de um *software* para apoiar os processos em uma área organizacional, representam uma parcela considerável dos esforços dos gestores para aprimorar e contribuir com as operações diárias.

O desenvolvimento deste trabalho proporcionou uma reflexão sobre os inúmeros aspectos relacionados ao problema. Este trabalho permitiu a elaboração e validação de um SI para apoiar as tarefas relaciondas às operações de compra e venda. Mais especificamente, a implantação desta proposta proporcionará um controle geral sobre as operações realizadas em períodos anteriores como também facilitará cada actividade realizada pelos colaboradores que utilizam destas operações para realizar tarefas pertinentes às outras, como o controle financeiro.

Foi realizada uma revisão teórica dos Sistemas de Informação, seus componentes, processo de implantação. Este processo de revisão subsidiou a análise dos procedimentos actuais adoptados pela organização e ajudou a definir o método utilizado para modelar o sistema.

Como é possível perceber, principalmente no capítulo 4, que não houveram tentativas de entrega do sistema, o que não é muito normal na área de desenvolvimento de *software*. Porém, esse projecto é considerado bem-sucedido não pelo facto de ter sido implantado e satisfazer as necessidades do cliente, mas por ele seguir e aplicar os processos bem definidos para o desenvolvimento de sistemas web.

De uma forma geral conclui-se que o objectivo fundamental deste trabalho foi alcançado, como resultado temos o sistema de gestão de vendas que ficará como legado para a Universidade Politécnica, como forma de agradecimento por todo o conhecimento provido.

5.2 Recomendações

O projecto do Sistema de Gestão de Vendas pode servir como referência para futuros trabalhos da mesma natureza, com a possibilidade de desenvolver novas funcionalidades com a capacidade de entrega de mais robustez e complexidade no desenvolvimento de sistemas relacionados. Portanto dois requisitos não funcionais que não foram levados em consideração durante o processo do desenvolvimento do sistema foram escalabilidade e interoperabilidade.

5.3 Referências bibliográficas

Deitel, P, J. (2010) *Java: como programar.* São Paulo: Person Prentice Hall. [8ª edição].

Felício, B., de C e Teodoro, E. (2014) *Revista dos alunos de Sistemas de Informação*.

Fowler, M. (2003) *Arquitetura de Aplicações Corporativas*. Porto Alegre: Bookman.

Horstmann, C. (2004) *Big Java*. Porto Alegre: Bookman.

Hubbard, J. R (2006) *Teoria e problemas da programação com Java*. Porto Alegre: Bookman. [2a edição].

LAUDON, K. C. e LAUDON J. P. (2004) *Sistemas de Informação Gerenciais: Administrando a Empresa Digital.*São Paulo: Prentice Hall. [5ª edição].

Mattos, A. C. M. (2005) *Sistemas de infomação: Uma visão executiva*. São Paulo: Saraiva.

Mendes, F. C. (2009) *Administração de Sistemas de Infomação Vol I.* Rio de Janeiro: Tereza Queiroz.

Milani, A. (2006) *MySQL: Guia do Programador.* São Paulo: Novatec.

Neves, P. M. e Ruas, R. P. (2005) *O Guia Prático do MySQL.* Lisboa: Centro Atlântico. [1a Edição]

Object Managament Group. (1998) XMI Partners. *XML Metadata Interchange (XMI)*.

O´BRIEN, J. A. (2004) *Sistemas de informação e as decisões gerenciais na era da Internet*.São Paulo: Saraiva. [2ª edição].

Pressman, R. S. (1995) *Engenharia de Software: Uma abordagem profissional*. Porto Alegre: AMGH Editora. [7a edição].

Pressman R. S. (2010) *Engenharia de Software*, AMGH Editora. [6a edição].

Pressman, R. S. (2011) *Engenharia de Software*. São Paulo: Makron Books. [6a edição].

Silva, A. & Videira, C. (2001) *UML: Metodologias e Ferramentas Case*. Lisboa: Centro Atlântico. [1a Edição].

Sommerville, I. (2007). *Engenharia de Software*. Addison Wesley. [8a Edição].

