UNIVERSIDADE FEDERAL RURAL DO SEMI-ÁRIDO
PRÓ-REITORIA DE GRADUAÇÃO
CENTRO DE CIÊNCIAS EXATAS E NATURAIS
CURSO DE CIÊNCIAS DA COMPUTAÇÃO

JEFFERSON CARLOS DA COSTA LINS

ANÁLISE DE PARADIGMAS DE BANCOS DE DADOS A PARTIR DO DESENVOLVIMENTO DE UM SISTEMA PARA ASSISTÊNCIA DE ENFERMAGEM NO RASTREAMENTO DE CÂNCER GINECOLÓGICO

MOSSORÓ
2016

JEFFERSON CARLOS DA COSTA LINS

ANÁLISE DE PARADIGMAS DE BANCOS DE DADOS A PARTIR DO DESENVOLVIMENTO DE UM SISTEMA PARA ASSISTÊNCIA DE ENFERMAGEM NO RASTREAMENTO DE CÂNCER GINECOLÓGICO

Monografia apresentada a Universidade Federal Rural do Semi-Árido como requisito para obtenção do título de Bacharel em Ciências da Computação.

Orientadora: Angélica Félix de Castro, Profª. Drª.

MOSSORÓ

2016

L759a Lins, Jefferson Carlos da Costa.
ANÁLISE DE PARADIGMAS DE BANCOS DE DADOS A PARTIR DO DESENVOLVIMENTO DE UM SISTEMA PARA ASSISTÊNCIA DE ENFERMAGEM NO RASTREAMENTO DE CÂNCER GINECOLÓGICO / Jefferson Carlos da Costa Lins. - 2016.
91 f. : il.

Orientadora: Angélica Félix de Castro.
Monografia (graduação) - Universidade Federal Rural do Semi-árido, Curso de Ciência da Computação, 2016.

1. Engenharia de Software. 2. Banco de Dados. 3. SAE. I. Castro, Angélica Félix de , orient. II. Título.

JEFFERSON CARLOS DA COSTA LINS

ANÁLISE DE PARADIGMAS DE BANCOS DE DADOS A PARTIR DO DESENVOLVIMENTO DE UM SISTEMA PARA ASSISTÊNCIA DE ENFERMAGEM NO RASTREAMENTO DE CÂNCER GINECOLÓGICO

Monografia apresentada a Universidade Federal Rural do Semi-Árido como requisito para obtenção do título de Bacharel em Ciências da Computação.

Defendida em: 04 / 11 / 2016.

BANCA EXAMINADORA

Angélica Félix de Castro, Profª. Drª. (UFERSA)
Presidente

Paulo Gabriel Gadelha Queiroz, Prof. Dr. (UFERSA)
Membro Examinador

Sairo Raoní dos Santos, Prof. Msc. (UFERSA)
Membro Examinador

AGRADECIMENTOS

Agradeço, primeiramente, a Deus, por ter me dado a oportunidade de adquirir tanto conhecimento ao longo dos últimos anos. Por ter me guiado pelos caminhos de sabedoria nos momentos difíceis e permitir que eu realizasse este sonho.

Agradeço infinitamente aos meus pais Deise e Jean Lins. Em especial, a minha mãe, por ter me apoiado sempre, por acreditar que eu seria capaz de conquistar esta vitória, dando suporte na realização deste sonho e sendo a minha base e o meu alicerce. Esta vitória é nossa!

Agradeço à minha família por toda a torcida e apoio. À minha irmã Emanuela Lins. Aos meus avós maternos Francisca e Francisco da Costa. Aos meus avós paternos Margarida e José Lins. Às minhas tias Edda Costa, Léa Leandro, Maria da Conceição e Meiriane Lins. Aos meus tios, Fábio e Fabiano Lins, Flávio Emanuel, Francisco Galberto e José Júnior. Aos primos e primas, em especial, Karen Aquino, João Victor e Weslley Henrique. Obrigado por sempre me darem o maior incentivo e suporte exatamente no momento certo, por terem depositado confiança em mim e compreenderem a importância e o tamanho do meu sonho.

Aos melhores amigos que fiz dentro da universidade. A Andrezza Kharla, por ser a grande incentivadora do grupo, mostrando a importância da oportunidade que nos foi concedida. A Maria Luzia, por ser a melhor parceira de trabalhos em dupla e organizar as melhores confraternizações da turma. A Queiróz Neto, por ser o melhor monitor, pela prestatividade e a boa vontade de nos ajudar quando precisávamos. Vocês são o meu melhor trio e juntos somos o melhor quarteto que este curso já teve.

Aos amigos Augusto Sávio e Glacilene Pires, por me darem tamanha inspiração e força de vontade durante estes anos de graduação.

A Ismael Vinícius por ter me concedido a direção inicial para a realização desta pesquisa, pela paciência e atenção durante as reuniões e, também, via *chat.* Sua ajuda foi de grande importância para este projeto. Muito obrigado por tudo!

À Associação Tecnológica Universitária Potiguar (#SETUP) e aos que fizeram parte dela comigo. Em especial, a Julyana Lima e Gleice Medeiros, que além de parceiras de trabalho, tornaram-se amigas muito queridas. Obrigado pela confiança e pela grande parceria que construímos durante os anos de trabalho na Empresa Júnior.

Aos professores, por toda a paciência e conhecimento compartilhado. Agradeço, especialmente, a Angélica Félix por me orientar na execução deste trabalho com tanta maestria, sabedoria e competência. Muito obrigado por todos os ensinamentos. Aos professores da banca examinadora, Paulo Gabriel e Sairo Santos, pela oportunidade de

fornecer ainda mais conhecimento para a minha formação e por terem aceitado avaliar este trabalho.

A Caio Cavalcante, por ser o meu refúgio e ser sempre paciente, ouvir os meus desabafos nos momentos complicados e ainda opinar a respeito, mostrando-me as melhores decisões a serem tomadas.

“Eu vejo um novo começo de era. De gente fina, elegante e sincera, com habilidade para dizer mais sim do que não [...]”.

Lulu Santos

RESUMO

A Clínica Escola da Universidade Potiguar (UnP) – Campus Mossoró (RN) – recebeu em 2016 uma Sistematização da Assistência de Enfermagem (SAE) que auxilia no rastreamento de câncer ginecológico. No entanto, esta metodologia não foi informatizada, ou seja, todos os registros dos processos são realizados no papel e armazenados em pastas, comprometendo quesitos, tais como: a segurança, a confidencialidade, a privacidade, a integridade, dentre outros. Com base nos paradigmas da Engenharia de *Software* e Bancos de Dados, esta monografia apresenta o processo de desenvolvimento de uma ferramenta computacional que engloba todos os procedimentos realizados na nova metodologia utilizada na Clínica Escola, e ainda ajuda a definir qual base de dados será empregada, fazendo um estudo comparativo entre o Banco de Dados Relacional *MariaDB* e o Banco de Dados Orientado a Objetos *DB4O*.

Palavras-chave: Engenharia de *Software*, Banco de Dados, SAE,

ABSTRACT

The Clínica Escola da Universidade Potiguar (UnP) – Campus Mossoró (RN) - received in 2016 a Sistematização de Assistência de Enfermagem (SAE) that aids in tracking gynecologic cancer. However, this methodology was not computerized, meaning all records of the proceedings are did on paper and stored in folders, compromising questions, such as: security, confidentiality, privacy, integrity, among others. Based on paradigms of Software Engineering and Database, this work presents the development process of a computational tool that encompasses all procedures performed in the new methodology used in the Clínica Escola, and also helps to define which database will be used, making a comparative study of the Relational Database *MariaDB* and Object Oriented Database *DB4O*.

Keywords: Software Engineering, Database, SAE.

LISTA DE FIGURAS

LISTA DE GRÁFICOS

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

Banco de Dados Orientado a Objetos (BDOO)

Banco de Dados Relacional (BDR)

Computer-Aided Software Engineering (CASE)

Create, Retrieve, Update and Delete (CRUD)

Data Definition Language (DDL)

Data Manipulation Language (DML)

Database for Objects (DB4O)

Diagrama de Classes (DC)

Diagrama de Entidade e Relacionamento (DER)

eXtensible Markup Language (XML)

Extreme Programming (XP)

HyperText Markup Language (HTML)

Hypertext Transfer Protocol (HTTP)

Java Community Process (JCP)

Java Persistence Query Language (JPQL)

Java Pesistence API (JPA)

Java Server Faces (JSF)

Modelo de Entidade e Relacionamento (MER)

Model-view-controller (MVC)

Object Identifier (OID)

Object-Relational Mapping (ORM)

Processo de Enfermagem (PE)

Registro Geral (RG)

Sistema de Assistência de Enfermagem – Rastreamento de Câncer Ginecológico (SAE-GIN)

Sistema de Gerência de Banco de Dados (SGBD)

Structured Query Language (SQL)

Unified Modeling Language (UML)

SUMÁRIO

1 INTRODUÇÃO

O grande avanço da Tecnologia da Informação (TI) possibilitou uma série de melhorias para a população em diversas áreas de trabalho. Atualmente, é possível perceber que grande parte das empresas e instituições existentes no mundo utiliza algum Sistema de Informação (SI) para realizar as tarefas corriqueiras que antes só eram feitas por meio do uso do papel. Estes sistemas tornaram-se indispensáveis para o cotidiano dessas empresas e acarretaram benefícios no âmbito da praticidade, agilidade e facilidade na execução das tarefas.

Mesmo com o avanço da TI e da constatação de que um sistema de informação é indispensável para o aprimoramento das tarefas realizadas nas empresas, algumas organizações não passaram a utilizar ferramentas computacionais, pois não acreditam nas vantagens que os sistemas podem proporcionar, consideram o serviço desnecessário, optam por não investir financeiramente na implantação e manutenção do *software* e, até mesmo, desconhecem tal mecanismo.

A Clínica Escola da Universidade Potiguar (UnP) – Campus Mossoró (RN) – recebeu em 2016 uma metodologia de assistência de enfermagem que auxilia no rastreamento de câncer ginecológico. Esta ferramenta visa a uma assistência bem elaborada e eficiente que acelera e qualifica o processo de trabalho da equipe de enfermagem, e ainda possibilita uma assistência completa do serviço de saúde, facilitando o trabalho da equipe e possibilitando a geração de expectativas positivas no decorrer das consultas ginecológicas.

Esta metodologia foi criada a partir da pesquisa de dois alunos do Curso de Enfermagem da universidade, porém não foi informatizada, ou seja, todos os registros são realizados no papel e armazenados em pastas. Logo, nota-se que o método de arquivamento usado é completamente vulnerável, pois compromete vários quesitos, tais como a segurança, a confidencialidade, a privacidade, a integridade, dentre outros.

Durante o desenvolvimento de um *software* bem estruturado, é necessária a utilização de padrões de projeto da Engenharia de *Software*. Esses padrões possuem etapas que, se bem realizadas, poderão facilitar todo o processo de desenvolvimento da aplicação, e, possivelmente, um excelente resultado será obtido. Para Pressman (2006), é possível, a partir da ideia inicial, evoluir até um produto concreto que deve oferecer aos usuários as funções necessárias para realizar suas tarefas. Porém, chegar a um produto de *software* a partir de uma ideia inicial não é uma atividade simples.

Podem ser encontradas, na literatura, propostas de processos e modelos de desenvolvimento de sistemas que são cada vez mais estudados e utilizados para garantir a qualidade do *software*, minimizar as dificuldades durante o seu desenvolvimento e reduzir o tempo necessário para tal (PRESSMAN, 2006; SOMMERVILLE, 2007).

Dentre os modelos de processos encontrados na literatura, o modelo cascata apresenta as características necessárias ao desenvolvimento do sistema de informação proposto para os procedimentos da Clínica Escola. Nesse modelo, todos os requisitos são identificados na fase inicial do projeto; em seguida, os requisitos encontrados são projetados; depois, acontece a execução do que foi projetado, ou seja, a implementação computacional dos requisitos; posteriormente, ocorre a fase de testes, a fim de verificar se o que foi implementado está de acordo com o requisitado; por fim, ocorre a implantação do sistema no ambiente do cliente que irá usar o *software*.

Durante o processo de desenvolvimento de um sistema de informação, é importante definir onde os dados serão armazenados, ou seja, qual Banco de Dados (BD) será utilizado. Segundo Heuser (2009), o conjunto de arquivos integrados que atendem a um conjunto de sistemas recebe o nome de Banco de Dados. É importante, também, definir o paradigma de Banco de Dados que será empregado. Os paradigmas de Bancos de Dados selecionados para essa pesquisa foram: Bancos de Dados Relacionais (BDRs) e Bancos de Dados Orientados a Objetos (BDOOs).

Os BDRs utilizam uma coleção de tabelas, todas com nomes únicos, que compõem a base de dados; essas tabelas podem estar relacionadas a uma ou mais tabelas. Pode-se observar que conceitos como integridade referencial de dados e chaves primárias estão presentes e garantem que um conjunto de informações seja representado de maneira consistente, independentemente da forma de acesso. Já os BDOOs apresentam maior flexibilidade na manipulação de seu conteúdo e, por meio de identificadores de objetos, manipulam os dados de forma consistente (BENEDICTO; BEZERRA; BOSCARIOLI; DELMIRO, 2006).

Com base nos paradigmas da Engenharia de *Software* e Bancos de Dados, este trabalho apresenta o processo de desenvolvimento de uma ferramenta computacional que engloba todos os procedimentos realizados na nova metodologia utilizada na Clínica Escola. Este desenvolvimento foi realizado a partir de algumas técnicas da Engenharia de *Software* que são explicadas nos próximos tópicos.

1.1 Justificativa

A escolha do tema se deu pela necessidade de ser desenvolvido um sistema de informação *web* que irá auxiliar os enfermeiros e estudantes da Clínica Escola da UnP a realizarem os procedimentos de uma consulta para detecção de câncer ginecológico.

Atualmente, os enfermeiros e estudantes da Clínica Escola realizam todos os procedimentos das consultas por meio de formulários em papel, que são preenchidos a lápis ou caneta; em seguida, esses formulários são armazenados em pastas. Diante dessa situação, a informatização dos procedimentos é importante, pois manterá os dados das consultas armazenados em segurança, obedecendo a alguns princípios como integridade, confidencialidade, disponibilidade, dentre outros.

1.2 Objetivos

O objetivo deste trabalho é descrever o processo de desenvolvimento de um sistema *web* que dá suporte aos enfermeiros e estudantes da Clínica Escola da UnP no método de diagnóstico de câncer ginecológico, visto que, atualmente, essa técnica é executada de forma retrógrada, manual e pouco dinâmica. Também é objetivo desta pesquisa verificar qual o paradigma de Banco de Dados a ser empregado para o armazenamento dos dados do sistema.

1.2.1 Objetivos Gerais

São objetivos gerais:

• Modelar, desenvolver, testar e validar um sistema *web* para auxílio aos enfermeiros e estudantes da Clínica Escola da UnP; e ainda, estudar, modelar e implementar o sistema utilizando um Banco de Dados Orientado a Objetos, a fim de compará-lo com o Banco de Dados Relacional.

1.2.2 Objetivos Específicos

O desenvolvimento deste trabalho considera, ainda, os seguintes objetivos:

- Conhecer o ambiente para o qual o sistema será desenvolvido;
- Realizar a modelagem do sistema proposto;

- Desenvolver um sistema *web* de acordo com as necessidades dos clientes, realizando, primeiramente, o armazenamento dos dados no banco *MariaDB* e, posteriormente, no banco *DB4O*;
- Realizar um comparativo entre o desempenho dos dois bancos a partir das transações realizadas.

1.3 Metodologia

A proposta deste trabalho consiste em utilizar a metodologia de pesquisa-ação, juntamente com métodos e técnicas da Engenharia de Software para o desenvolvimento de um sistema que automatize os processos realizados pelos enfermeiros na Clínica Escola da UnP. A pesquisa-ação consiste essencialmente em acoplar pesquisa e ação em um único processo, no qual os atores implicados participam, junto com os pesquisadores, para chegarem interativamente a elucidar a realidade em que estão inseridos, identificando problemas coletivos, buscando e experimentando soluções em situação real. Simultaneamente, há produção e uso de conhecimento (THIOLLENT, 1997).

1.4 Estrutura do trabalho

No segundo capítulo deste trabalho, será apresentado o Referencial Teórico, em que são explanadas abordagens importantes sobre o processo de desenvolvimento de um sistema, as metodologias de desenvolvimento, características de Bancos de Dados Relacionais e Bancos de Dados Orientados a Objetos, além de alguns aspectos sobre a Sistematização da Assistência de Enfermagem (SAE).

No terceiro capítulo, explana-se o Desenvolvimento do Sistema, descrevendo as suas fases de criação.

No quarto capítulo, realiza-se um comparativo entre os bancos usados nos dois protótipos do sistema (Banco de Dados Relacional e Banco de Dados Orientado a Objetos).

Por fim, no quinto capítulo, são apresentadas as conclusões do trabalho**.**

2 REFERENCIAL TEÓRICO

Neste capítulo, são abordados os conceitos sobre os processos e as metodologias de desenvolvimento de *software*, os paradigmas dos Bancos de Dados Relacionais e dos Bancos de Dados Orientados a Objetos e, ainda, algumas características da Sistematização da Assistência de Enfermagem (SAE).

2.1 Processos de desenvolvimento de softwares

Um processo de desenvolvimento de *software* pode ser definido como um conjunto de atividades e resultados associados que levam à produção de um produto de *software* (SOMMERVILLE, 2007).

De acordo com Souza Neto (2004), o Processo de Desenvolvimento de *Software*, assim como qualquer outro, tem um compromisso com a qualidade, um enfoque que se desenvolveu com o passar do tempo e deu origem a métodos e abordagens de Engenharia de *Software* cada vez mais amadurecidos, gerando produtos de melhor qualidade e menores custos.

Para PMBOK (2008), as atividades básicas de todo e qualquer processo são: entender as necessidades do cliente; planejar a solução; implementar a solução; validar esta solução e entregar o produto ao cliente. Já para Sommerville (2007), usando termos da Engenharia de *Software*, as atividades fundamentais e comuns a todos os processos de *software* são: especificação de *software*; projeto e implementação de *software*; validação de *software* e, posteriormente, evolução de *software*. Traçando-se um comparativo entre os dois autores e suas considerações, é possível observar que existem diversas formas de realizar um processo de desenvolvimento de *software*, mas é bastante comum a presença de etapas semelhantes em processos distintos.

Sommerville (2007) caracteriza as etapas do processo da seguinte forma:

1. Especificação: é necessário definir a funcionalidade do *software* e as restrições em sua operação;
2. Projeto e implementação: o *software* deve ser produzido de modo a cumprir a sua especificação;
3. Validação: o *software* precisa ser validado para garantir a realização do que o cliente deseja;

4. Evolução: o *software* precisa evoluir para atender às necessidades mutáveis do cliente.

O processo de *software* é representado através de um modelo que tem como objetivo definir os seguintes quesitos: o ciclo de vida do desenvolvimento, os papéis dos desenvolvedores, os resultados esperados (saídas) e o que é necessário para obter esses resultados (entradas). Conforme afirma Sommerville (2007), essas entradas e saídas são conhecidas como artefatos, que podem ser diagramas, códigos fontes, lista de requisitos ou documentos.

2.2 Metodologias de desenvolvimento de softwares

O conjunto de atividades recomendadas para o desenvolvimento de *software* denomina-se metodologia de desenvolvimento. Existem diversas metodologias que se enquadram em modelos tradicionais (pesados) e modelos ágeis (leves).

2.2.1 Metodologias Tradicionais de Desenvolvimento de Software

O modelo tradicional tem uma forte influência e importância para a história da Engenharia de *software*, pois ainda é bastante utilizado nas grandes empresas. Segundo Souza Neto (2004), as metodologias tradicionais, por vezes chamadas de “pesadas”, têm como característica principal a divisão em etapas e/ou fases. Essas fases são bem definidas e englobam atividades como Análise, Modelagem, Desenvolvimento e Testes.

Souza Neto (2004) ainda afirma que as metodologias tradicionais focam na previsibilidade dos requisitos do sistema, o que traz a vantagem de tornar os projetos completamente planejados, facilitando a sua gerência e caracterizando o processo como bastante rigoroso.

Nos modelos tradicionais, ao final de cada fase, um marco é gerado. Dependendo do final de que etapa foram criados, esses marcos podem ser documentos (como diagramas), algum protótipo do *software* ou até mesmo uma versão do sistema.

Os próximos tópicos mostram os principais modelos tradicionais usados por empresas de desenvolvimento.