Aéce, I. (2005) *Analisando o Microsoft Petshop*. <http://www.projetando.net/Sections/ViewArticle.aspx?ArticleID=14>. Consultado em: 08-10 2019.

Araújo, M. J. E. (2010) *A evolução dos departamentos de TI* <http://student.dei.uc.pt/~maraujo/csi/artigo2.htm>. Consultado em: 20-09-2019.

Baeldung (2018) *Introduction to Primefaces*. <https://www.baeldung.com/jsf-primefaces>. Consultado em 10-09-2019.

Caetano, D. (2013) *Programação Servidor para Sistemas Web*. < http://www.caetano.eng.br/aulas/2013b/getfile.php?fn=psw_ap06.pdf>. Consultado em: 05-09-2019.

Braga, A. (2000) *A gestão da informação.* <http://www.ipv.pt/millenium/19_arq1.html>. Consultado em: 09-09-2019.

Devmedia (2019). *Modelagem de Sistemas através de UML: Uma visão geral*. < https://www.devmedia.com.br/modelagem-de-sistemas-atraves-de-uml-uma-visao-geral/27913>. Consultado em: 02-09-2019.

Dimtsa, K. (2014) *JUnit Tutorial for Unit Testing.* The Ultimate Guide. < http://enos.itcollege.ee/~jpoial/allalaadimised/reading/JUnit-Tutorial.pdf>. Consultado em: 14-11-2019.

Egidio, A. F. (2013) *Desenvolvimento Web Java.*

< https://www.javaavancado.com/ebooks/JSF_PrimeFaces_Hibernate.pdf>. Consultado em: 20-10-2019.

Gouveia, L. (1997) *Custos e benefícios*.<http://www2.ufp.pt/~lmbg/cadeiras/gst_cap5.pdf>. Consultado em 08-08-2019.

Hennebrueder, S. (2013). *Guide to Java Persistence and Hibernate* <http://www.lalilуna.com/download/java-persistence-developer-guide.pdf>. Consultado em 10-09-2019.

IDE. (2019) *Eclipse.* <https://eclipse.org/ide>. Consultado em: 13-08-2019.

Majeed, A. & Rauf, I. (2018) *MVC Architecture: A Detailed Insight to the Modern Web Applications Development*. Peer Rev J Sol Photoen Sys. < https://crimsonpublishers.com/prsp/pdf/PRSP.000505.pdf>. Consultado em 13-10-2019.

Mois, M. (2015) *Hibernate Tutorial: The Ultimate Guide.* <http://enos.itcollege.ee/~jpoial/allalaadimised/reading/Hibernate-Tutorial.pdf> Consultado em: 12-10-2019.

Moresi, E. A. D. (2000) *Delineando o valor do sistema de informação de uma organização*. <http://www.scielo.br/pdf/ci/v29n1/v29n1a2.pdf>. Consultado em 23-07-2019.

Rocha, H., da (2013) *Java Server Faces: Breve introdução prática*. < http://www.argonavis.com.br/palestras/java/j559/JSF2MiniCurso.pdf> Consultado em 20-10-2019.

Pires, B. C. C. (1994) *Sistemas de informação inter-organizacionais vantagem competitiva: um estudo de caso em uma agência de viagens e turismo.*

<http://www.ead.fea.usp.br/cad-pesq/arquivos/C00-art06.pdf>. Consultado em 09-07-2019.

Silva, S. L,. da (2004) *Gestão do conhecimento uma revisão crítica: orientada pela abordagem da criação do conhecimento*. <http://www.scielo.br/pdf/%0D/ci/v33n2/a15v33n2.pdf>. Consultado em 03-07-2019.

Souza, M. S. (2008) *A utilização da TI como ferramenta para atuar na estão Organizacional*. <http://books.google.com.br/books?>. Consultado em 20-08-2019.

SUN. (2007) *Core J2EE patterns: data access object*. http://java.sun.com/blueprints/corej2eepatterns/Pattrerns/DataAccessObject.html>. Consultado em 15-09-2019.

Tyson, M. (2018) *The Java Web Development Series.* <https://www.javaworld.com/article/3322533/what-is-jsf-introducing-javaserver-faces.html>. Consultado em 10-09-2019.