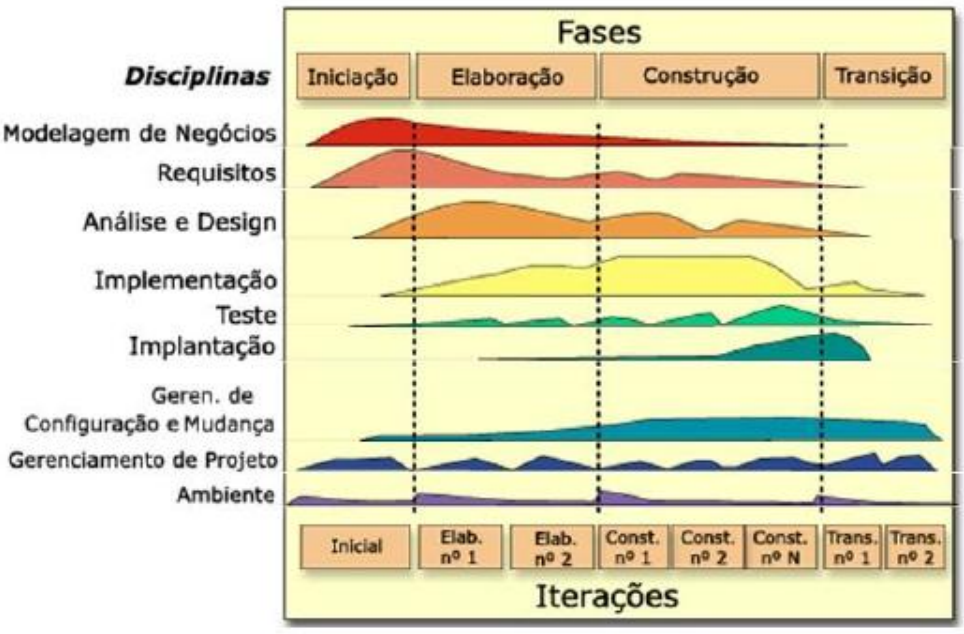
2.2.1.1. Rational Unified Process (RUP)

O *Rational Unified Process* (RUP é um Processo de Engenharia de *Software* criado pela *Rational Software Corporation* e oriundo do Processo Unificado (UP). O RUP descreve como desenvolver um *software* usando técnicas comerciais, com o objetivo de aumentar a qualidade dos *software*s gerados pela empresa desenvolvedora. O modelo visa a auxiliar no controle do planejamento da qualidade, verificando-a na construção de todo o processo e envolvendo todos os membros da equipe de desenvolvimento.

Esse modelo, além de oferecer uma abordagem baseada em disciplinas para atribuir tarefas e responsabilidades dentro de uma organização de desenvolvimento, é também guiado por casos de uso, projetado e documentado através da notação *Unified Modeling Language* (UML) para ilustrar os processos em ação, utilizando técnicas e práticas aprovadas comercialmente (JACOBSON, et al., 1999).

As fases desse modelo são representadas na Figura 1.

Figura 1 – As fases do RUP e a distribuição do volume de atividades em cada uma delas.



Fonte: Ferreira (2013).

Segundo Ferreira (2013 apud RATIONAL, 2001), as definições das fases do RUP são:

- Iniciação: a meta dominante da fase de iniciação é atingir um consenso entre todos os envolvidos nos objetivos do ciclo de vida do projeto. A fase de iniciação tem muita importância principalmente para os novos esforços de desenvolvimentos, pois possuem

maiores riscos de negócios e diversos requisitos que precisam ser preenchidos para que o projeto possa prosseguir.

- Elaboração: a meta da fase de elaboração é criar a *baseline* (linha de base) para a arquitetura do sistema, a fim de fornecer uma base estável para o esforço da fase de construção.
- Construção: a meta desta fase é esclarecer os requisitos restantes e concluir o desenvolvimento do sistema com base na arquitetura da *baseline* (linha de base).
- Transição: o foco desta fase é assegurar que o *software* esteja disponível para seus usuários finais. Esta fase pode atravessar várias interações e inclui testar o produto em preparação para *release* e ajustes pequenos com base no *feedback* do usuário.

2.2.1.2 Modelo Sequencial Linear

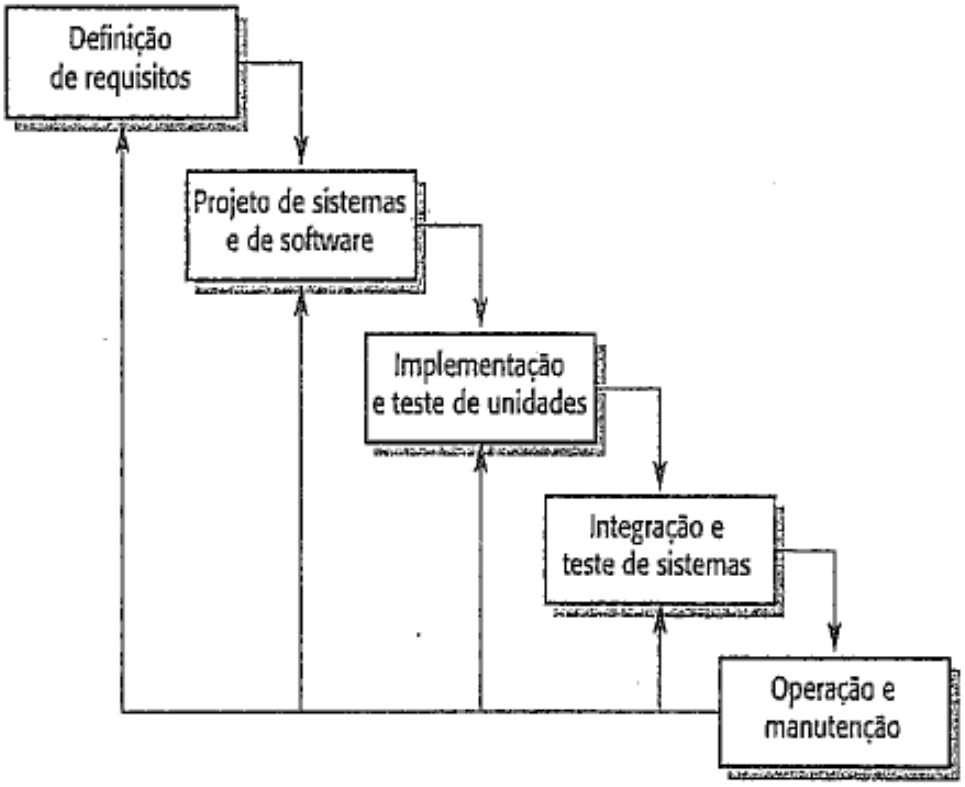
O Modelo Sequencial Linear, proposto em 1970, é também conhecido como Modelo em Cascata, no qual as fases definidas são sistematicamente seguidas de maneira linear (PRESSMAN, 2006).

As fases do Modelo em Cascata são definidas abaixo, segundo Sommerville (2007).

- Análise e definição de requisitos: consultam-se os usuários do sistema a fim de estabelecer funções, objetivos e restrições que, em seguida, são definidos em detalhes e servirão como especificação da aplicação.
- Projeto de sistemas e de *software*: o projeto de sistemas agrupa os requisitos de hardware ou de *software* e estabelece uma arquitetura da aplicação geral. O projeto de *software* identifica e descreve as abstrações fundamentais do sistema a ser desenvolvido e suas relações.
- Implementação e teste de unidades: neste estágio, o projeto de *software* pode ser compreendido como um conjunto de programas ou unidades de programa, enquanto o teste de unidades envolve verificar se cada unidade está atendendo à sua especificação.
- Integração e teste de sistemas: as unidades de programa são integradas e testadas como um sistema completo, com o intuito de verificar se os requisitos de *software* foram atendidos. Após esta fase, o sistema de *software* é entregue ao cliente.
- Operação e manutenção: geralmente, esta é a fase mais longa do ciclo de vida. O sistema é instalado e colocado em operação. Caso apareçam erros que não foram descobertos em estágios anteriores, a manutenção os corrige e, à medida que se criam novos requisitos, as funções do sistema são aumentadas.

A Figura 2 representa as fases explicadas anteriormente.

Figura 2 – O ciclo de vida do *software*.



Fonte: Sommerville (2007).

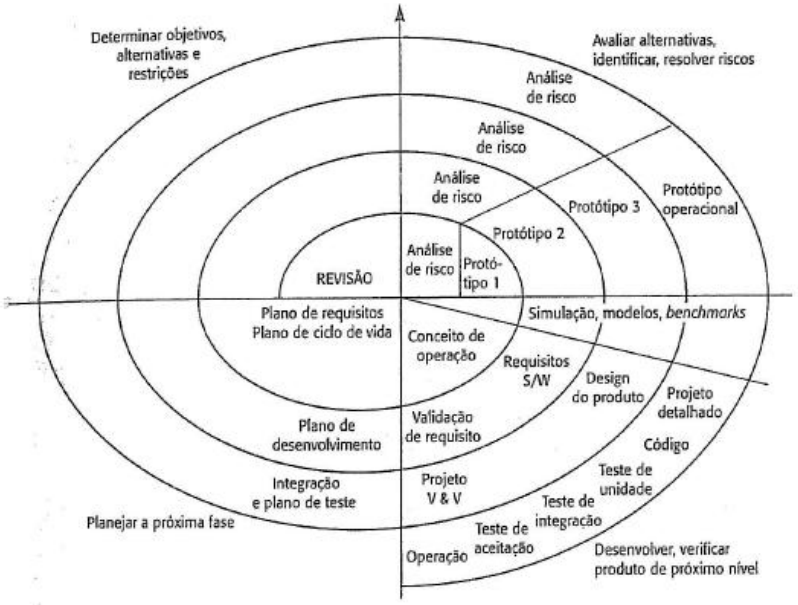
Pressman (2006) afirma que este é o modelo mais usado em todo o mercado, porém não é o mais eficaz, pois raros projetos seguem um fluxo linear. Além disso, as mudanças de requisitos que ocorrem no decorrer do projeto não são de fácil adaptação, pois alteram toda a documentação já desenvolvida, o que implica em retrabalho.

Apesar de ser tradicionalmente reconhecido pela segurança no processo de desenvolvimento e ser um dos modelos mais utilizados, o Modelo em Cascata apresenta algumas desvantagens e restrições, tais como: impede a atualização ou redefinição das fases anteriores; sem suporte para modificações nos requisitos; imprevidência de riscos; impede a reutilização de componentes; suas fases devem estar sempre sincronizadas, pois o atraso em uma delas afeta todo o processo, e ocorre demora na entrega do sistema.

2.2.1.3 Modelo Espiral

O Modelo Espiral foi proposto nos anos 1980 e, ao contrário do Modelo em Cascata, não representa o processo de desenvolvimento de *software* como uma sequência de atividades com algum retorno de uma atividade para outra. Esta metodologia, como sugere seu nome, é representado como uma espiral. Conforme afirma Sommerville (2007), uma fase é representada por uma volta na espiral, e essa é dividida em quatro setores, como mostra a Figura 3.

Figura 3 – O modelo espiral.



Fonte: Sommerville (2007).

Os quatro setores de uma volta da espiral, apresentados na Figura 3, são definidos a seguir, de acordo com Sommerville (2007):

- Definição de objetivos: nesta etapa, definem-se os objetivos específicos; identificam-se as restrições para o processo e o produto, e, prepara-se detalhadamente o plano de gerenciamento. Se forem identificados riscos no projeto, poderão ser planejadas estratégias alternativas.
- Avaliação e redução de riscos: é realizada uma análise detalhada para cada um dos riscos de projeto identificados; são tomadas as providências para reduzir esses riscos.

- Desenvolvimento e validação: após a avaliação dos riscos, para o melhor desenvolvimento do sistema, escolhe-se um modelo de desenvolvimento de acordo com os riscos encontrados.
- Planejamento: nesse estágio, decide-se se é válido continuar com a próxima volta da espiral. Faz-se uma revisão do projeto, e se a decisão for continuar, são traçados os planos para a próxima fase.

Sommerville (2007) ainda destaca que este modelo é considerado o mais realístico possível, pois reconhece que usuários, analistas e desenvolvedores adquirem maior conhecimento sobre o projeto com o decorrer do tempo.

Souza Neto (2004) aponta que uma forte característica desse modelo, que o difere de outros e o torna bastante utilizado pelas empresas, é o fato de acompanhar toda a vida do *software*, mesmo depois da entrega ao cliente, pois é na manutenção que muitas empresas desenvolvedoras têm seu principal foco de atuação.

2.2.2 Metodologias Ágeis

Na tentativa de minimizar as dificuldades encontradas durante a criação de um *software*, os modelos de desenvolvimento de sistemas de informação passaram a ser cada vez mais estudados, havendo a necessidade de se construírem *software*s com mais qualidade em menor tempo (PRESSMAN, 2006).

Diante dessa problemática, no ano de 2001, foi publicado o Manifesto Ágil (MANIFESTO, 2001), assinado por produtores, desenvolvedores e consultores de *software* que integram a Aliança Ágil, no intuito de combinar uma filosofia e um conjunto de diretrizes de desenvolvimento. O estilo ágil de desenvolvimento aborda diretamente os problemas de mudanças rápidas (COCKBURN e HIGHSMITH, 2001).

O Manifesto Ágil destacava quatro valores, que são listados a seguir:

- Indivíduos e iterações ao invés de processos e ferramentas;
- *Software* funcional ao invés de documentação detalhada;
- Colaboração do cliente ao invés de negociação de contratos;
- Responder às mudanças ao invés de seguir um plano.

De acordo com Pressman (2006), as metodologias ágeis têm conquistado seu espaço na indústria de *software*s. Esse modelo foi desenvolvido para vencer as fraquezas percebidas e reais da engenharia de *software* convencional. Silva (2005) destaca que uma das características positivas desse método é o fato de não ser centrado nos artefatos, optando por

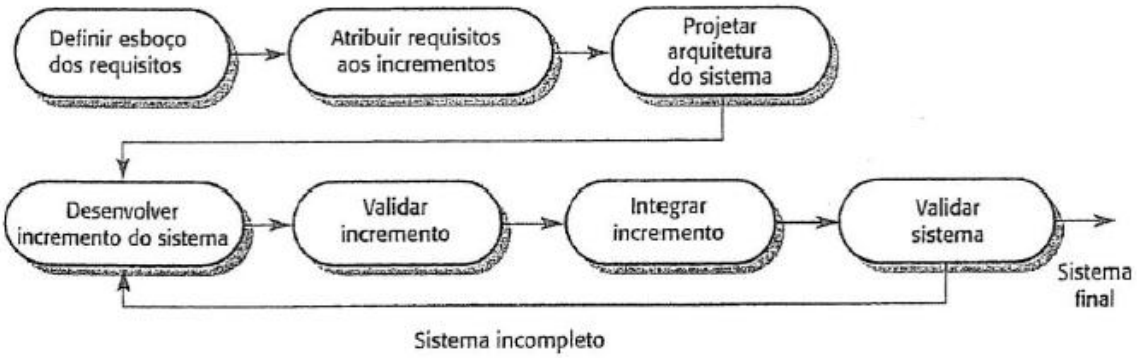
uma documentação apropriada para evitar redundâncias e excessos, e auxiliando efetivamente o desenvolvimento do *software*.

Os próximos tópicos mostram os modelos ágeis usados com maior frequência no mercado.

2.2.2.1 Iterativo e Incremental

Os modelos iterativos apresentam o processo de *software* como um ciclo de atividades. A iteratividade apoia as partes do processo que são repetidas à medida que os requisitos do sistema evoluem. Esse modelo foi desenvolvido como uma tentativa de reduzir o "retrabalho" no processo de desenvolvimento e de ceder aos clientes algumas oportunidades de adiar decisões sobre seus requisitos detalhados, caso alguns requisitos não sejam necessários (SOMMERVILLE, 2007). A Figura 4 mostra a sequência das etapas do modelo de desenvolvimento iterativo e Incremental.

Figura 4 – Desenvolvimento Iterativo e Incremental.



Fonte: Sommerville (2007).

O desenvolvimento iterativo e incremental torna o projeto mais prático, pois ocorre uma divisão do trabalho em pedaços menores ou pequenos projetos. Cada pequeno projeto é uma iteração que resulta em um incremento. Iterações são passos em um fluxo de trabalho e incrementos são crescimentos do produto, ou seja, a inserção de novas funiconalidades.

Esse modelo proporciona diversos benefícios. Segundo Ferreira (2013 apud MARTINS, 1999), são eles:

- Redução dos riscos envolvendo custos a um único incremento;

- Redução do risco de lançar o projeto no mercado fora da data planejada;
- Redução do tempo de desenvolvimento do projeto como um todo;
- Reconhecimento de uma realidade frequentemente ignorada: as necessidades dos usuários e os requisitos correspondentes não podem ser totalmente definidos no início do processo.

2.2.2.2 Scrum

O modelo *Scrum* é um processo de desenvolvimento de *software* iterativo, incremental e adaptativo. Seu principal objetivo é entregar a maior quantidade de requisitos de *software* com a melhor qualidade possível, e por meio de uma série de intervalos de tempo, fixos e curtos, que geralmente duram cerca de um mês (FIGUEIREDO, 2006).

Esse processo define um ciclo de vida básico, com alguns papéis e práticas que têm por finalidade garantir, além da visibilidade e controle total do projeto, poder e liberdade para a equipe de desenvolvimento, a ponto de permitir que esta equipe defina a tática para atingir a meta. Não há uma descrição de atividades técnicas e processos; o foco do método é o gerenciamento e controle de um projeto de desenvolvimento de *software*, descrevendo apenas as diretrizes básicas a serem adotadas de forma a garantir o sucesso do projeto (FIGUEIREDO, 2006).

As fases do *Scrum*, segundo Bassi Filho (2008), são:

- Planejamento: etapa na qual são conhecidas as características do produto, definindo-se as metas de desenvolvimento, as tecnologias e ferramentas de trabalho, e as tarefas que levarão ao cumprimento das metas;
- Sprint: é a fase de implementação de uma funcionalidade que esteja prevista para aquela iteração;
- Avaliação: logo após o término do *sprint*, é feita uma avaliação pela equipe sobre o trabalho realizado, identificando-se as oportunidades de melhorias de desempenho nos próximos *sprints*.

2.2.2.3 Extreme Programming (XP)

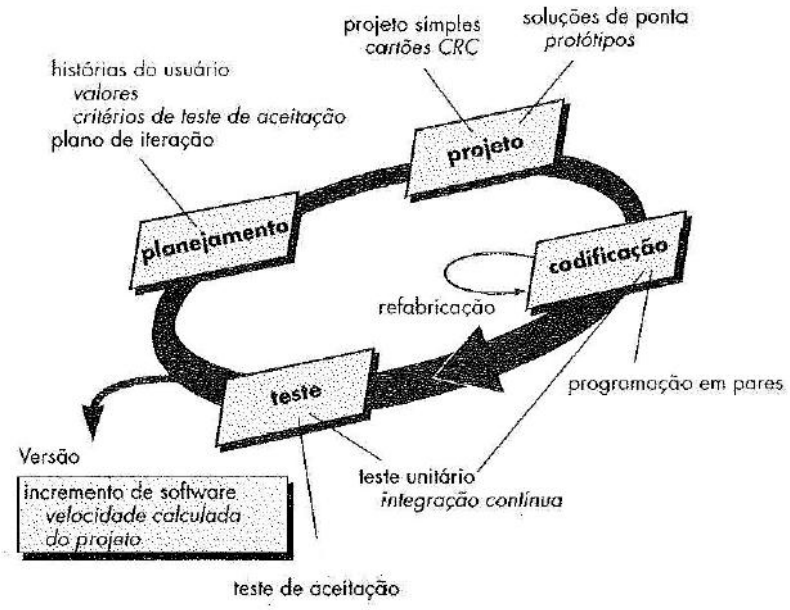
Criado por Kent Beck e Ward Cunningham, o XP era o mais popular dos métodos ágeis (SILVA, 2005). De acordo com Pressman (2006), esse modelo de processo usa uma abordagem orientada a objetos como seu paradigma predileto, incluindo um conjunto de

regras e práticas que ocorrem no contexto de quatro atividades de estrutura: planejamento, projeto, codificação e teste. Além disso, conforme afirma Beck (2000), esse método tem seus princípios baseados em quatro valores que, em alinhamento com o Manifesto Ágil, definiram as prioridades da metodologia. Os quatro valores são: comunicação, simplicidade, *feedback e* coragem.

Segundo Beck (1999), o XP tinha como objetivo inicial ser aplicado em desenvolvimento de sistemas de pequeno a médio porte. Após algum tempo, o XP tornou-se mais inclusivo, abrangente e flexível por meio do enfoque nos valores da metodologia (BECK, 2004). Foi possível, então, tentar reduzir o custo da mudança em ambientes mais desafiadores, ou seja, em sistemas de grande porte.

A Figura 5 apresenta o processo XP e algumas de suas atividades principais.

Figura 5 – O Processo Extreme Programming.



Fonte: Pressman (2006).

As atividades ilustradas na Figura 5 são brevemente descritas a seguir:

- Planejamento: nessa atividade, cria-se um conjunto de histórias, também conhecidas como histórias de usuário, que irão descrever as características e funcionalidades requeridas para o *software* que será construído;
- Projeto: nesta etapa, o projeto é feito, preferivelmente de forma simples, e irá fornecer diretrizes de implementação do *software* de acordo com a história escrita;

- Codificação: após o desenvolvimento das histórias e projeto, a equipe irá desenvolver uma série de testes unitários que exercitarão cada uma das histórias que devem ser incluídas na versão atual do sistema; depois da criação dos testes, o desenvolvedor estará mais bem preparado para focalizar no que precisa ser implementado; e então, após a implementação, o código é submetido ao teste unitário, a fim de fornecer um *feedback* instantâneo para os desenvolvedores. O XP recomenda que duas pessoas trabalhem juntas em um computador para criar o código correspondente a uma história;
- Teste: aqui são executados testes de integração e validação diariamente, a fim de fornecer à equipe XP uma indicação contínua de progresso e também levantar sinais de alerta se os problemas estiverem se deteriorando. Os testes de aceitação, também conhecidos como testes do cliente, são especificados pelo cliente e derivam-se das histórias de usuário.

Após serem relatados os principais pontos da Engenharia de *Software*, faz-se necessária uma abordagem sobre o armazenamento dos dados, isto é, os Bancos de Dados utilizados, com a finalidade de verificar quais vantagens e desvantagens eles podem proporcionar para a aplicação desenvolvida.

2.3 Banco de Dados

Dados são compreendidos como objetos que podem ser armazenados e que possuem um significado implícito. Assim, um Banco de Dados é definido como um conjunto de dados devidamente relacionados. No entanto, o termo “Banco de Dados” pode ser explanado de uma forma mais aprofundada e detalhada.

Segundo Machado (2008), um Banco de Dados possui as seguintes propriedades:

- É uma coleção lógica coerente de dados com um significado inerente; desde modo, uma disposição desordenada dos dados não pode ser referenciada como Banco de Dados;
- Ele é projetado, construído e preenchido com valores de dados para um propósito específico; um Banco de Dados possui um conjunto predefinido de usuários e de aplicações;
- Ele apresenta algum aspecto do mundo real, o qual é chamado de minimundo; qualquer alteração efetuada no minimundo é automaticamente refletida no Banco de Dados.

Heuser (2009) afirma que o conjunto de arquivos integrados que atendem a um conjunto de sistemas recebe o nome de Banco de Dados. Para o autor, o compartilhamento de dados tem reflexos na estrutura do *software*: a estrutura interna dos arquivos passa a ser mais complexa, pois estes devem ser construídos de forma a atender às necessidades dos diferentes

sistemas. Para contornar este problema, usa-se um Sistema de Gerência de Banco de Dados (SGBD), um *software* que incorpora as funções de definição, recuperação e alteração de dados em um Banco de Dados.

Antes de criar um Banco de Bados, é necessário, primeiramente, elaborar seu projeto. De acordo com Oliveira (2013 *apud* Elmasri e Navathe, 2011), o projeto de uma nova aplicação para um Banco de Dados existente ou de um novo Banco de Dados começa com uma fase chamada especificação e análise de requisitos. Esses requisitos são documentados com detalhes e transformados em um "projeto conceitual", que pode ser representado e manipulado por ferramentas computadorizadas, para que seja facilmente mantido, modificado e transformado em uma implementação de Banco de Dados. O projeto é então traduzido para um "projeto lógico", que pode ser expresso em um modelo de dados implementado em um SGBD. O estágio final é o "projeto físico", no qual outras especificações são fornecidas para armazenar e acessar o Banco de Dados.

Assim como na engenharia de *software*, foram criados paradigmas para a criação de Bancos de Dados. Dentre esses paradigmas, podem ser destacados o relacional e o orientado a objetos, que serão explicados nos próximos tópicos.

2.3.1 Bancos de Dados Relacionais (BDRs)

Criados por Edgar F. Codd nos anos 1970, os Bancos de Dados Relacionais começaram a ser utilizados nas empresas a partir de 1987. A abordagem relacional está baseada no princípio de que as informações em uma base de dados podem ser consideradas relações matemáticas e estão representadas de maneira uniforme com o uso de tabelas bidimensionais (MACHADO, 2008).

O modelo relacional representa o Banco de Dados como uma coleção de relações. Informalmente, cada relação é semelhante a uma tabela de valores. Quando uma relação é considerada uma tabela de valores, cada linha na tabela representa uma coleção de dados relacionais; uma linha representa um fato que normalmente corresponde a uma entidade ou relacionamento do mundo real; os nomes da tabela são usados para ajudar a interpretar o significado dos valores em cada linha; os nomes de coluna representam os valores de dados em cada linha, com base na coluna em que cada valor se encontra; e todos os valores em uma coluna são do mesmo tipo de dado (OLIVEIRA, 2013).

De acordo com Elmasri e Navathe (2011), na terminologia formal do modelo relacional, uma linha é chamada de tupla, um cabeçalho da coluna é chamado de atributo, e

uma tabela é chamada de relação. O tipo de dado que descreve os valores de cada coluna é representado por um domínio de valores possíveis. Segundo Machado (2008), a teoria relacional possui premissas que definem uma tabela de dados:

- Cada uma das tabelas é chamada de relação;
- O conjunto de uma linha e suas colunas é chamado de tupla;
- Cada coluna dessa tabela tem um nome e representa um domínio da tabela;
- A ordem das linhas é irrelevante;
- Não há duas linhas iguais;
- Usamos nomes para fazer referência às colunas;
- A ordem das colunas também é irrelevante;
- Cada tabela tem um nome próprio distinto de qualquer outra tabela no Banco de Dados.

Para exemplificar uma tabela de Banco de Dados, a Tabela 1 exibe um esquema representando um cadastro de clientes, no qual são definidos os seguintes atributos: CodCliente, NomeCliente, Sexo, Telefone.

Tabela 1 - Exemplo de uma tabela (relação) cliente.

CodCliente	NomeCliente	Sexo	Telefone
100	João Paulo	Masculino	(84) 8888-8888
101	Francisco José	Masculino	(84) 8877-8877
102	Ana Maria	Feminino	(84) 8866-8866
103	Larissa Silva	Feminino	(84) 8855-8855

Fonte: Autoria Própria.

Tomando como exemplo a primeira linha da tabela, pode-se concluir que existe um cliente de nome João Paulo, de CodCliente 100, do sexo masculino, com telefone (84) 8888-8888. Assim, percebe-se que cada linha de uma tabela representa um objeto, um assunto que é descrito pelos valores de cada uma das colunas.

2.3.1.1 Características gerais dos Bancos de Dados Relacionais

Sobre as características dos Bancos de Dados relacionais, Oliveira (2013) destaca que existem quatro operações básicas que podem ser aplicadas nos estados das relações de um Banco de Dados: inserir, selecionar, excluir e alterar. Elas inserem novos dados, recuperam registros já persistidos, excluem dados antigos ou modificam registros de dados existentes. Estas quatro operações são denominadas na literatura especializada de CRUD, acrônimo originado da expressão em língua Inglesa: *Create, Retrieve, Update and Delete*.

Uma característica importante é que em toda e qualquer tabela de um Banco de Dados relacional, haverá uma coluna ou um conjunto de colunas concatenadas, cujos valores são únicos na tabela, isto é, determinado valor nunca se repete em nenhuma outra linha da tabela. Esta coluna ou conjunto de colunas concatenadas identifica uma única linha da tabela, formando o que se conhece como "chave primária" da tabela (MACHADO, 2008).

Oliveira (2013) afirma que para definir a estrutura de um Banco de Dados, a maioria dos SGBDs emprega uma Linguagem de Definição de Dados (*DDL – Data Definition Language*). Para tanto, o SGBD possui um compilador da DDL, cuja função é processar instruções da DDL a fim de identificar as descrições dos construtores de esquema e armazenar a descrição de esquema no catálogo do SGBD. Após a compilação dos esquemas, por meio da definição da DDL, o Banco de Dados pode ser preenchido e os usuários precisam manipular os dados, então o SGBD oferece um conjunto de operações ou uma linguagem chamada de Linguagem de Manipulação de Dados (*DML – Data Manipulation Language*). A DML oferece as operações típicas: inserção, recuperação, exclusão e modificação de dados.

Nos Bancos de Dados relacionais, existe a SQL (*Structured Query Language*) ou Linguagem de Consulta Estruturada, um exemplo típico de linguagem de Banco de Dados abrangente, que representa uma combinação de DDL e DML. A linguagem SQL é um dos principais motivos para o sucesso dos Bancos de Dados relacionais comerciais, pois, dentre outros benefícios, proporciona a padronização (OLIVEIRA, 2013).

2.3.2 Bancos de Dados Orientados a Objetos (BDOOs)

O paradigma da orientação a objetos possui uma maneira própria de representar um problema. Esta maneira difere muito da forma tradicional de modelagem de dados utilizada pelos Bancos de Dados Relacionais, apesar de guardar algumas semelhanças (BENEDICTO; BEZERRA; BOSCARIOLI; DELMIRO, 2006).

A modelagem de um Banco de Dados Orientado a Objetos está intimamente ligada aos diagramas de classes gerados para o sistema. Para Chaudri & Zicari (2001), uma base de dados orientada a objetos é apenas uma coleção de objetos, enquanto em um sistema orientado a objetos, cada objeto representa uma entidade do mundo real.

Os Bancos de Dados Orientados a Objetos foram propostos para atender a algumas das necessidades de aplicações mais complexas, como a necessidade de novos tipos de dados para armazenar imagens, vídeos ou itens de textos grandes; a carência de transações de maior duração; e a necessidade de definir operações fora do padrão que são específicas da aplicação. Um recurso chave dos BDOOs é o poder que eles concedem ao projetista de especificar tanto a estrutura dos objetos complexos quanto as operações que são aplicadas a esses objetos [OLIVEIRA *apud* ELMASRI; NAVATHE, 2011].

2.3.2.1 Características gerais dos Bancos de Dados Orientados a Objetos

O paradigma dos BDOOs possui características do paradigma da programação orientada a objetos, tais como: objeto, classe, herança, polimorfismo e encapsulamento.

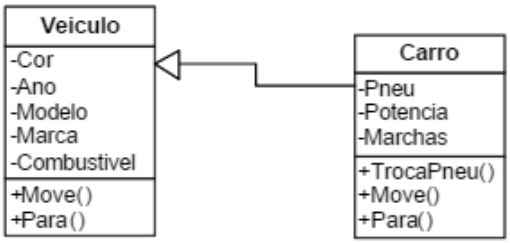
Sobre os objetos, pode-se afirmar que cada um deles possui um estado e comportamentos próprios, sendo diferenciado por um indicador único (OID ou *object identifier*). O estado do objeto depende do valor de cada uma de suas propriedades, e o comportamento é especificado pelas operações que podem ser executadas pelo objeto, possivelmente modificando seu estado. As propriedades podem ser atributos ou relações entre esse objeto e outros objetos que façam parte do domínio do problema (BENEDICTO; BEZERRA; BOSCARIOLI; DELMIRO, 2006).

As classes são agrupamentos de objetos de um mesmo tipo, com comportamentos (operações) e propriedades (atributos e relações) em comum. Os tipos de objetos (classes) podem ser utilizados como domínios para propriedades ou argumentos para operações. Os Bancos de Dados orientados a objetos possibilitam que novos tipos sejam criados para se adequarem aos requerimentos de cada aplicação, podendo ser combinados com os tipos já existentes no sistema e ser utilizados exatamente da mesma maneira (BENEDICTO; BEZERRA; BOSCARIOLI; DELMIRO, 2006).

Outra característica é a herança, que permite que uma classe seja estendida por outra classe. Tomando o exemplo da classe Veículo, pode-se estendê-la, adicionando atributos e comportamentos para atender ao problema, como a classe Carro mostrada na Figura 6. A partir deste conceito básico, supõe-se que uma infinidade de classes pode ser derivada de

Veículo, especializando atributos e comportamentos. Essa característica torna-se bastante útil quando há a necessidade de expressar um problema que envolva diversos tipos com atributos e/ou operações em comum, mas que possuam particularidades inerentes a algumas classes (BENEDICTO; BEZERRA; BOSCARIOLI; DELMIRO, 2006).

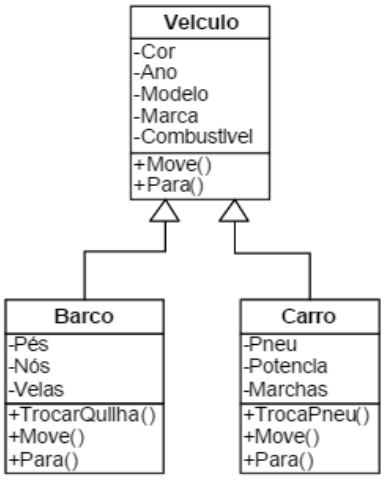
Figura 6 – Classe Carro estendendo a classe Veículo.



Fonte: Benedicto; Bezerra; Boscarioli; Delmiro (2006).

A característica do polimorfismo permite que um comportamento diferenciado seja implementado por duas classes, desde que sejam especializações de uma mesma classe e o método tenha a mesma assinatura (BENEDICTO; BEZERRA; BOSCARIOLI; DELMIRO, 2006). A partir da classe Veículo, cria-se uma nova classe Barco, como na Figura 7. Percebe-se que os comportamentos (métodos) Move e Para são comuns às três classes. Quando o programa estiver em execução, caso o método Move seja invocado na classe Veículo, as especificidades de cada uma das classes serão utilizadas a fim de causar o efeito desejado, independentemente de qual seja esta classe, mas pelo tipo da instância do objeto.

Figura 7 – Classes Carro e Barco especializadas em Veículo.



Fonte: Benedicto; Bezerra; Boscarioli; Delmiro (2006).

Por fim, destaca-se o encapsulamento, que está diretamente ligado à abstração de dados, permitindo que apenas as informações consideradas apropriadas pelo objeto sejam visualizadas externamente, de acordo com o contexto da aplicação. Este conceito também pode ser utilizado para acoplar dados com operações comumente executadas, como obter a idade de um Veículo. Seria possível, ainda, tornar o atributo Ano invisível para os objetos externos e permitir a visualização apenas da operação Idade (BENEDICTO; BEZERRA; BOSCARIOLI; DELMIRO, 2006).

2.4 Outros Conceitos importantes

Nesta seção, serão explicados alguns conceitos importantes para o entendimento da pesquisa.

2.4.1 Linguagem de Modelagem Unificada (UML)

A UML é uma linguagem visual para modelar sistemas orientados a objetos, ou seja, é uma linguagem que define elementos gráficos (visuais) que podem ser utilizados na modelagem de sistemas. Esses elementos permitem representar os conceitos do paradigma da orientação a objetos. Por meio deles, pode-se construir diagramas que representam diversas perspectivas de um sistema (BEZERRA, 2007).

Ainda de acordo com Bezerra (2007), a UML é independente tanto de linguagens de programação quanto de processos de desenvolvimento. Ela pode ser utilizada para a modelagem de sistemas, não importando a linguagem de programação que será utilizada na implementação do sistema ou a forma (processo) de desenvolvimento adotada.

A UML permite criar os diagramas necessários para a implementação de um sistema, ou seja, por meio dela, pode-se planejar como será o sistema, criar os diagramas de casos de uso, modelar o Banco de Dados e, assim, definir todos os detalhes antes do desenvolvimento (FERREIRA, 2013).

Para dar suporte à modelagem de sistemas, são comumente utilizadas ferramentas CASE. Essas ferramentas são abordadas no próximo tópico.

2.4.2 Ferramentas CASE

Existem sistemas de *software* que são utilizados para dar suporte ao ciclo de vida de desenvolvimento de um sistema. Dois tipos de *software* que têm esse objetivo são as ferramentas CASE (*Computer-Aided Software Engineering*) e os ambientes de desenvolvimento (BEZERRA, 2007).

Segundo Sommerville (2007), as ferramentas CASE, ou engenharia de *software* com o auxílio de computador, são *software*s utilizados para apoiar as atividades de processo de *software*. Este apoio é proporcionado pela automação de algumas atividades de processo e pelo fornecimento de informações sobre o *software* que está sendo desenvolvido. Sommerville (2007) ainda afirma que algumas funções das ferramentas CASE são: planejamento, edição, gerenciamento de mudança, prototipação, processamento de linguagem, testes, depuração, documentação, reengenharia, dentre outras.

De acordo com Ferreira (2013), a tecnologia das ferramentas CASE proporcionou diversas melhorias na qualidade e na produtividade de *software*. As ferramentas, atualmente, estão disponíveis para a maioria das atividades de rotina no processo de *software*.

Algumas ferramentas CASE existentes atualmente são: *Subversion, Microsoft Project, Eclipse, DBDesigner, NetBeans, Rational Rose, Astah Community, ArgoUML, Star UML, Postgres, JUnit, Dreamweaver*. Dentre estas, a *Astah Community*, a *DBDesigner* e a *NetBeans* foram essenciais para o desenvolvimento deste trabalho. A primeira auxiliou a criação de diagramas, a segunda a modelagem de entidades e relacionamentos, e a terceira a codificação do sistema.

2.4.3 *Framework*

Um *framework* é um conjunto de classes (abstratas e concretas) que podem prover uma estrutura ou implementação inicial para uma aplicação. Graças ao uso de *frameworks*, uma aplicação pode ser construída bem mais rapidamente do que se fosse construída sem nenhuma base inicial. Existem *frameworks* para a construção de diferentes aspectos de uma aplicação: interface gráfica com o usuário, criação de testes, persistência de objetos em um SGBD, dentre outros (BEZERRA, 2007). Este último é considerado um *framework ORM*, que será explicado no próximo tópico.

2.4.3.1 *Framework* ORM

Um *framework* ORM (*Object-Relational Mapping*) é um conjunto de classes que realiza o mapeamento transparente entre objetos da aplicação e tabelas em um SGBDR. Os *frameworks* ORM tentam resolver o problema do descasamento de informações fornecendo classes que o corrigem de forma transparente para o programador da aplicação. Normalmente, um *framework* necessita que o desenvolvedor forneça a correspondência entre a estrutura de objetos da aplicação e o esquema relacional do Banco de Dados. Esta correspondência é fornecida por um arquivo de configuração, denominado arquivo de mapeamento. De posse da correspondência, o *framework* estará apto a mapear qualquer requisição por uma informação armazenada no SGBDR (BEZERRA, 2007).

2.4.3.2 *Framework MVC*

É notória a dificuldade de manutenção de sistemas *web* quando códigos HTML (*HyperText Markup Language*) se misturam a código Java. Atenta a este problema, a comunidade de desenvolvedores começou a criar ferramentas que auxiliam na separação das responsabilidades na aplicação. Estas ferramentas são conhecidas como *frameworks* MVC (*Model-view-controller*) (FRASSON, 2014).

Segundo Frasson (2014), o MVC separa a codificação em três etapas:

- Modelo: parte do código onde estão representados os objetos de negócio, que por sua vez, mantêm o estado da aplicação e fornecem informações do Banco de Dados ao controlador. O modelo é responsável pelos dados, regras de negócio e modificação dos dados, realizando também a comunicação com a base de dados e outros sistemas existentes.
- Visualização: responsável pela interface do usuário. Na camada de visualização, é definida a forma como os dados são apresentados e são encaminhadas para o controlador as ações dos usuários.
- Controle: responsável por efetuar a ligação entre o modelo e a visualização. O controle apresenta as solicitações dos usuários, aciona as opções corretas a serem realizadas no sistema, encaminha essas ações ao modelo, e retorna as visualizações das solicitações correspondentes.

De acordo com Frasson (2014), a utilização do MVC garante algumas vantagens no processo de desenvolvimento. Dentre elas, está o reaproveitamento dos componentes do modelo. Ao utilizar o MVC, é possível usar os objetos de negócio definidos na camada

modelo em diversas classes/arquivos de visualização. Além disso, com o aumento da complexidade do *software*, a sua manutenção torna-se um processo mais difícil. A separação das responsabilidades em camadas pré-estabelecidas deixa a manutenção do código mais compreensível.

Diante da aceitação do MVC pela comunidade de programadores Java, surgiram alguns *frameworks* baseados na ferramenta. Os mais conhecidos são: *JSF, Struts, VRaptor, Spring,* dentre outros.

No tópico a seguir, será explicado com mais ênfase os *frameworks* ORM e MVC, usados para o desenvolvimento do *software* deste trabalho.

2.4.4 *Hibernate* e JSF

O *Hibernate* foi o *framework* ORM utilizado para a implementação do sistema desta pesquisa. Esse é o *framework* mais conhecido do mercado e, como os demais existentes, disponibiliza toda a programação necessária para o mapeamento objeto-relacional e a persistência dos dados. Com a aceitação dos *frameworks* ORM pela comunidade de programadores e a previsão de surgimento de muitos outros *frameworks* ORM, foi incluída, a partir do Java 5, uma interface de persistência chamada *Java Pesistence API* ou JPA, que padroniza o mapeamento objeto-relacional. A JPA possibilita trabalhar diretamente com as classes sem a utilização de consultas nativas de cada bando de dados (FRASSON, 2014).