5.4 Anexos

Anexo.A – Configuração do Hibernate no projecto

hibernate.cfg.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE hibernate-configuration PUBLIC
"-//Hibernate/Hibernate Configuration DTD 3.0//EN"
"http://www.hibernate.org/dtd/hibernate-configuration-3.0.dtd">

<hibernate-configuration>

    <!-- Responsável por fazer as conexões -->
    <session-factory>

        <!-- Configurações da conexão -->

        <property name="connection.driver_class">com.mysql.cj.jdbc.Driver</property>
        <property name="connection.url">jdbc:mysql://localhost:3306/db_farmacia</property>
        <property name="connection.username">root</property>
        <property name="connection.password">Eit2019Ap#</property>

        <!-- Pool de conexões JDBC -->
        <property name="connection.pool_size">1</property>

        <!-- Dialecto SQL (Idioma entre o Hibernate e o MySQL) -->
        <property name="dialect">org.hibernate.dialect.MySQL5InnoDBDialect</property>

        <!-- Gerenciamento de sessão -->
        <property name="current_session_context_class">thread</property>


        <!-- Desabilita a cache -->
        <property name="cache.provider_class">org.hibernate.cache.internal.NoCacheProvider</property>


        <!-- Exibir os comandos SQL (Mostrar os comandos para ver o que o Hibernate
            está fazendo) -->
        <property name="show_sql">true</property>
```

Figura 11 - Ficheiro de configuração do Hibernate no projecto (Fonte: Autor, 2019).

Anexo.B – Camada DAO do Funcionário com o método salvar

FuncionarioDAO.java

```
public class FuncionarioDAO {

    public void salvar(Funcionario funcionario) {
        Session sessao = HibernateUtil.getSessionFactory().openSession();
        Transaction transacao = null;
        try {
            transacao = sessao.beginTransaction();
            sessao.save(funcionario);
            transacao.commit();

        } catch (RuntimeException ex) {
            if (transacao != null) {
                transacao.rollback();
            }
            throw ex;
        } finally {
            sessao.close();
        }