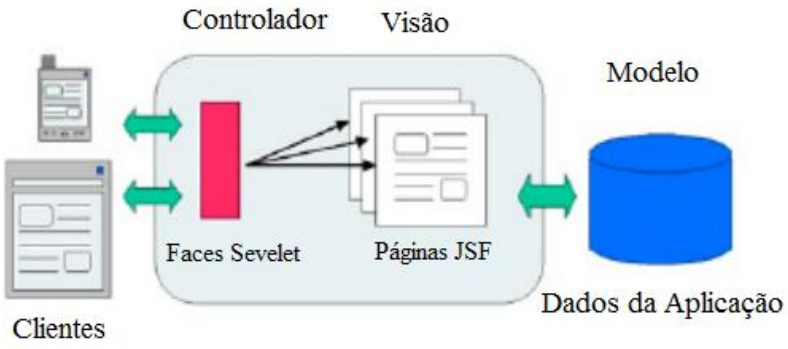
Algumas características da JPA, como afirma Frasson (2014), são:

- Qualquer classe que contenha um construtor sem parâmetros pode se tornar persistente sem nenhuma alteração no código; apenas algumas anotações são necessárias;
- Como a JPA oferece diversos valores padrão, bastam poucas anotações para efetuar o mapeamento;
- As consultas com a JPA são efetuadas por meio da JPQL (*Java Persistence Query Language*) que possui sintaxe similar às *queries* SQL.

Já o *Java Server Faces* foi o *framework* MVC utilizado para o desenvolvimento da aplicação desta pesquisa. Também conhecido como JSF, é uma tecnologia para desenvolvimento *web* que utiliza um modelo de interfaces gráficas baseado em eventos. Esta tecnologia foi definida pelo JCP (*Java Community Process*), o que a torna um padrão de desenvolvimento e facilita o trabalho dos fornecedores de ferramentas, pois possibilita a criação de produtos que valorizam a produtividade no desenvolvimento de interfaces visuais (FARIA, 2013).

O JSF é baseado no MVC, e a separação das camadas fica muito clara para o desenvolvedor. Em JSF, o controle é feito por meio de uma *servlet* chamada *Faces Servlet (opcional)*, por arquivos de configuração XML (*eXtensible Markup Language*) e por vários manipuladores de ações e observadores de eventos. A *Faces Servlet* recebe as requisições dos usuários na *web*, redireciona para o modelo e envia uma resposta. O modelo é representado por objetos de negócio, que executam uma lógica de negócio ao receber dados oriundos da camada de visualização. A visualização é composta por uma hierarquia de componentes (*component tree*), o que torna possível unir componentes para construir interfaces mais ricas e complexas (FARIA, 2013). A Figura 8 mostra a estruturação dos componentes JSF.

Figura 8 – Estrutura JSF.



Fonte: Adaptado de Faria (2013).

Algumas vantagens do JSF, de acordo com Frasson (2014), são:

- Componentes customizados: além dos componentes básicos fornecidos pelo *framework*, podem ser utilizadas bibliotecas de componentes de terceiros e/ou fabricação de novos componentes;
- Conversão de dados: os dados digitados pelo usuário podem facilmente ser convertidos para tipos específicos como datas, números, dentre outros;
- Validação: o JSF facilita o processo de validações básicas, como campos requeridos, formatos de data, telefone, dentre outros;
- Manipulação de erros: a manipulação de erros e a customização de mensagens de erro apropriadas são facilitadas com o uso do JSF.

2.5 A Sistematização da Assistência de Enfermagem (SAE)

A Sistematização da Assistência de Enfermagem (SAE) é uma ferramenta estratégica de investigação, que visa a uma assistência elaborada e eficiente, que agilize e qualifique o processo de trabalho da equipe de enfermagem e ofereça cuidado humanizado ao paciente, permitindo estabelecer vínculos entre ele e a equipe e proporcionando autonomia para o profissional de enfermagem (CHAVES, 2013). O termo “sistematização” remete ao real papel da SAE, que é sistematizar/organizar o Processo de Enfermagem (PE) em cinco etapas: Investigação, Diagnóstico, Planejamento, Implementação e Avaliação (ALFARO-LEFREVE, 2005).

De acordo com Neves (2006), a primeira etapa da Sistematização da Assistência de Enfermagem é o Histórico de Enfermagem ou Investigação, que consiste na coleta de dados fundamentada em três caracteres básicos de necessidade psicobiológica, psicossocial e psicoespiritual, compondo um roteiro sistematizado e hierarquizado. Já o Diagnóstico de Enfermagem consiste em uma identificação das necessidades básicas individuais, familiares ou em sociedade; por meio dele, o enfermeiro determina a extensão de dependência do paciente.

A terceira fase é o Plano Assistencial de Enfermagem ou Planejamento, que consiste no plano de cuidado estabelecido a partir do Diagnóstico de enfermagem. A quarta etapa da SAE caracteriza-se pela Prescrição de Enfermagem, tida como um roteiro que direciona a equipe de enfermagem para executar os cuidados anteriormente planejados, adaptando as necessidades básicas e individuais de cada paciente. Por fim, a última etapa consiste na avaliação dos resultados, relatando as mudanças sucessivas individuais ou coletivamente assistidas pela enfermagem (NEVES, 2006).

Essa sistematização é empregada para a detecção de diversas doenças, inclusive o câncer ginecológico. De fato, como foi mencionado no primeiro capítulo desta pesquisa, essa é a metodologia adotada pela Clínica Escola da UnP para detectar o câncer ginecológico. No ambiente da clínica, os enfermeiros realizam os processos através do preenchimento (a lápis ou caneta) dos formulários impressos em papel, que possuem os requisitos estabelecidos pela SAE.

No capítulo a seguir, será mostrado todo o processo de desenvolvimento do sistema para informatizar as etapas da SAE, desde o cadastro de paciente até a realização dos exames necessários para o efetivo uso dessa metodologia.

3 DESENVOLVIMENTO DO SISTEMA

O desenvolvimento do sistema desta pesquisa foi definido com base nas práticas de Engenharia de *Software*. O objetivo de usar essas práticas é chegar a um produto de *software* por meio de atividades e um ciclo de vida bem definidos, tendo como base as necessidades do cliente.

Para o processo de desenvolvimento do sistema, será utilizado o modelo em cascata, pois, diante dos modelos apresentados no capítulo anterior, este é mais viável, pois os requisitos do projeto foram muito bem reconhecidos e definidos; ainda deve-se levar em consideração que a implantação do produto final, neste caso, o *software*, só poderá ser realizada no mês de novembro. Essa circunstância descarta o uso do modelo iterativo e incremental, uma vez que este modelo gera protótipos para implantação a cada iteração concluída.

Utilizando uma abordagem baseada em disciplinas, o desenvolvimento do projeto será guiado por casos de uso. Seu ciclo de vida é em cascata, projetado e documentado utilizando a notação UML. No entanto, o projeto é leve, eficiente, flexível e possui os valores da simplicidade, comunicação e *feedback* definidos em processos ágeis.

As fases de desenvolvimento deste trabalho serão:

- *Definição de requisitos*: fase na qual serão conhecidos os *stakeholders (partes interessadas)* e suas necessidades para o sistema, gerando artefatos de especificação de requisitos, casos de uso, entre outros.
- *Elaboração do projeto*: é feita a análise de forma mais detalhada sobre o domínio do problema, assim como a modelagem da arquitetura do projeto. Nesta fase, são gerados artefatos como diagramas e modelagem do banco de dados.
- *Implementação*: fase na qual ocorre a concretização dos requisitos, arquitetura e modelagem propostos, gerando um produto de *software*.

Os próximos tópicos apresentam detalhadamente cada fase de desenvolvimento deste trabalho.

3.1 Especificação de Requisitos

Na fase da especificação de requisitos, o objetivo principal é conhecer os *stakeholders*, ou seja, as partes interessadas no sistema, e ainda analisar as necessidades, avaliar a

possibilidade de desenvolvimento computacional, especificar a solução de modo não ambíguo, negociar uma condição razoável e validar a especificação.

Os *stakeholders* são pessoas que têm interesse ou são afetadas pelo projeto de alguma forma. Os *stakeholders* desse projeto são:

- *Analista de Sistemas*: realiza o levantamento e análise dos requisitos do *software*;
- *Engenheiro de Software*: responsável pela implementação do sistema e pelo *design* da aplicação;
- *Projetista de Banco de Dados*: realiza o projeto de banco de dados da aplicação;
- *Enfermeiro*: funcionário da Clínica Escola da UnP que gerencia os dados dos pacientes e provê informações necessárias para o desenvolvimento da aplicação;
- *Paciente*: paciente da Clínica Escola da UnP que fornece seus dados e outras informações para preenchimento do banco de dados do sistema, e também provê informações necessárias para o desenvolvimento da aplicação.

3.1.1 Requisitos do Sistema

O *software* desta pesquisa tem por objetivo gerenciar as informações relevantes das atividades da SAE executadas pelos enfermeiros, permitindo o acesso aos dados via *web*. Este sistema deve ser acessível aos enfermeiros e, por isso, é necessário que esteja disponível em um servidor *web* para que os usuários possam acessá-lo a partir de qualquer computador com um navegador *web* e acesso à *Internet.*

Para a identificação dos requisitos, foram realizadas reuniões presenciais entre os enfermeiros e a equipe de desenvolvimento. Durante essas reuniões, foram levantados os requisitos indispensáveis, considerando os processos da SAE realizados pelos enfermeiros. Esses processos eram impressos em formulários, que os enfermeiros preenchiam a lápis ou caneta. Os formulários, então, serviram como fonte principal para o desenvolvimento do sistema, pois todos os processos já estavam muito bem articulados e elaborados. De fato, o que não existia era uma ferramenta computacional capaz de gerenciar todos os processos e dados envolvidos.

O sistema deve abranger dois tipos de usuários: o Enfermeiro Administrador e o Enfermeiro Comum. O usuário Enfermeiro Administrador gerencia o cadastro dos Enfermeiros Comuns e também o cadastro dos pacientes e seus respectivos exames. Já o usuário Enfermeiro Comum tem apenas acesso ao cadastro de pacientes e seus exames. Os dois tipos de usuários executam tarefas semelhantes (cadastro de pacientes e seus exames), a

única diferença é que o Enfermeiro Administrador pode executar tarefas relativas aos Enfermeiros Comuns, de forma que estes possam, assim, ser cadastrados e ter acesso ao sistema.

Apenas para o Enfermeiro Administrador, estão disponíveis no sistema as funcionalidades de cadastro de enfermeiro, listagem dos enfermeiros cadastrados, busca de enfermeiro, edição e exclusão de dados de enfermeiros. Porém, tanto para o Enfermeiro Administrador como para o Enfermeiro Comum, estão disponíveis no sistema as funcionalidades de cadastro de paciente, listagem dos pacientes cadastrados, busca de paciente, edição e exclusão de dados dos pacientes, cadastro de exame do paciente, listagem dos exames dos pacientes, edição e exclusão dos exames dos pacientes.

Os exames correspondem a todo o processo da SAE, que para ser executado adequadamente, precisa realizar, sem exceção, todos os exames. Os exames são: inspeção das mamas, palpação das mamas, inspeção vulvar, palpação vulvar, citopatológico, toque bimanual, diagnóstico de enfermagem, prescrição de enfermagem, avaliação e planejamento.

A seguir, os requisitos do sistema serão listados e classificados como funcionais e não funcionais.

O *software* deste trabalho tem como requisitos funcionais:

- Para o Enfermeiro Administrador:
    - Cadastro de enfermeiros;
    - Edição do cadastro de enfermeiros;
    - Exclusão do cadastro de um enfermeiro;
    - Listagem dos enfermeiros cadastrados;
    - Busca por um enfermeiro cadastrado;
- Para o Enfermeiro Administrador e Comum:
    - Cadastro de pacientes;
    - Edição do cadastro de pacientes;
    - Exclusão do cadastro de um paciente;
    - Listagem dos enfermeiros cadastrados;
    - Busca por um enfermeiro cadastrado;
    - Cadastro de exame “X” do paciente;
    - Edição do cadastro de exame “X” do paciente;
    - Exclusão do cadastro de um exame “X” do paciente;
    - Listagem dos exames “X” do paciente.

O “X” equivale ao tipo de exame que o enfermeiro irá cadastrar para o paciente.

Os requisitos não funcionais para os dois usuários são: usabilidade, segurança e acesso remoto.

3.1.2 Diagrama de Arquitetura do Software

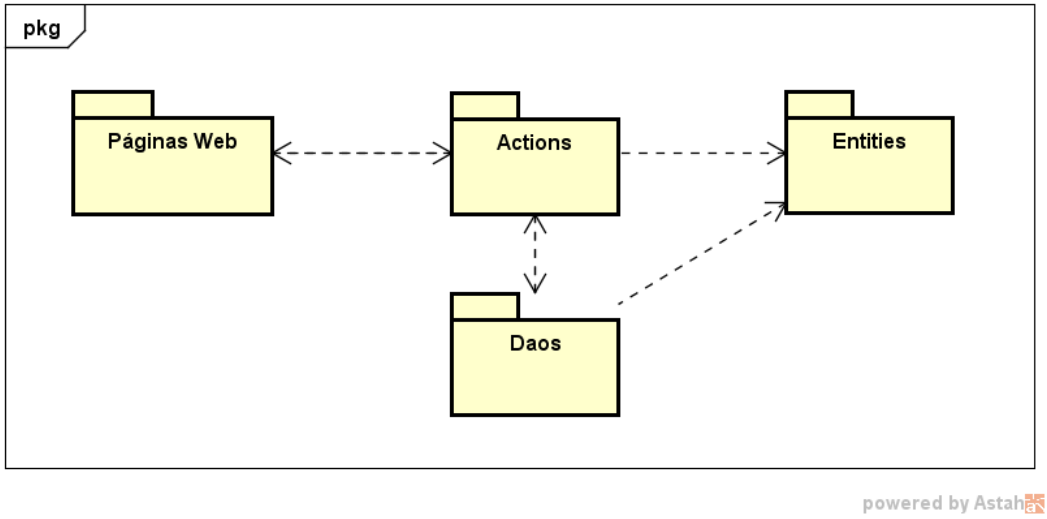
A arquitetura consiste em um modelo de alto nível que possibilita um entendimento e uma análise mais fácil do software a ser desenvolvido. O uso de arquitetura para representar soluções de software foi incentivada principalmente por duas tendências (GARLAN e PERRY, 1995; KAZMAN, 2001):

- O reconhecimento por parte dos projetistas que o uso de abstrações facilita a visualização e o entendimento de certas propriedades do software;
- a exploração cada vez maior de frameworks visando diminuir o esforço de construção de produtos através da integração de partes previamente desenvolvidas.

Outra propriedade da arquitetura é a possibilidade de usá-la como ferramenta para comunicar a solução projetada aos diversos stakeholders que participam do processo de desenvolvimento do software (GARLAN, 2000). Contudo, para que essa comunicação seja possível, a arquitetura deve ser representada através de um documento, conhecido como documento arquitetural.

A figura 9 apresenta o Diagrama de Arquitetura de Software para o sistema desta pesquisa.

Figura 9 – Diagrama de Arquitetura de Software.



Fonte: Autoria própria.

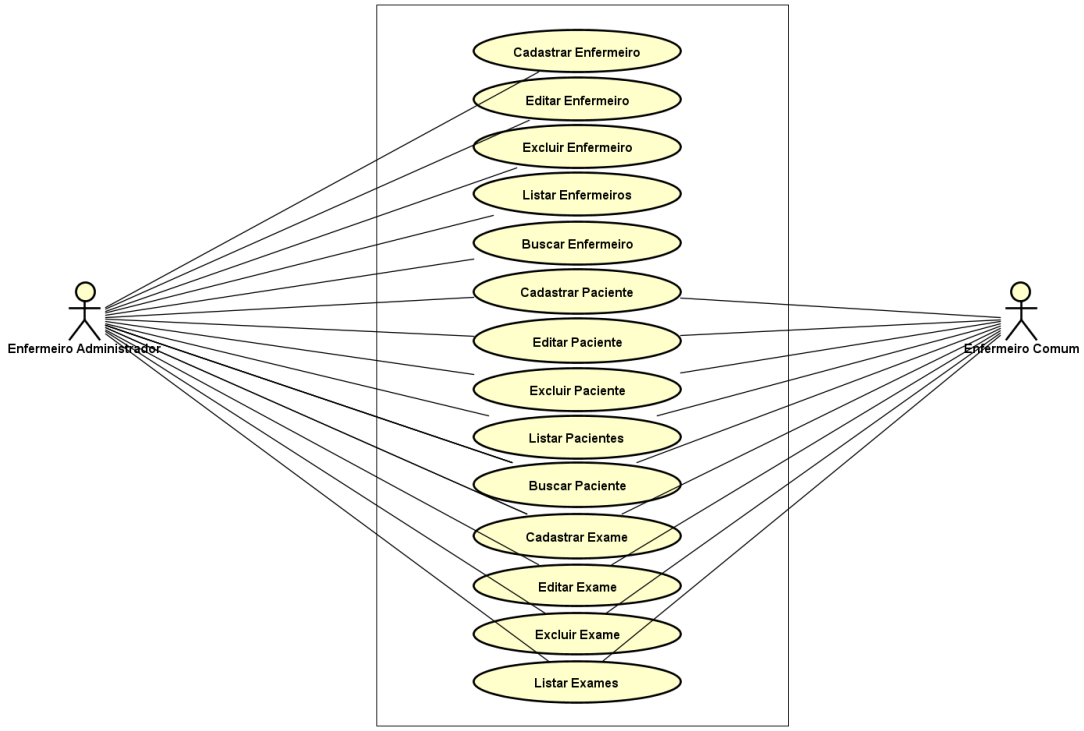
O Diagrama de Arquitetura de Software da Figura 9 apresenta quatro pacotes que são: Páginas *Web*, *Actions*, *Entities*, *Daos.* No pacote Páginas *Web*, de fato, estão as páginas *web* do sistema que interagem com as instâncias das classes do pacote *Actions*. As instâncias das classes desse pacote interagem com as do pacote *Daos*, que por sua vez, realiza a interação com o Banco de Dados associado. O pacote *Entities* possui as classes que realizam todo o mapeamento das entidades do Banco de Dados.

3.1.3 Diagrama de Caso de Uso

Os Casos de Uso descrevem as funcionalidades que serão construídas no sistema proposto e também a sequência de eventos de um ator que usa a aplicação para completar um dos processos. Cada requisito funcional foi formalizado em um Caso de Uso e pode ser encontrado no Apêndice A desta pesquisa.

A figura 10 apresenta o Diagrama de Casos de Uso relacionados aos atores Enfermeiro Administrador e Enfermeiro Comum.

Figura 10 – Diagrama de Casos de Uso.



Fonte: Autoria própria.

O diagrama acima, que foi modelado a partir da notação UML e a ferramenta CASE *Astah Community*, tem por objetivo descrever uma visão externa do sistema e representar as relações entre os atores Enfermeiro Administrador e Enfermeiro Comum, e os casos de uso a eles associados. Os atores são representados pelos bonecos localizados à direita e esquerda da figura. Cada Caso de Uso é representado pelas elipses que estão no centro da imagem. As linhas que ligam os atores aos Casos de Uso representam a comunicação entre eles, o que significa que cada ator interage com o sistema, trocando informações por meio desses Casos de Uso.

As funcionalidades propostas pelos Casos de Uso foram implementadas e são apresentadas no tópico 3.3.

3.2 Projeto

A fase de projeto é a parte da engenharia de *software* que se encarrega de transformar os resultados da análise de requisitos em um documento ou conjunto de documentos e/ou diagramas, capazes de serem interpretados pelo engenheiro de *software*.

Nesta pesquisa, foram elaborados o Diagrama de Classes e o Diagrama de Entidade e Relacionamento, apresentados nos tópicos a seguir.

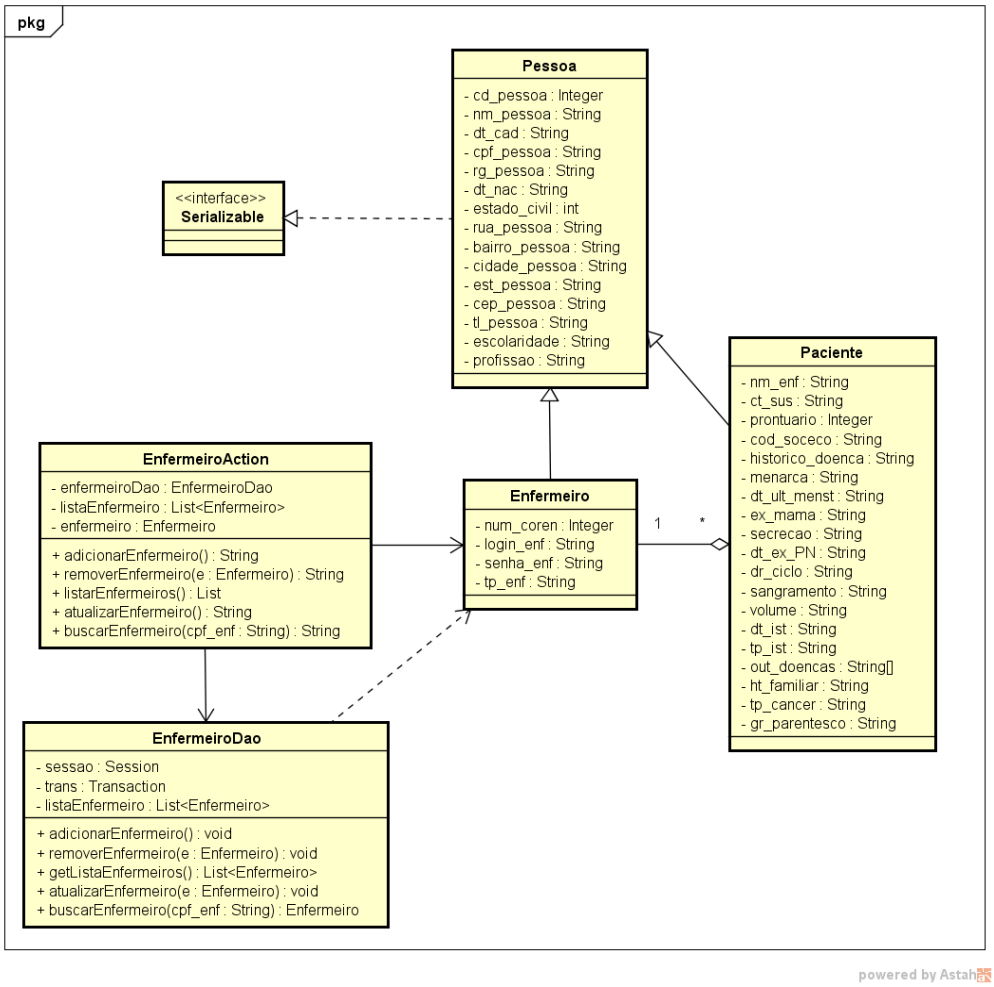
3.2.1 Diagrama de Classes (DC)

O Diagrama de Classes (DC) é uma modelagem útil e muito significativa para o desenvolvimento de sistemas. Este diagrama, cuja modelagem também é feita utilizando UML, deve ser compreendido pela equipe de desenvolvimento que definirá todas as classes, com seus atributos e operações necessários para o *software*. Neste DC, as classes são representadas por retângulos; os relacionamentos entre classes são representados por linhas; e a conectividade de associação entre classes é indicada pelos números acima das linhas.

Para uma melhor organização e estruturação do projeto deste trabalho, as classes foram separadas por pacotes, sendo os principais: *Entities, Daos e Actions.* O pacote *Entities* conterá as classes que fazem o mapeamento com as tabelas do BD. Já o pacote *Daos* conterá as classes que interagem diretamente com o BD, de acordo com a *Entity* equivalente. E o pacote *Actions* conterá as classes que fazem o mapeamento com as páginas *web*, também de acordo com a *Entity* equivalente.

Os Diagramas de Classes do sistema dessa pesquisa foram construídos com base em módulos. O primeiro DC apresenta o módulo do *Enfermeiro*, o segundo mostra o módulo de *Paciente* e o terceiro expõe o módulo de *Exame*. A figura 11 apresenta o Diagrama de Classes do módulo *Enfermeiro*.

Figura 11 – Diagrama de Classes do módulo Enfermeiro.



Fonte: Autoria própria.

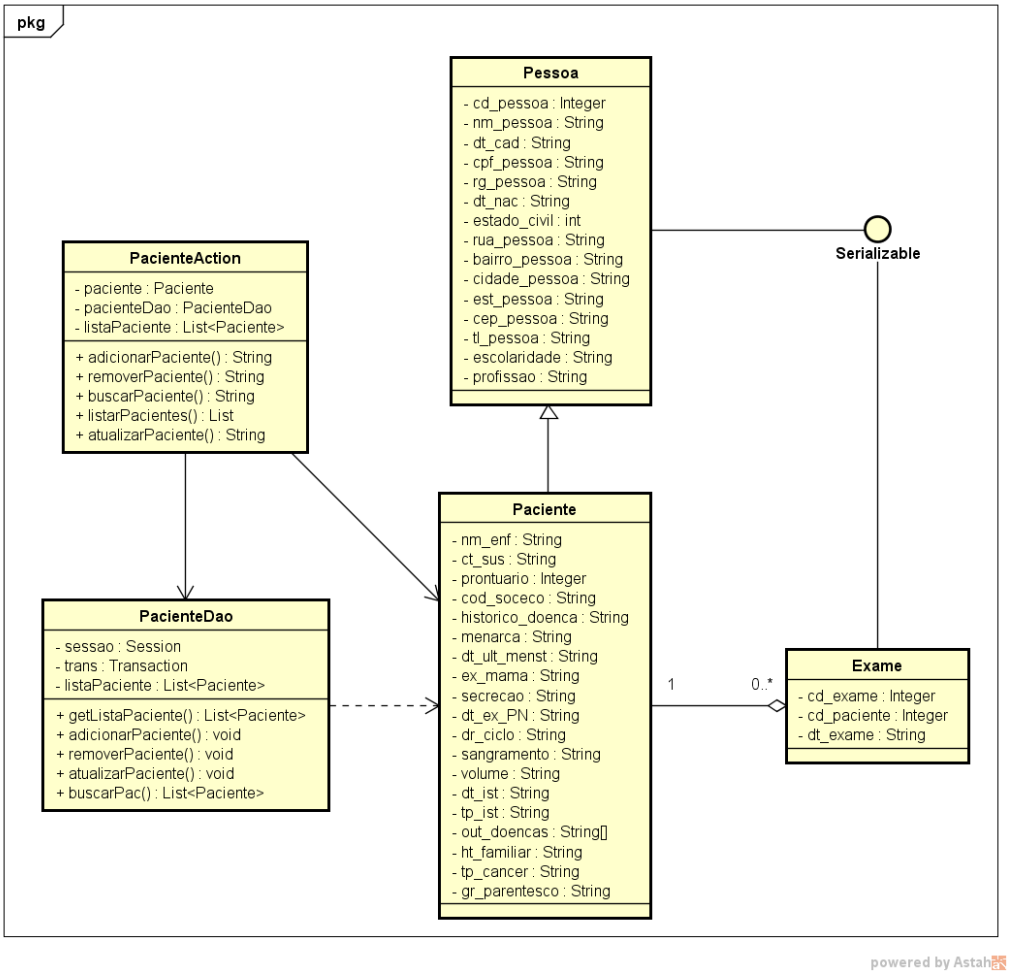
No DC do módulo Enfermeiro, é possível perceber que *Pessoa* implementa a interface *Serializable.* Esta classe tem como objetivo salvar, gravar e capturar o estado de um objeto. As relações de *Enfermeiro* e *Paciente* com *Pessoa* são de herança, representadas pelos segmentos de retas com setas de cor branca, de forma que as duas primeiras herdam os atributos e métodos da última, tornando-se também classes serializáveis. Os segmentos de retas com setas representam uma relação de associação, que é formada por instâncias de objetos durante a execução do sistema. Assim, tem-se que *EnfermeiroAction* está associada aos objetos das classes *Enfermeiro* e *EnfermeiroDao.* O fragmento de reta tracejada que liga

EnfermeiroDao e *Enfermeiro* significa que a primeira possui uma relação de dependência com a segunda.

As relações de agregação são um tipo especial de associação, representadas por uma linha com um losango branco tocando uma das classes. O objeto da classe tocada pelo losango vai referenciar um objeto da outra classe agregada. Como exemplo de agregação, na Figura 10, pode-se citar um objeto da classe *Paciente*, que conterá uma referencia do objeto do tipo *Enfermeiro*, uma vez que, ao cadastrar um paciente, é necessário saber qual enfermeiro o cadastrou.

A conectividade do DC acima é definida pela associação de *Enfermeiro* e *Paciente*, pois um *Enfermeiro* pode cadastrar um ou muitos *Pacientes*, e um ou muitos *Pacientes* são cadastrados por um *Enfermeiro*. A seguir, na Figura 12, será apresentado o Diagrama de Classes do módulo *Paciente*.

Figura 12 – Diagrama de Classes do módulo Paciente.



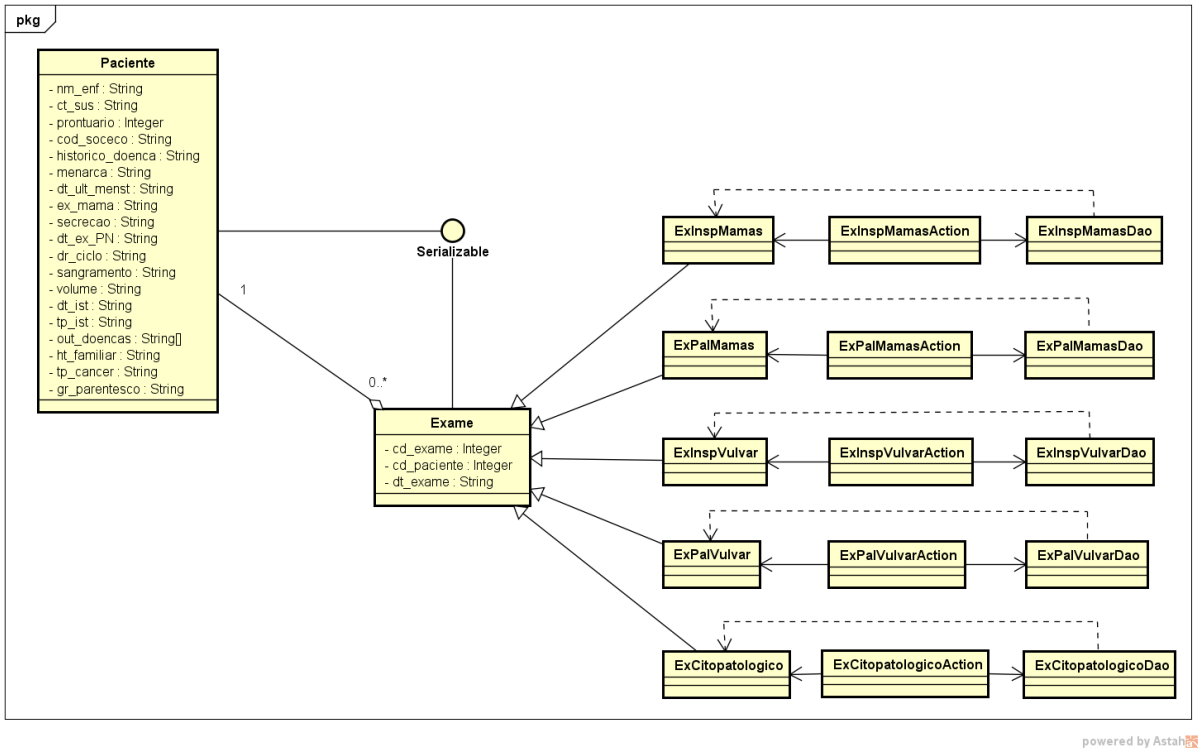
Fonte: Autoria Própria.

No DC do módulo Paciente, a interface *Serializable* e as classes *Pessoa* e *Paciente* são exibidas novamente para a melhor compreensão da equipe de desenvolvimento. Como mencionado anteriormente, *Pessoa* implementa a interface *Serializable*, e *Paciente* herda os atributos e métodos de *Pessoa*. A associação existente nesse DC é formada por *PacienteAction*, *Paciente* e *PacienteDao*. Há também uma relação de dependência entre *PacienteDao* e *Paciente*, sendo que a primeira é dependente da última.

A agregação existente, na Figura 11, ocorre entre as classes *Paciente* e *Exame*, uma vez que é imprescindível saber a quem pertencem os exames cadastrados. A conectividade dessa associação é assim definida: um paciente possui zero ou muitos exames; zero ou muitos exames pertencem a um paciente.

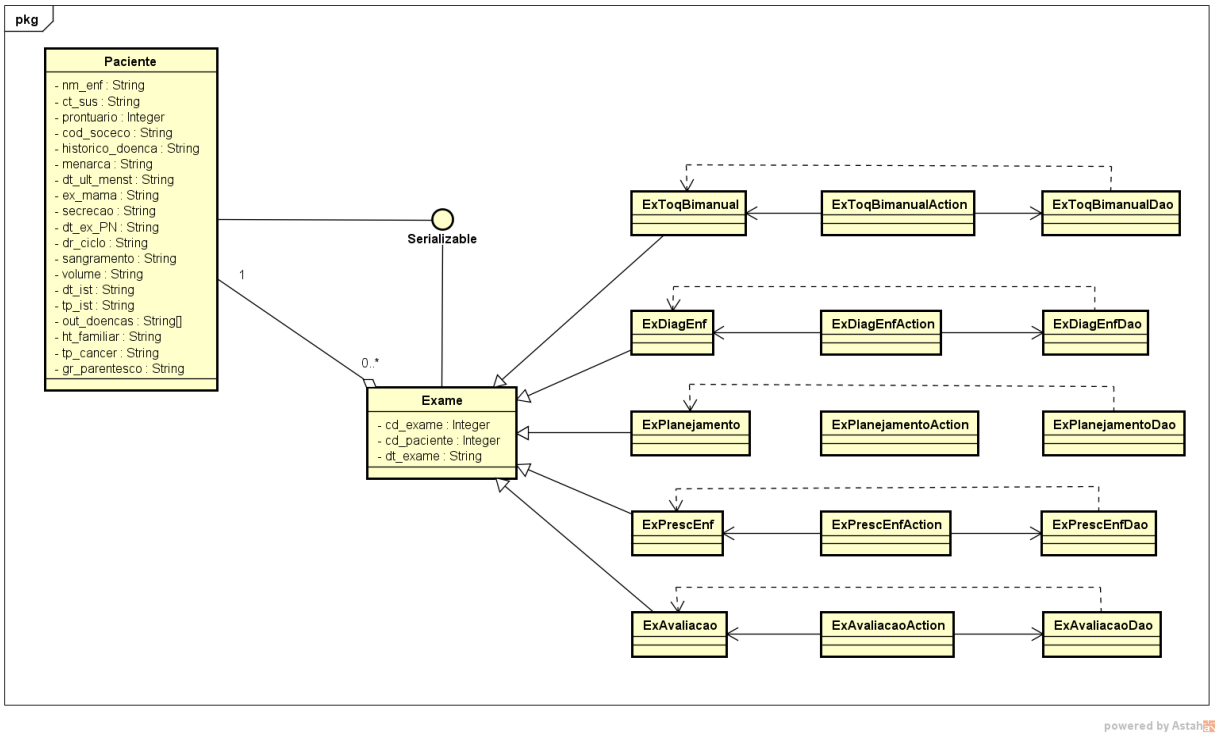
As figuras 13 e 14 expõem o módulo de Exames. Nota-se, nas figuras, que as classes referentes aos tipos de exames estão sem seus atributos e métodos. No entanto, é possível visualizar cada classe em seu estado completo no apêndice B deste trabalho. O DC deste módulo foi dividido em duas etapas para uma melhor visualização do que foi projetado.

Figura 13 – Diagrama de Classes do módulo Exames – Parte I.



Fonte: Autoria própria.

Figura 14 – Diagrama de Classes do módulo Exames – Parte II.



Fonte: Autoria própria.

Nos Diagramas de Classes do módulo Exames, ocorrem relações de herança entre as classes referentes aos tipos exames, ou seja, as classes *ExAvalicao*, *ExCitopatologico*, *ExDiagEnf*, *ExInspMamas*, *ExInspVulvar*, *ExPalMamas*, *ExPalVulvar*, *ExPlanejamento*, *ExPrescEnf* e *ExTocBimanual* herdam os atributos e métodos de *Exame*.

As relações de associação existentes nesses DC são formadas pelas classes *Actions* e *Daos* com a classe *Entity* equivalente. Há, por exemplo, uma associação entre as classes *ExAvalicaoAction, ExAvalicaoDao e ExAvalicao.* As classes *Daos* dependem das classes *Entities* equivalentes.

3.2.2 Diagrama de Entidade e Relacionamento (DER)

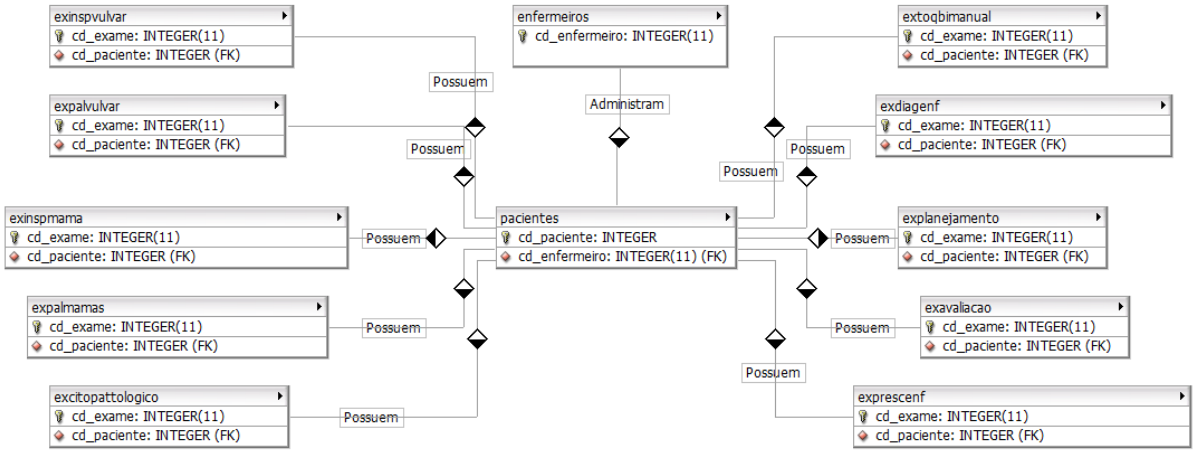
O Diagrama de Entidade e Relacionamento (DER) é a representação gráfica do Modelo de Entidade e Relacionamento (MER). O MER é um modelo abstrato que tem como finalidade descrever, de maneira conceitual, os dados a serem utilizados em um sistema de informação. Este modelo é uma representação da realidade, composto por entidades, relacionamentos e atributos.

Neste diagrama, as entidades são representadas por retângulos: na parte superior, é descrito o nome da entidade; na parte interna, são descritos os seus atributos. As linhas que ligam as entidades representam os relacionamentos entre elas.

Em termos de Bancos de Dados, cada entidade será uma tabela, e os seus atributos serão as colunas da tabela. Os relacionamentos determinam o modo como cada registro de uma tabela se associa a registros de outras – quando um paciente pode possuir vários exames, por exemplo.

A figura 15 apresenta o Diagrama de Entidade e Relacionamento (DER) do sistema desta pesquisa.

Figura 15 – Diagrama de Entidade e Relacionamento.



Fonte: Autoria própria.

Neste DER, as tabelas são representadas pelos retângulos, e os relacionamentos entre elas, pelas linhas. Os losangos indicam o tipo de relacionamento que as tabelas possuem. O diagrama mostra apenas um tipo de relacionamento denominado *1..n* (um para muitos), que é indicado pelo losango preto e branco, cuja parte branca equivale ao *1*, e a parte preta, ao *n*. Desta forma, a partir do DER, conclui-se que as entidades da tabela “enfermeiros” administram as da tabela “pacientes”, e que as entidades “pacientes” possuem as das tabelas de exames (representadas pelas tabelas com o prefixo “ex”).

3.3 Desenvolvimento

A linguagem de programação escolhida para o desenvolvimento do sistema foi a JAVA, por ser uma linguagem bastante comercial e reconhecida pela maioria dos programadores.

A interface foi desenvolvida utilizando a Linguagem de Marcação de Hipertexto (HTML) com a linguagem de estilo *Cascading Style Sheets* (CSS) e também a Java Server Faces (JSF).

No que se refere ao armazenamento dos dados, primeiramente no âmbito dos bancos de dados relacionais, a aplicação foi desenvolvida usando o *MariaDB*, banco de dados que utiliza a linguagem SQL como interface e é atualmente um dos mais usados comercialmente. Na sequência, já no âmbito dos bancos de dados orientados a objetos, a aplicação foi desenvolvida usando o *DB4O (Database for Objects)*, um dos bancos mais conhecidos no paradigma da orientação a objetos.

Todo este processo foi desenvolvido a partir dos *frameworks Hibernate* e JSF, baseado nas definições da especificação de requisitos e nos diagramas produzidos e apresentados nas seções anteriores.

3.3.1 Sistema de Assistência de Enfermagem – Rastreamento de Câncer Ginecológico

O Sistema de Assistência de Enfermagem – Rastreamento de Câncer Ginecológico (SAE-GIN), nome dado ao *software* desta pesquisa, é um sistema de informação *web* que auxiliará os enfermeiros da Clínica Escola da UnP na detecção de câncer ginecológico. Esta aplicação provê apoio aos enfermeiros e será utilizada para gerenciar os processos da SAE que são aplicados nas pacientes da clínica, que até então eram registrados em papel.

O SAE-GIN possui dois tipos de usuários: o Enfermeiro Administrador e o Enfermeiro Comum. O Enfermeiro Administrador é responsável por gerenciar o cadastro dos enfermeiros (Administrador ou Comum) que passarão a ter acesso ao sistema; ele também pode gerenciar o cadastro dos pacientes e seus respectivos exames. O Enfermeiro Comum tem como responsabilidade o gerenciamento do cadastro de pacientes e seus exames. Na Figura 16, é apresentada a tela inicial do SAE-GIN, na qual os usuários realizam o *login* e acessam as funcionalidades do *software*.