    }

}
```

Figura 12 - Camada DAO do funcionário contendo o método Salvar (Fonte: Autor, 2019).

Anexo.C – Tela de Login do Sistema

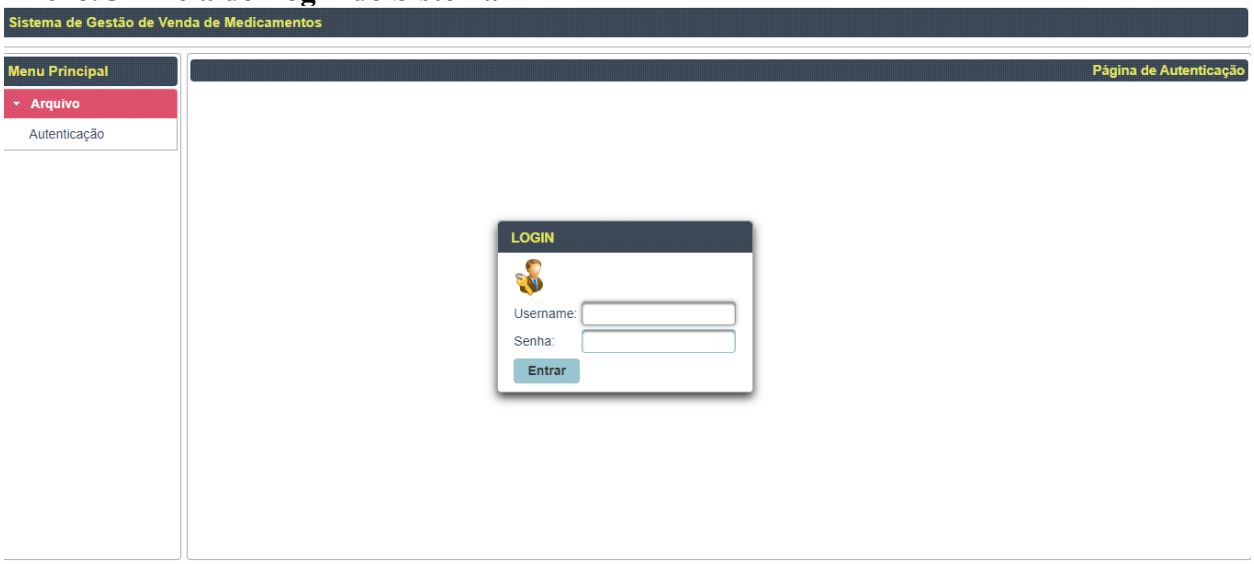

Figura 13 - Tela de Login do Sistema (Fonte: Autor, 2019).

Anexo.D – Tela de Cadastro do Fornecedor

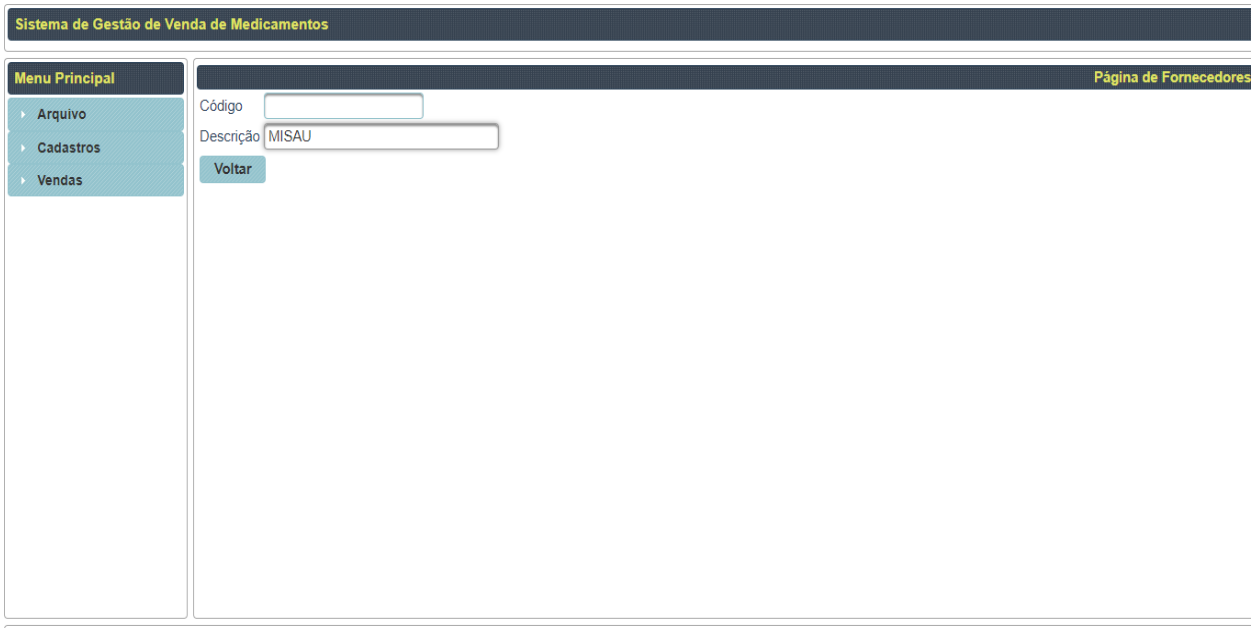