Figura 16 – Tela inicial do SAE-GIN.

Fonte: Autoria própria.

O usuário, após efetuar o *login*, terá acesso a um menu principal, no qual poderá acessar as funcionalidades disponíveis de acordo com o tipo de usuário. Nos tópicos seguintes, serão apresentados os módulos dos usuários Enfermeiro Comum e Enfermeiro Administrador.

3.3.1.1 Módulo do usuário Enfermeiro Comum

Para o usuário do tipo Enfermeiro Comum, estão disponíveis as funcionalidades de cadastro de paciente, listagem das pacientes cadastradas, edição de dados de paciente, exclusão do cadastro de paciente, busca de paciente, cadastro de exame “X”, listagem dos exames “X” cadastrados, edição de dados de exame “X” e exclusão do cadastro de exame “X”. O “X” representa um exame do processo SAE que é realizado para uma paciente. A tela de menu principal do Enfermeiro Comum, que é exibida após a efetuação do *login*, é mostrada na Figura 17.

Figura 17 – Menu principal do Enfermeiro Comum.

Fonte: Autoria própria.

O menu principal do Enfermeiro Comum oferece três opções: “Início”, “Pacientes” e “Sair”. A opção “Início” faz o usuário voltar para o menu principal. A opção “Sair” faz o usuário sair do sistema, ou seja, é o *logout* do sistema. A Figura 18 mostra a tela exibida quando o Enfermeiro Comum acessa a opção “Pacientes”.

Figura 18 – Listagem das pacientes cadastradas.

Fonte: Autoria própria.

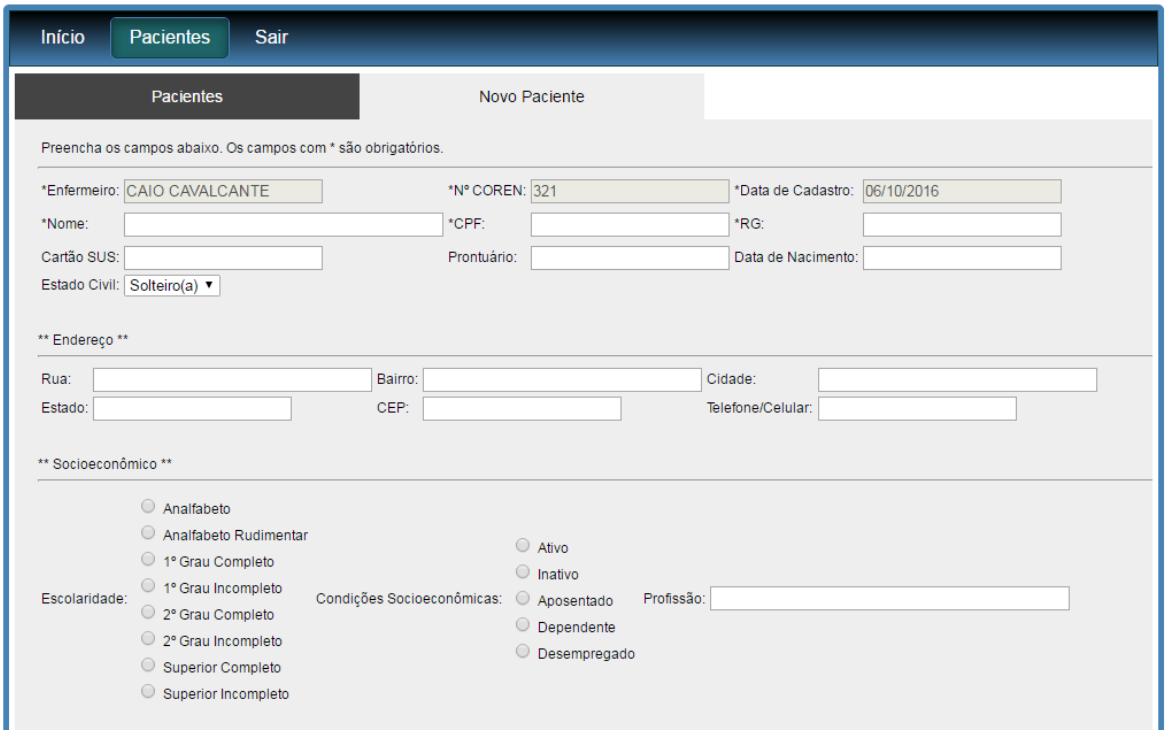
Na opção “Pacientes”, o Enfermeiro Comum tem acesso a uma tela que exibe a listagem das pacientes já cadastradas. Nota-se que, nessa listagem, o código e o nome das pacientes são exibidos, além dos botões:

- Excluir, para excluir a paciente, caso seja necessário;
- Editar, para editar os dados da paciente, se necessário;
- Exames, parar realizar o cadastro dos exames do processo SAE;
- Imprimir, para realizar a impressão dos dados da paciente.

Para o Enfermeiro Comum realizar alguma das ações listadas acima, basta clicar sobre o botão correspondent. A tela também oferece a realização de busca de paciente. Essa busca pode acontecer pelo número do Cadastro de Pessoa Física (CPF), pelo nome ou ainda pelo número do Registro Geral (RG) da paciente.

A tela ainda fornece a opção para cadastrar uma nova paciente. A Figura 19 mostra a tela de cadastro de paciente. Ela é exibida quando o Enfermeiro Comum clica sobre a opção “Novo Paciente”.

Figura 19 – Cadastro de paciente.



Fonte: Autoria própria.

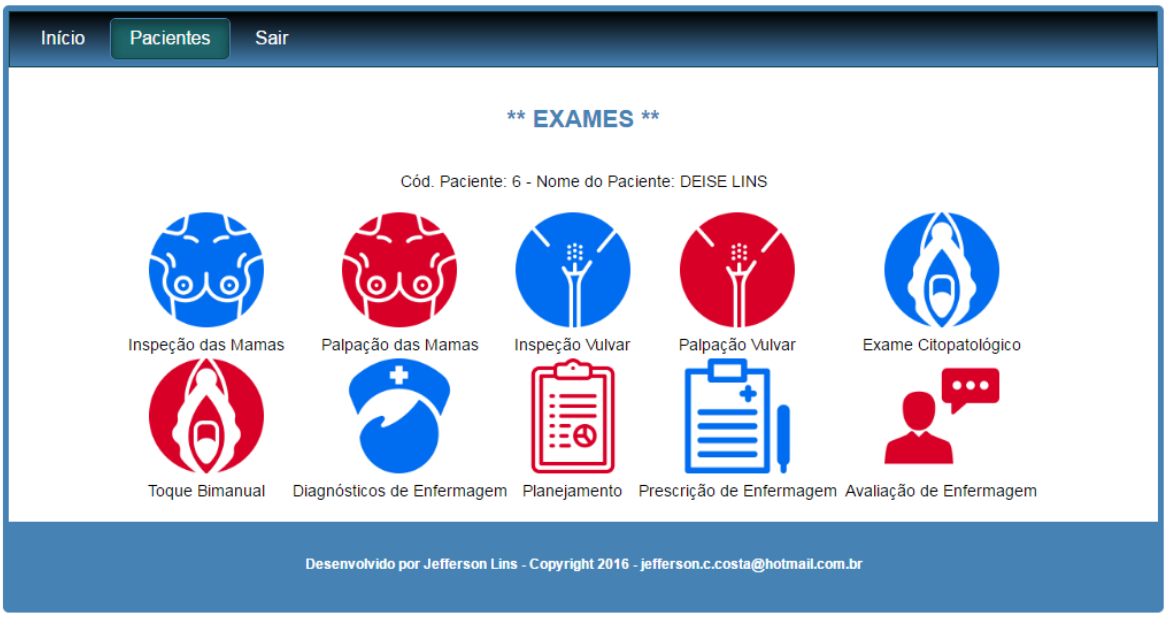
Em “Novo Paciente”, o Enfermeiro Comum pode cadastrar uma paciente, por meio do preenchimento do formulário com os dados da solicitante. O formulário exige alguns dados

obrigatórios, e estes estão sinalizados com um asterisco (*). Caso esses dados obrigatórios não sejam informados, o sistema exibirá uma mensagem exigindo o preenchimento destas informações. Não será permitido o cadastro de paciente com o CPF, RG e número do cartão do Sistema Único de Saúde (SUS) já inseridos no sistema. A paciente só será cadastrada se nenhuma mensagem de erro aparecer.

Após o cadastro, a paciente passa a incorporar a listagem de pacientes, e o Enfermeiro Comum pode realizar o processo SAE para ela. Ao cadastro da paciente, incorporam-se alguns dados do enfermeiro que a cadastrou, tais como: nome do enfermeiro e seu número do Conselho Regional de Enfermagem (COREN). No entanto, todos os enfermeiros têm acesso a todas as pacientes cadastradas.

A Figura 20 mostra a tela exibida quando o Enfermeiro Comum clica sobre o botão “Exames”.

Figura 20 – Tela referente aos exames do processo SAE.



Fonte: Autoria própria.

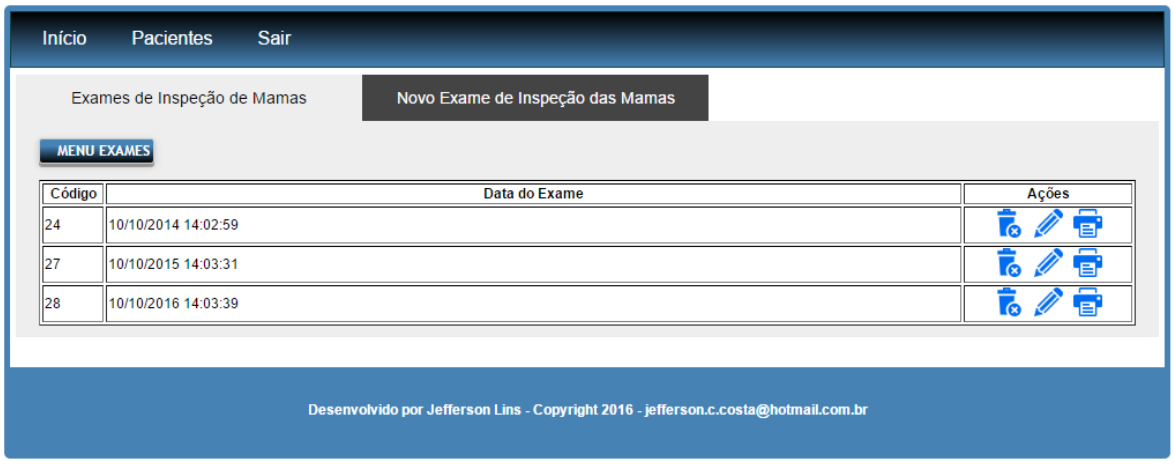
A tela de “Exames” dá acesso a todos os processos da SAE que o Enfermeiro pode realizar para a paciente selecionada na listagem. Ao clicar sobre algum dos ícones exibidos nesta tela, o usuário é enviado para a página de listagem do exame correspondente.

As telas de cada exame estão padronizadas da seguinte forma: inicialmente, a tela de listagem dos exames cadastrados é exibida informando o código e a data da realização do exame, além dos botões de exclusão, edição e impressão do exame; cada tela de exame ainda

fornece a opção para o enfermeiro cadastrar um novo; esta opção encaminha o usuário para outra tela contendo um formulário que requisita os dados do exame a ser cadastrado; acima da listagem de exames, há um botão chamado “MENU EXAMES”, o qual, ao ser clicado, direciona o usuário para a tela de exames.

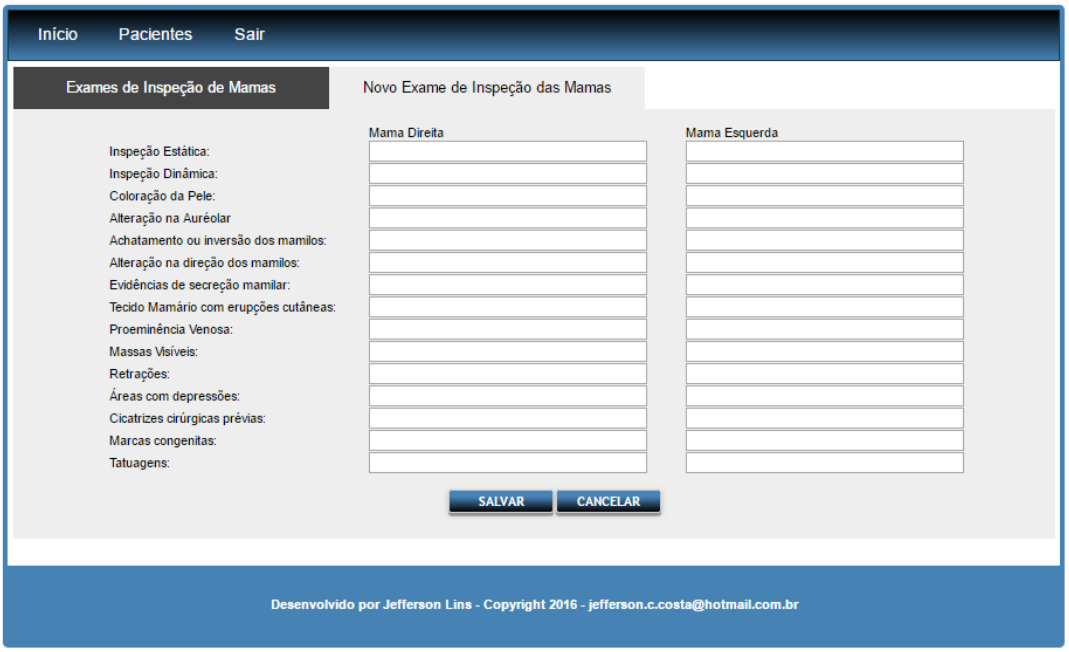
O exame de Inspeção das Mamas será considerado para exemplificar as características já mencionadas. A Figura 21 mostra a tela de listagem dos Exames de Inspeção das Mamas de uma paciente, e a Figura 22 mostra a tela de cadastro do Exame de Inspeção de Mamas.

Figura 21 – Tela de listagem dos Exames de Inspeção das Mamas.



Fonte: Autoria própria.

Figura 22 – Tela de cadastro do Exame de Inspeção de Mamas.



Fonte: Autoria própria.

O mesmo padrão de listagem e cadastro é incorporado aos outros exames; o que se altera são os dados que cada um deles requer.

Estas são as principais funcionalidades disponibilizadas para o usuário Enfermeiro Comum. O usuário Enfermeiro Administrador também tem acesso a estas funcionalidades. No entanto, ele é o responsável por cadastrar novos enfermeiros (Administrador e Comum) no sistema. Será apresentado, no próximo tópico, o módulo do usuário Enfermeiro Administrador.

3.3.1.2 Módulo do usuário Enfermeiro Administrador

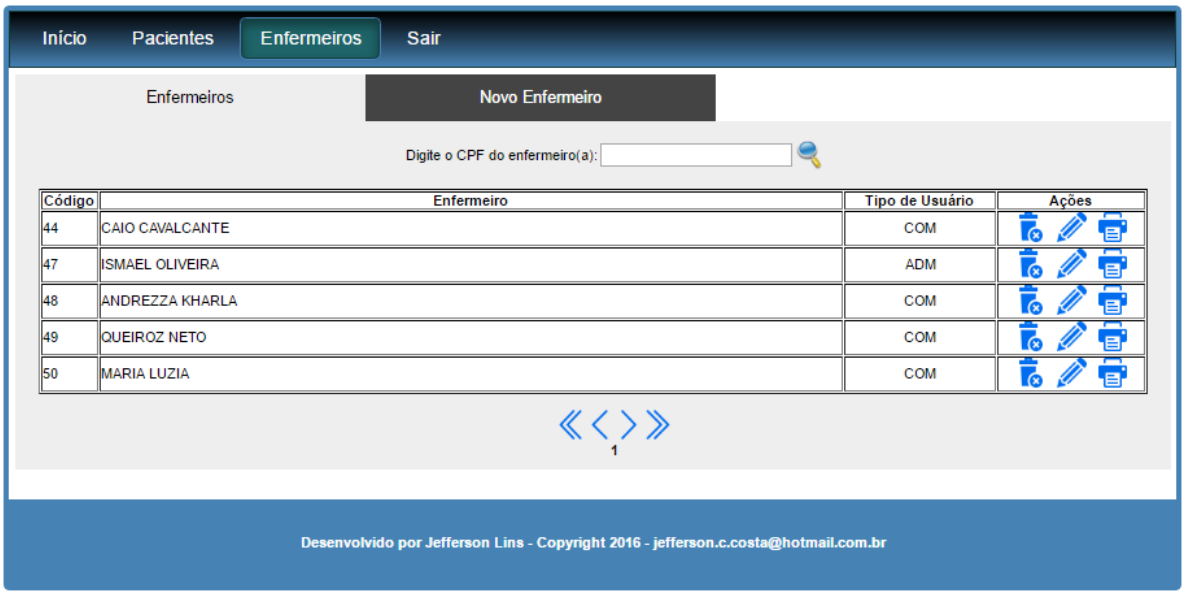
Para o usuário do tipo Enfermeiro Administrador, estão disponíveis as funcionalidades de cadastro de enfermeiro, listagem dos enfermeiros cadastrados, edição de dados de enfermeiro, exclusão do cadastro de enfermeiro, busca de enfermeiro, e todas as funcionalidades presentes no módulo do Enfermeiro Comum. A tela de menu principal do Enfermeiro Administrador, que é exibida após a efetuação do *login*, é mostrada na Figura 23.

Figura 23 – Tela do menu principal do Enfermeiro Administrador.

Fonte: Autoria própria.

O menu principal do Enfermeiro Administrador possui 4 opções: “Início”, “Enfermeiros”, “Pacientes” e “Sair”. A opção “Início” faz o usuário voltar para o menu principal de qualquer página do sistema em que ele estiver. A opção “Sair” faz o usuário sair da aplicação. A Figura 24 mostra o que é exibido quando o Enfermeiro Administrador acessa a opção “Enfermeiros”.

Figura 24 – Listagem dos enfermeiros cadastrados.



Fonte: Autoria própria.

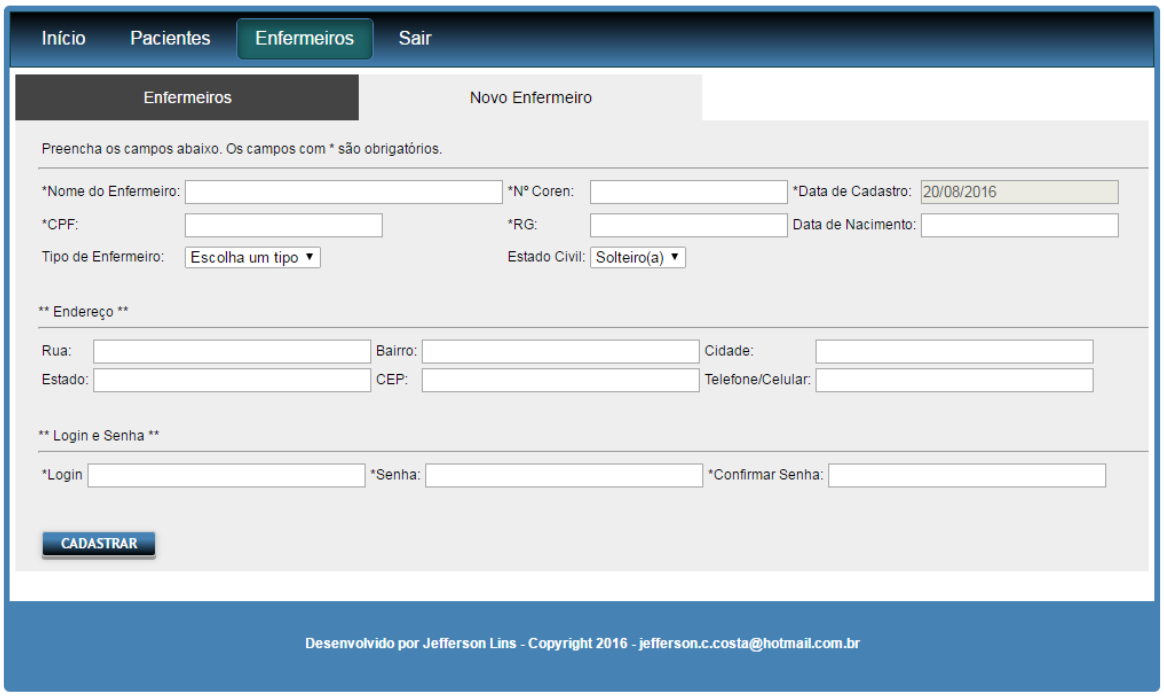
Na opção “Enfermeiros”, o Administrador tem acesso a uma tela que exibe a listagem dos enfermeiros já cadastrados. Essa listagem exibe o código, o nome e o tipo de usuário, além dos botões:

- Excluir, para excluir enfermeiro, caso seja necessário;
- Editar, para editar os dados do enfermeiro, se necessário;
- Imprimir, para realizar a impressão dos dados do enfermeiro.

Para realizar alguma das ações listadas acima, o Administrador apenas precisa clicar sobre o botão correspondente ao que deseja. A tela também oferece a realização de busca de enfermeiro – busca realizada a partir do número do CPF – e a opção para cadastrar um novo enfermeiro.

A Figura 25 mostra a tela cadastro de enfermeiro. Esta tela é exibida quando o Administrador clica sobre a opção “Novo Enfermeiro”.

Figura 25 – Tela de Cadastro de enfermeiro.



Fonte: Autoria própria.

Na opção “Novo Enfermeiro”, o Administrador pode cadastrar um enfermeiro, por meio do preenchimento do formulário com os dados requeridos. Assim como no cadastro de paciente, este formulário exige alguns dados obrigatórios que estão sinalizados com um asterisco (*). Caso esses dados obrigatórios não sejam informados, o sistema exibirá uma mensagem exigindo o seu preenchimento. Não será permitido o cadastro de enfermeiro com o CPF, RG e número COREN já inseridos no sistema. Após o cadastro, o enfermeiro passa a incorporar a listagem de enfermeiros.

Neste capítulo, foram apresentadas as principais telas e funcionalidades do SAE-GIN, tanto para o módulo Enfermeiro Comum como para o módulo Enfermeiro Administrador. No decorrer da realização deste projeto, a importância do SAE-GIN foi reconhecida pelos *stakeholders*, pois a ferramenta é proporciona um gerenciamento de dados dinâmico, organizado e eficaz para o cotidiano dos enfermeiros.

4 COMPARAÇÃO ENTRE OS BANCOS BDR E BDOO

Neste capítulo, será apresentado um estudo de caso elaborado a partir do sistema desta pesquisa. O estudo consiste em um teste de comparação desempenho entre os Bancos de Dados usados. Neste sentido, foram consideradas duas técnicas de persistência baseadas nas seguintes ferramentas: *DB4O*, como banco de dados orientado a objeto, e *Hibernate*, para o mapeamento objeto-relacional com o *MariaDB,* banco de dados relacional. A seguir, descreve-se o ambiente de teste de cada técnica utilizada.

4.1 Ambiente de Teste

Nesta seção, serão expostos os *softwares* e *hardwares* utilizados para a realização dos testes de desempenho dos Bancos de Dados relacionais e orientados a objetos.

4.1.1 *Softwares*

Os *softwares* utilizados foram: *MariaDB, Hibernate*, *DB4O, Glassfish* e *JAVA SE*. Sobre cada um deles, tem-se:

- *MariaDB:* escolhido como SGBD baseado no Modelo Relacional. Este BDR é um dos servidores mais populares. Foi desenvolvido pelos desenvolvedores originais do *MySQL* e, além de ser gratuito, seu código é aberto (MARIADB, 2016). Pode ser encontrado no site: *www.mariadb.org*. A versão utilizada nesta pesquisa foi a 10.1;
- *Hibernate*: para facilitar o mapeamento dos atributos entre uma base tradicional de dados relacionais e o modelo orientado a objeto de uma aplicação, foi utilizado o *framework Hibernate*. Sua principal característica é a transformação das classes em Java para tabelas de dados. Este *framework* gera as chamadas SQL e libera o desenvolvedor do trabalho manual da conversão dos dados resultante, mantendo o programa portável para quaisquer bancos de dados SQL (HIBERNATE, 2016). O site oficial do *Hibernate* é: *www.hibernate.org*. A versão usada nesta pesquisa foi a 4.3.1;
- *DB4O*: escolhido como SGBD baseado no Modelo Orientado a Objetos. Este BDOO pode ser utilizado em aplicações do tipo embarcada, cliente-servidor e *desktop*, e ainda permite armazenar classes Java diretamente no banco, sem a necessidade de utilizar consultas SQL. O site oficial desse BDOO é: *www.DB4O.com.* Como opção alternativa, é

possível, ainda, realizar o *download* do *DB4O* no site: *www.db4o.soft32.com*. A versão usada neste trabalho foi a 8.0;

- *Glassfish*: além das tecnologias citadas anteriormente, foi utilizado um servidor *web* definido como um programa de computador que recebe requisições HTTP (*Hypertext Transfer Protocol*) e retorna uma resposta, também HTTP, para cada uma delas (GLASSFISH, 2016). Esse servidor pode ser encontrado em seu site oficial: *www.oracle.com/technetwork/java/javaee/downloads/ogs-3-1-1-downloads-439803.html.* A versão usada neste trabalho é a 4.1;
- JAVA SE: o *Java Standard Edition* (JAVA SE) é utilizado para desenvolver e executar aplicações JAVA. Ele contém todo o ambiente necessário para desenvolvimento: máquina virtual Java (JVM), compilador Java, APIs do JAVA, dentre outros mecanismos. O JAVA SE deve ser instalado antes das ferramentas citadas anteriormente. O site oficial do JAVA SE é: *www.oracle.com/technetwork/java/javase/downloads*. Para *download* do *Mysql Connector*: *www.dev.mysql.com/downloads/connector/j*. Versão utilizada: 5.1.

4.1.2 *Hardware*

O *hardware* utilizado foi:

- *Notebook* DELL;
- Processador Intel® Core (TM) i3-5005U CPU 2.00GHz;
- 4.00GB de memória RAM;
- *Windows 10 Home Single Language*.

4.2 Metodologia dos testes

No terceiro capítulo, foram apresentados os Diagramas de Classes elaborados para o desenvolvimento do sistema desta pesquisa. A partir destes diagramas, foi explicado que as classes do pacote *Daos* seriam as responsáveis por interagir com o Banco de Dados, realizando funções CRUD (inserção, atualização, deleção e consulta) quando fosse necessário.

Seguindo este raciocínio, as primeiras classes *Daos* implementadas contemplavam as funções CRUD para o BDR utilizado neste trabalho, ou seja, o *MariaDB.* Posteriormente, em um novo pacote chamado *DaosOO*, foram implementadas as classes que interagem com o BDOO escolhido para esta análise, o *DB4O*.

A partir das classes dos pacotes *Daos* e *DaosOO,* foi desenvolvida uma aplicação que tem como objetivo verificar o tempo médio para *N* execuções das funções CRUD destas classes. A aplicação executa uma rotina que realiza automaticamente cada operação CRUD por 100 vezes repetidamente, calculando o tempo médio para cada operação. O tempo de execução para cada operação foi constatado a partir da diferença, em nanossegundos, entre o instante que imediatamente antecedia e o instante posterior à instrução que requisitava o acesso à base de dados. Posteriormente, este tempo é convertido em segundos, para uma melhor análise e interpretação dos resultados obtidos.

Durante a execução da aplicação de teste de desempenho, algumas medidas foram adotadas: os testes foram iniciados com os bancos sem registros; após a realização de cada teste, *Glassfish* foi reiniciado; para realizar os testes, o antivírus foi desativado; os testes foram feitos sem conexão com a internet e nenhum aplicativo aberto.

4.3 Resultados dos testes

Neste tópico, serão apresentados os resultados obtidos a partir da execução de testes de desempenho dos Bancos de Dados utilizados. Estes resultados foram gerados com base nas operações CRUD das entidades Enfermeiro, Paciente e Exame (será considerado apenas o teste do Exame de Avaliação). As tabelas e gráficos apresentados a seguir exibem o tempo médio de 100 execuções consecutivas de cada operação CRUD, efetuadas por três vezes.

A Tabela 2 exibe os resultados relativos às operações CRUD da entidade Enfermeiro efetuadas pelo BDR *MariaDB*.

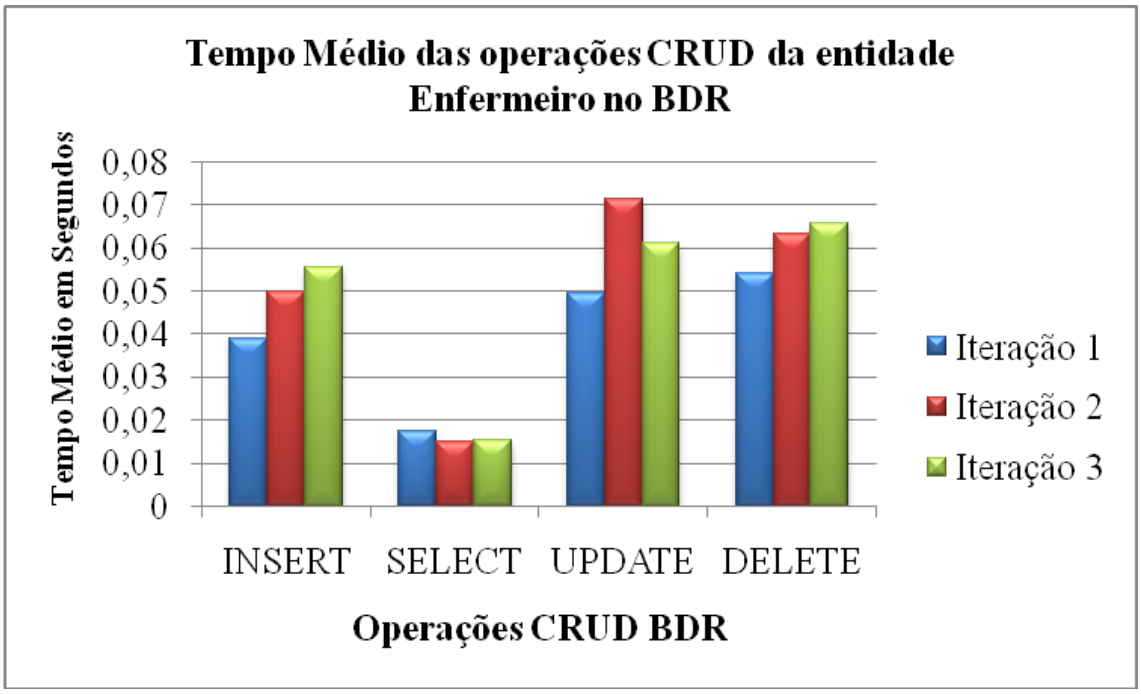
Tabela 2 - Tempo Médio das operações CRUD da entidade Enfermeiro no BDR.

Iterações	Tempo Médio das operações CRUD da entidade Enfermeiro no BDR			
	INSERT	SELECT	UPDATE	DELETE
Iteração 1	0,039041853	0,017376533	0,049438863	0,054326081
Iteração 2	0,049804220	0,014986211	0,071462597	0,063329112
Iteração 3	0,055643410	0,015199064	0,061323544	0,065991976

Fonte: Fonte:

O Gráfico 1 mostra os resultados relativos às iterações das operações CRUD da entidade Enfermeiro efetuadas pelo BDR *MariaDB*.

Gráfico 1 – Tempo Médio das operações CRUD da entidade Enfermeiro no BDR.



Fonte: Autoria própria.

A Tabela 3 apresenta os resultados relativos às operações CRUD da entidade Enfermeiro efetuadas pelo BDOO *DB4O*.

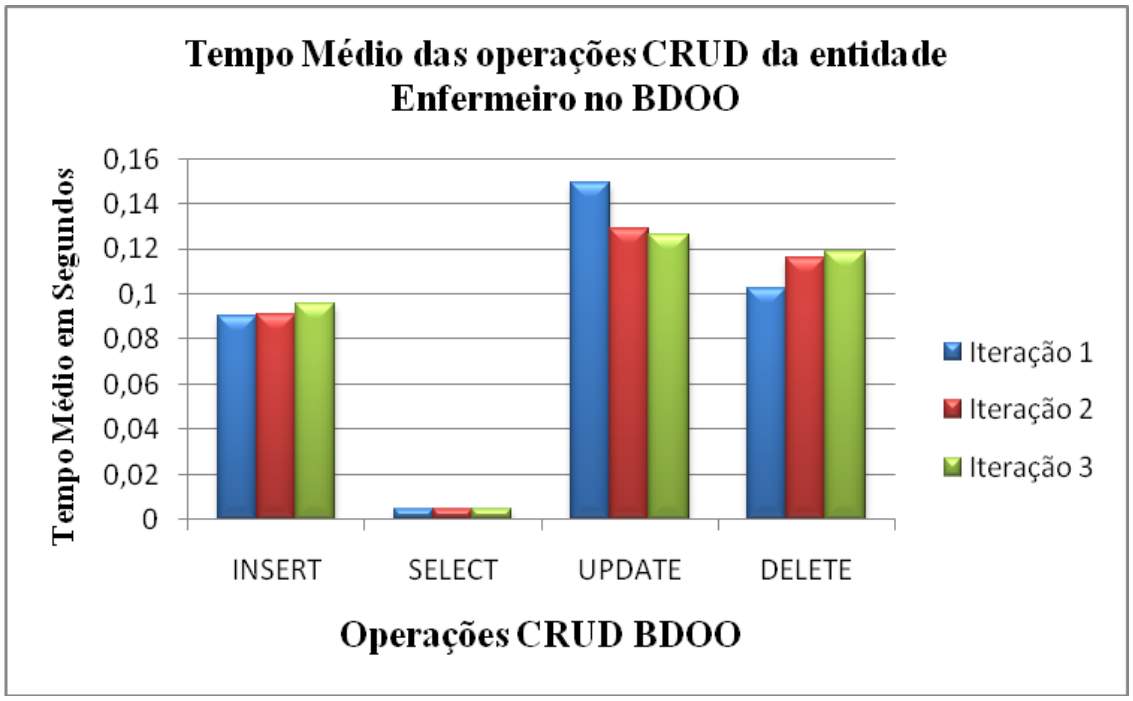
Tabela 3 - Tempo Médio das operações CRUD da entidade Enfermeiro no BDOO.

Iterações	Tempo Médio das operações CRUD da entidade Enfermeiro no BDOO			
	INSERT	SELECT	UPDATE	DELETE
Iteração 1	0,090075582	0,004711298	0,149226779	0,102661617
Iteração 2	0,091168395	0,004725421	0,129218764	0,115886260
Iteração 3	0,095849834	0,004573261	0,126105261	0,118779903

Fonte: Autoria própria.

O Gráfico 2 expõe os resultados relativos às iterações das operações CRUD da entidade Enfermeiro efetuadas pelo BDOO *DB4O*.

Gráfico 2 – Tempo Médio das operações CRUD da entidade Enfermeiro no BDR.



Fonte: Autoria própria.

A Tabela 4 apresenta um comparativo entre o tempo médio das três iterações das operações CRUD da entidade Enfermeiro para cada Banco de Dados.

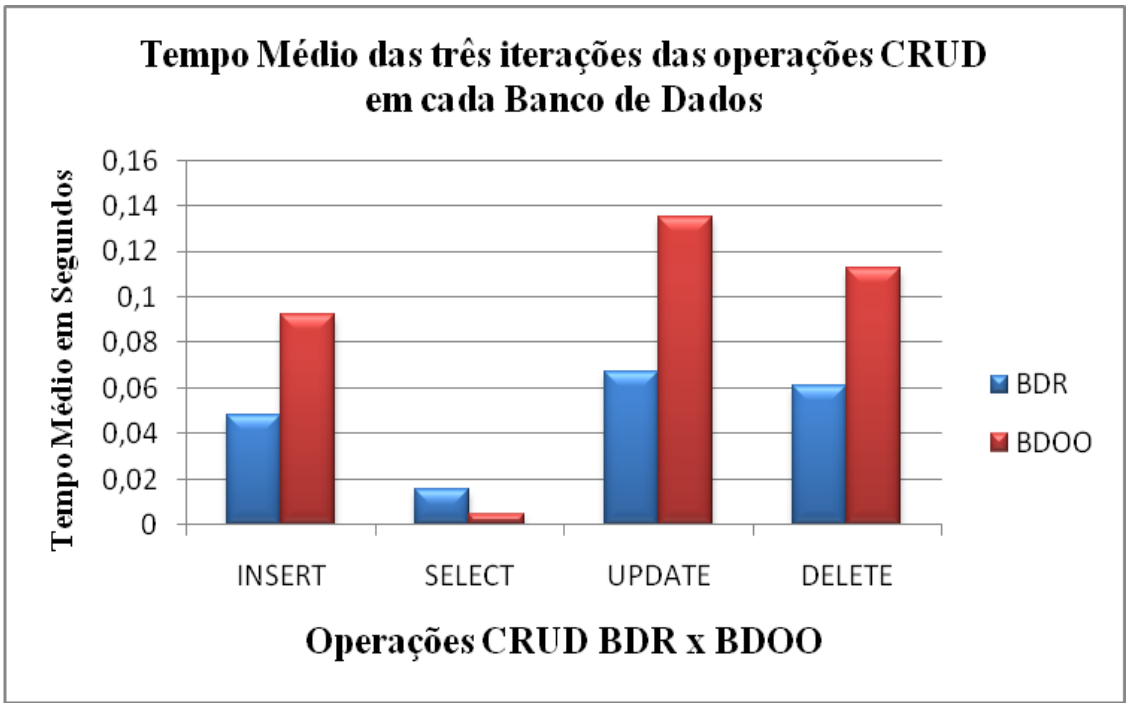
Tabela 4 - Tempo Médio das três iterações das operações CRUD da entidade Enfermeiro em cada Banco de Dados.

Tempo Médio	Tempo Médio das três iterações das operações CRUD em cada Banco de Dados			
	INSERT	SELECT	UPDATE	DELETE
BDR	0,048163161	0,015853936	0,06741668	0,061215723
BDOO	0,092364603	0,004669993	0,134850268	0,112442593

Fonte: Autoria própria.

O Gráfico 3 expõe um comparativo entre o tempo médio das três iterações das operações CRUD para cada Banco de Dados em relação à entidade Enfermeiro.

Gráfico 3 – Tempo Médio das três iterações das operações CRUD da entidade Enfermeiro em cada Banco de Dados.



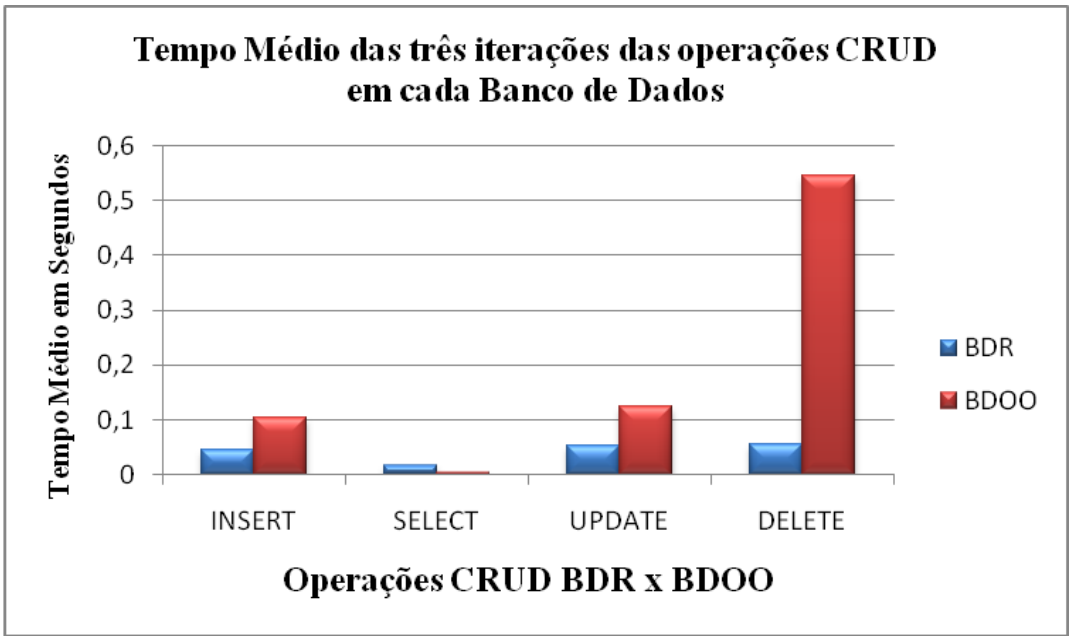
Fonte: Autoria própria.

A partir da análise dos gráficos, pode-se perceber que o BDOO tem o menor tempo apenas na operação de consulta, ou seja, o *DB4O* possui melhor desempenho somente na operação de busca. Já o BDR se mostrou mais eficiente nas operações de inserção, atualização e deleção, apresentando um desempenho superior ao BDOO para estes três procedimentos.

Para a demonstração de que o BDR possui um melhor desempenho nas funções mencionadas anteriormente, nos próximos gráficos, será exibido um comparativo entre o tempo médio de três iterações das operações CRUD para cada Banco de Dados: primeiramente, em relação à entidade Paciente; em seguida, para Exame de Avaliação.

O Gráfico 4 apresenta uma análise entre o tempo médio das três iterações das operações CRUD para cada Banco de Dados em relação à entidade Paciente.