Figura 14 - Tela de Cadastro do Fornecedor (Fonte: Autor, 2019).

Anexo.E – Tela de Cadastro do Funcionário

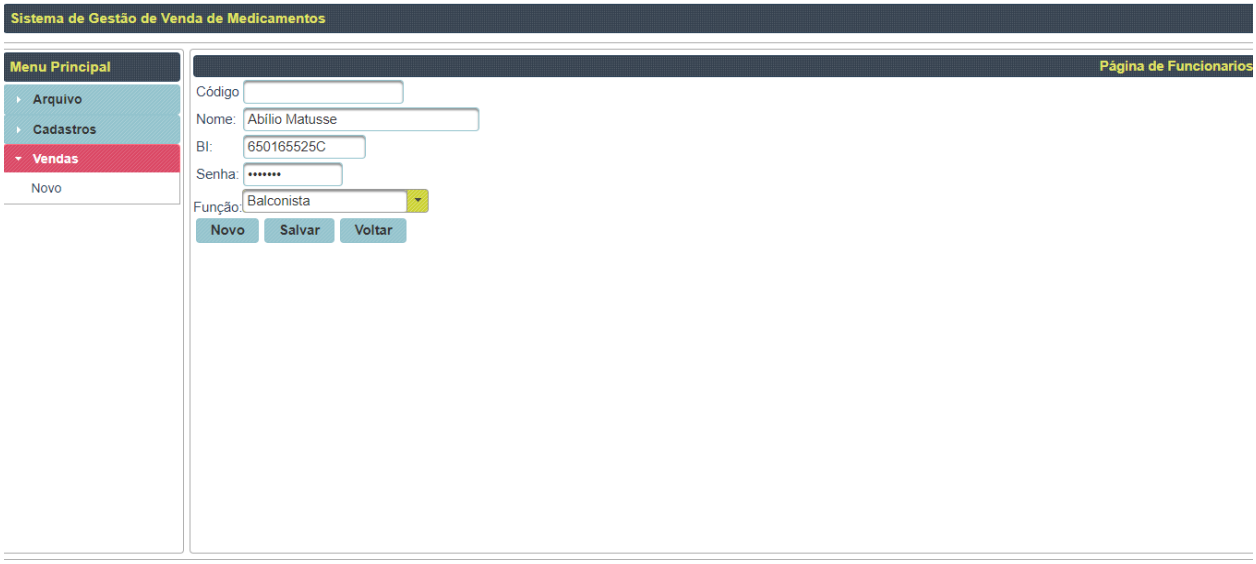

Figura 15 - Tela de Cadastro do Funcionário (Fonte: Autor, 2019).

Anexo.F – Tela de Finalização da Venda

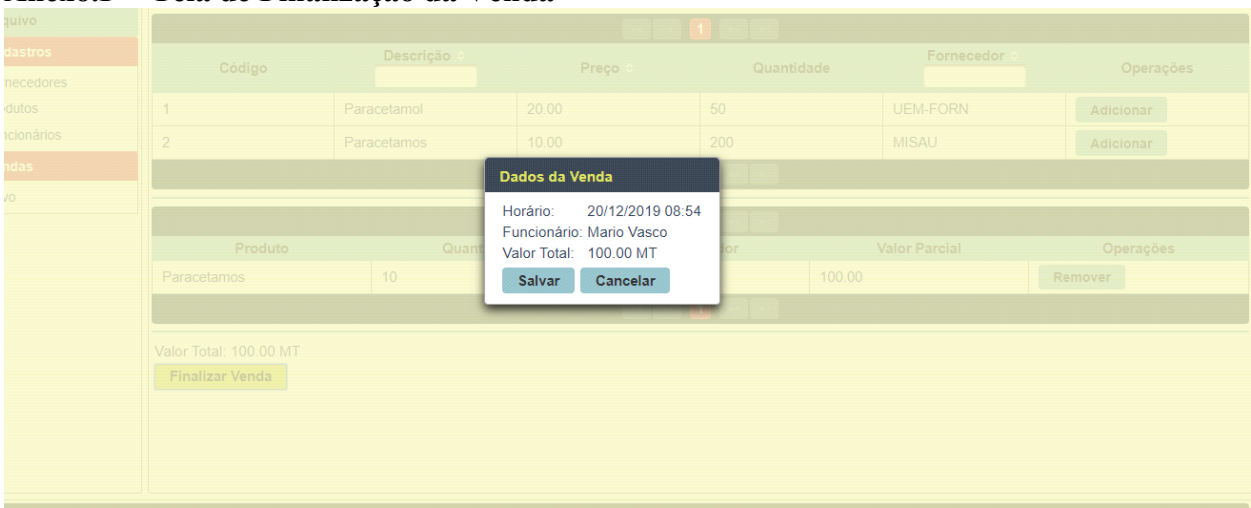

Figura 16 - Tela de Finalização da Venda (Fonte: Autor, 2019).