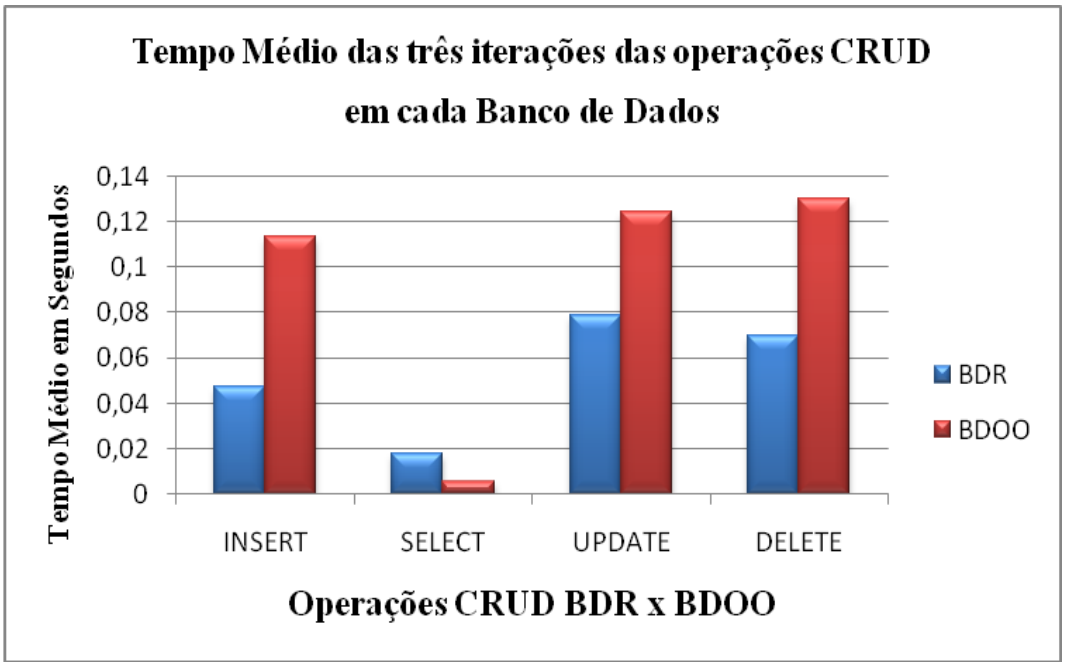
Gráfico 4 – Tempo Médio das três iterações das operações CRUD da entidade Paciente em cada Banco de Dados.



Fonte: Autoria própria.

O Gráfico 5 mostra um comparativo entre o tempo médio das três iterações das operações CRUD para cada Banco de Dados em relação à entidade Exame de Avaliação.

Gráfico 5 – Tempo Médio das três iterações das operações CRUD da Entidade Exame de Avaliação em cada Banco de Dados.



Fonte: Autoria própria.

A partir destas análises, constata-se que, para o sistema desenvolvido nesta pesquisa, o Banco de Dados mais adequado é o BDR *MariaDB* (mapeado pelo *framework Hibernate*), em virtude de seu melhor desempenho e estabilidade na execução das operações de inserção, atualização e deleção das entidades envolvidas.

5 CONCLUSÕES

Para que um sistema seja funcional, cumpra os seus requisitos e tenha qualidade, é fundamental que siga o processo de desenvolvimento mais adequado. As atividades do processo escolhido para o desenvolvimento da aplicação desta pesquisa, utilizando a metodologia do Modelo Cascata, foram realizadas com o objetivo de satisfazer as necessidades do cliente. Todas as atividades tiveram grande importância na criação do SAE-GIN, pois, por meio dos recursos que elas forneceram, foi possível construir uma aplicação capaz de auxiliar as tarefas realizadas pelos enfermeiros da Clínica Escola da UnP.

O SAE-GIN contribuirá para, possivelmente, extinguir as deficiências encontradas nas atividades dos enfermeiros da clínica, desde a necessidade de gerenciar o controle de dados relacionados ao cadastro dos enfermeiros que utilizarão o *software*, até a administração das informações das pacientes e os processos da SAE realizados para elas.

O objetivo de desenvolver um sistema que auxiliasse nas atividades dos enfermeiros da Clínica Escola da UnP foi alcançado. O *software* proporcionará aos enfermeiros uma melhor organização dos dados, concentrando-os em um único sistema, que pode ser acessado de qualquer lugar com acesso à *Internet*.

Esta pesquisa também constatou que, para o desenvolvimento do SAE-GIN, entre o Banco de Dados Relacional *MariaDB* e o Banco de Dados Orientado a Objetos *DB4O*, o que possui melhor desempenho em relação à maioria das operações realizadas é o BDR *MariaDB.* De acordo com as análises realizadas, concluiu-se que, em termos de desempenho, o BDR, mapeado pelo *framework Hibernate*, é melhor que o BDOO nas funções de inserção, alteração e deleção de dados;

O BDOO obteve melhor desempenho apenas na operação de consulta, apontando um tempo médio superior ao BDR em relação a esta função. Porém, ele apresentou certa desvantagem em relação ao tempo médio para as outras operações, que, em alguns casos, foram realizadas com quase o dobro de tempo, comparando-se ao BDR.

Em virtude de o SAE-GIN ser uma aplicação web, sugere-se, como trabalho futuro, realizar uma nova investigação, fazendo uso real da arquitetura cliente-servidor com o intuito de simular um cenário mais realista. Entende-se que a realização desta nova pesquisa forneceria informações ainda mais aprofundadas para este estudo, possibilitando otimizações e aprimoramentos necessários ao sistema. Sabe-se, também, que existem outros BDOOs, sendo assim, ainda pode ser feito um novo estudo de caso a partir de outros bancos desse mesmo paradigma.

REFERÊNCIAS BIBLIOGRÁFICAS

ALFARO-LEFEVRE, R. **Aplicação do processo de enfermagem: promoção do cuidado colaborativo.** 5.ed. Porto Alegra: ARTMED, 2005.

AMBLER, S.W. **Modelagem Ágil: Práticas eficazes para a Programação eXtrema e o Processo Unificado**. Bookman. 2004.

BASSI FILHO, Dairton Luiz. **Experiências com desenvolvimento ágil**. 2008. 170 f. Dissertação (Mestrado em Ciência da Computação) Universidade de São Paulo, São Paulo - SP, 2008.

BECK, K. **Embracing change with extreme programming. Computer**, vol.32, no.10, pp.70-77. 1999.

BECK, K. **Extreme Programming Explained: Embrace Change**. Addison-Wesley Professional, 2000.

BECK, K. **Programação extrema explicada: acolha as mudanças**. Bookman. 2004.

BENEDICTO, M. de; BEZERRA, A.; BOSCARIOLI, C.; DELMIRO, G; **Uma reflexão sobre Banco de Dados Orientados a Objetos. In: IV CONGED Congresso de Tecnologias para Gestão de Dados do Cone Sul**, 23 e 25, 2006, Ponta Grossa – PR. Disponível em: < http://conged.deinfo.uepg.br/artigo4.pdf>. Acesso em: 16 set. 2016.

BEZERRA, E. **Princípios de Análise e Projeto de Sistemas com UML**. Campus, 2007.

CHAUDRI, A. B; ZICARI, R. (2001). **Succeeding with Object Databases: A Practical Look at Today's Implementations with Java and XML**. Willey Computer Publishing, 1st edition.

CHAVES, L. D. **Sistematização da Assistência de Enfermagem, considerações teóricas e aplicabilidade**. 2. Ed. São Paulo: Martinari; 2013.

COCKBURN, A.; HIGHSMITH, J. **Agile software development: the people factor**. Computer, vol.34, no.11, pp.131-133, Nov 2001.

DB4O. DB4Objects. Disponível em: <http://www.DB4O.com/portugues/>. Acesso em: 20 Out. 2016.

ELMASRI, Ramez; NAVATHE, Ehamkant. **Sistemas de Banco de Dados**. 6. ed. Traduzido por Daniel Vieira. São Paulo: Pearson Addison Wesley, 2011.

FARIA, T. **Java EE 7 com JSF. PrimeFaces e CDI.** AlgaWorks, 2013.

FERREIRA, Julyana Lima. Desenvolvimento de um sistema de apoio à pesca (SISAPE). Monografia (Graduação em Ciência da Computação), Departamento de Ciências Exatas e Naturais, Universidade Federal Rural do Semi-árido, Mossoró-RN, 2013.

FIGUEIREDO, André Luís Gouvêa de. **ECO : um ecossistema para o desenvolvimento ágil de sistemas web**. 2006. 137 f. Dissertação (Mestrado em Ciência da Computação) Universidade de São Paulo, São Carlos - SP, 2005.

GARLAN, D. **Software architecture: a roadmap**. In: Proceedings of The Conference on The Future of Software Engineering, 2000.

GARLAN, D., PERRY, D. **Introduction to the Special Issue on Software Architecture**. In: IEEE Transactions on Software Engineering, v. 21, April 1995.

GLASSFISH. Site oficial. Disponível em: <http://glassfish.java.net/>. Acesso em: 20 Out. 2016.

FRASSON, R. **Java para web na prática.** Criciúma, 2014.

JACOBSON, Ivar; BOOCH, Grady; e RUMBAUGH, James. **The Unified Software Development Process**. Addison-Wesley, 1999.

HEUSER, C. A. **Projeto de Banco de Dados**. Editora Bookman, 6a Edição, 2009.

HIBERNATE. Site oficial. Disponível em: <http://www.hibernate.org/>. Acesso em: 20 Out. 2016.

MACHADO, F. N. R. **Banco de Dados Projeto e Implementação**. Editora Érica, 2ª Edição, 2008.

MANIFESTO for agile *software* development. Feb. 2001, Disponível em <www.agilemanifesto.org>. Acesso em: 16 set. 2016.

MARIADB. Site oficial. Disponível em: <*https://mariadb.org/*>. Acesso em: 20 Out. 2016.

NEVES, R.S. **Sistematização da assistência de enfermagem em unidade de reabilitação segundo o modelo conceitual de horta**. Rev. bras. enferm. Disponível em <http://www.scielo.br/pdf/reben/v59n4/a16v59n4.pdf>. 2006. Acesso em: 26/09/2016.

OLIVEIRA, Manoel Marisergio Alves de. **Um estudo comparativo de desempenho entre técnicas de persistências de dados, no contexto de uma aplicação web**. Monografia (Graduação em Ciência da Computação), Departamento de Ciências Exatas e Naturais, Universidade Federal Rural do Semi-árido, Mossoró-RN, 2013.

PRESSMAN, R.S. **Engenharia de Software**. Editora McGraw-Hill, 6ª Edição, 2006.

PMBOK, **Project Managment Body of Knowledge Guide**. Project Managment Institute, Inc (PMI), 2008.

SILVA, J. **Padrões e Métodos Ágeis agilidade no processo de desenvolvimento de software. Anais**. 5th Latin American Conference on Pattern Languages of Programming. Campos do Jordão, Brasil. 16 a 19 de agosto, 2005.

SOFT32. Soft32. Disponível em: <http://db4o.soft32.com//>. Acesso em: 20 Out. 2016.

SOUZA NETO, Oscar Nogueira de. **Análise comparativa das metodologias de desenvolvimento de softwares tradicionais e ágeis**. 2004. 74 f. Monografia (Graduação em Ciência da Computação), Centro de Ciências Exatas e Tecnologia, Universidade da Amazônia, Belém-PA, 2004.

SOMMERVILLE, I. **Engenharia de Software**. Pearson Addison Wesley. 2007.

THIOLLENT, M. Pesquisa-Ação nas Organizações. São Paulo: Atlas, 1997.

APÊNDICE A – ESPECIFICAÇÃO DE CASOS DE USO

Tabela 5 - Caso de uso Cadastrar Enfermeiro.

Nome do caso de uso	Cadastrar Enfermeiro
Objetivo	O usuário cadastra o enfermeiro de acordo com as informações que são passadas.
Ator principal	Enfermeiro Administrador
Pré-condições	O usuário deve ter efetuado o *login* e ainda precisa ser devidamente identificado como sendo do tipo Enfermeiro Administrador.
Fluxo Principal	**P1.1.** O usuário escolhe a opção "Enfermeiros". **P1.2.** O sistema mostra uma tela que possui duas guias que podem ser escolhidas pelo usuário, sendo primeiramente exibida a guia que possui a listagem dos enfermeiros cadastrados (intitulada Enfermeiros). A outra guia (intitulada Novo Enfermeiro) apresenta o formulário de cadastro de enfermeiro. **P1.3.** O usuário clica sobre a guia intitulada Novo Enfermeiro. **P1.4.** O sistema mostra um formulário de Cadastro de Enfermeiro. **P1.5.** O usuário preenche os dados do enfermeiro no formulário e ao término clica no botão "CADASTRAR". **<E1, E2>** **P1.6.** O sistema exibe uma mensagem informando se o enfermeiro foi cadastrado.
Fluxo de Exceção	**E1.** Caso o usuário não preencha todos os campos necessários do formulário, os seguintes passos serão executados: **E1.1.** O sistema exibe uma mensagem informando quais os campos obrigatórios não foram preenchidos. **E1.2.** O sistema retorna para o passo 1.2 do fluxo Principal. **E2.** Caso o usuário informe o número do COREN, CPF e RG já inseridos no sistema, os seguintes passos serão

	executados: **E2.1.** O sistema exibe uma mensagem informando qual dado já está inserido no sistema. **E2.2.** O sistema retorna para o passo 1.2 do fluxo Principal.
Pós-condições	O cadastro do enfermeiro deve ser efetuado com sucesso.

Fonte: Autoria própria.

Tabela 6 - Caso de uso Editar Enfermeiro.

Nome do caso de uso	Editar Enfermeiro
Objetivo	O usuário edita as informações de cadastro de um enfermeiro quando necessário.
Ator principal	Enfermeiro Administrador
Pré-condições	O usuário deve ter efetuado o *login* e ainda precisa ser devidamente identificado como sendo do tipo Enfermeiro Administrador. O enfermeiro escolhido pelo usuário deve estar previamente cadastrado.
Fluxo Principal	**P1.1.** O usuário escolhe a opção "Enfermeiros". **P1.2.** O sistema mostra uma tela que possui duas guias que podem ser escolhidas pelo usuário, sendo primeiramente exibida a que possui a listagem dos enfermeiros cadastrados (intitulada Enfermeiros). **P1.3.** O usuário escolhe o enfermeiro que pretende editar e clica no botão "Editar". **P1.4.** O sistema exibe o formulário para visualizar todas as informações do enfermeiro e também permitir a edição das mesmas, caso seja necessário. **P1.5.** O usuário realiza as atualizações necessárias e ao término clica no botão "ATUALIZAR". **<E1>** **P1.6.** O sistema exibe uma mensagem perguntando se o usuário realmente deseja fazer a atualização, oferecendo as opções SIM ou NÃO. **P1.7**. O usuário clica sobre a opção SIM. **P1.8.** O sistema exibe uma mensagem informando que os dados do enfermeiro foram atualizados.

Fluxo de Exceção	**E1.** Caso o usuário não preencha todos os campos necessários do formulário, os seguintes passos serão executados: **E1.1** O sistema exibe uma mensagem informando os campos que não foram preenchidos. **E1.2** O sistema retorna para o passo 1.4 do Fluxo Principal. **E2.** Caso o usuário informe um *Login* já inserido no sistema, os seguintes passos serão executados: **E2.1.** O sistema exibe uma mensagem informando que o dado já está inserido no sistema. **E2.2** O sistema retorna para o passo 1.4 do Fluxo Principal.
Pós-condições	Ao término desta ação, a edição do enfermeiro deve ter sido efetuada com sucesso.

Fonte: Autoria própria.

Tabela 7 - Caso de Uso Excluir Enfermeiro.

Nome do caso de uso	Excluir Enfermeiro
Objetivo	O usuário exclui o enfermeiro, caso necessário.
Ator principal	Enfermeiro Administrador
Pré-condições	O usuário deve ter efetuado o *login* e ainda precisa ser devidamente identificado como sendo do tipo Enfermeiro Administrador. O enfermeiro escolhido pelo usuário deve estar previamente cadastrado.
Fluxo Principal	**P1.1.** O usuário escolhe a opção "Enfermeiros". **P1.2.** O sistema mostra uma tela que possui duas guias que podem ser escolhidas pelo usuário, sendo primeiramente exibida a que possui a listagem dos enfermeiros cadastrados (intitulada Enfermeiros). **P1.3.** O usuário escolhe o enfermeiro que pretende excluir e clica no botão "Excluir". **P1.4.** O sistema exibe uma mensagem perguntando se o usuário realmente deseja fazer a exclusão, oferecendo as opções SIM ou NÃO. **P1.5**. O usuário clica sobre a opção SIM. **P1.6.** O sistema exibe uma mensagem informando que o enfermeiro foi

	excluído.
Fluxo de Exceção	Não existe Fluxo de Exceção.
Pós-condições	Ao término desta ação, a exclusão do enfermeiro deve ter sido efetuada com sucesso.

Fonte: Autoria própria.

Tabela 8 - Caso de Uso Listar Enfermeiros.

Nome do caso de uso	Listar Enfermeiros
Objetivo	O usuário lista os enfermeiros já cadastrados no sistema.
Ator principal	Enfermeiro Administrador
Pré-condições	O usuário deve ter efetuado o *login* e ainda precisa ser devidamente identificado como sendo do tipo Enfermeiro Administrador.
Fluxo Principal	**P1.1.** O usuário escolhe a opção "Enfermeiros". **P1.2.** O sistema mostra uma tela que possui duas guias que podem ser escolhidas pelo usuário, sendo primeiramente exibida a que possui a listagem dos enfermeiros cadastrados (intitulada Enfermeiros).
Fluxo de Exceção	Não existe Fluxo de Exceção.
Pós-condições	Ao término desta ação, o usuário poderá ter acesso a todos os enfermeiros cadastrados no sistema.

Fonte: Autoria própria.

Tabela 9 - Caso de Uso Buscar Enfermeiro.

Nome do caso de uso	Buscar Enfermeiro
Objetivo	O usuário busca por um enfermeiro já cadastrado no sistema.
Ator principal	Enfermeiro Administrador
Pré-condições	O usuário deve ter efetuado o *login* e ainda precisa ser devidamente identificado como sendo do tipo Enfermeiro Administrador.
Fluxo Principal	**P1.1.** O usuário escolhe a opção "Enfermeiros". **P1.2.** O sistema mostra uma tela que

	possui duas guias que podem ser escolhidas pelo usuário, sendo primeiramente exibida a que possui a listagem dos enfermeiros cadastrados (intitulada Enfermeiros). Esta tela também mostra um campo de busca de enfermeiro a partir do CPF. **P1.3**. O usuário digita o número do enfermeiro que deseja buscar. <**E1**> **P1.4**. O sistema exibe na listagem o enfermeiro encontrado.
Fluxo de Exceção	**<E.1>** O enfermeiro não consta no banco de dados do sistema. **<E.1.1>** O sistema mostra uma mensagem "ENFERMEIRO NÃO ENCONTRADO(A)!".
Pós-condições	Ao término desta ação, o usuário terá acesso às informações do enfermeiro – caso ele esteja cadastrado.

Fonte: Autoria própria.

Tabela 10 - Caso de uso Cadastrar Paciente.

Nome do caso de uso	Cadastrar Paciente
Objetivo	O usuário cadastra a paciente de acordo com as informações que são passadas.
Ator principal	Enfermeiro Administrador ou Comum
Pré-condições	O usuário deve ter efetuado o *login* no sistema.
Fluxo Principal	**P1.1.** O usuário escolhe a opção "Pacientes". **P1.2.** O sistema mostra uma tela que possui duas guias que podem ser escolhidas pelo usuário, sendo primeiramente exibida a guia que possui a listagem das pacientes cadastradas (intitulada Pacientes). A outra guia (intitulada Novo Paciente) apresenta o formulário de cadastro de paciente. **P1.3.** O usuário clica sobre a guia intitulada Nova Enfermeiro. **P1.4.** O sistema mostra um formulário de Cadastro de Paciente. **P1.5.** O usuário preenche os dados da paciente no formulário e, ao término, clica no botão "CADASTRAR". **<E1, E2>** **P1.6.** O sistema exibe uma mensagem informando se a paciente foi cadastrada.

Fluxo de Exceção	**E1.** Caso o usuário não preencha todos os campos necessários do formulário, os seguintes passos serão executados: **E1.1.** O sistema exibe uma mensagem informando quais os campos obrigatórios não foram preenchidos. **E1.2.** O sistema retorna para o passo 1.2 do fluxo Principal. **E2.** Caso o usuário informe CPF, RG e cartão do SUS já inseridos no sistema, os seguintes passos serão executados: **E2.1.** O sistema exibe uma mensagem informando qual dado já está inserido no sistema. **E2.2.** O sistema retorna para o passo 1.2 do fluxo Principal.
Pós-condições	O cadastro da paciente deve ser efetuado com sucesso.

Fonte: Autoria própria.

Tabela 11 - Caso de uso Editar Paciente.

Nome do caso de uso	Editar Paciente
Objetivo	O usuário edita as informações de cadastro de uma paciente quando necessário.
Ator principal	Enfermeiro Administrador ou Comum
Pré-condições	O usuário deve ter efetuado o *login* e ainda precisa ser devidamente identificado como sendo do tipo Enfermeiro Administrador ou Comum. A paciente escolhida pelo usuário deve estar previamente cadastrada.
Fluxo Principal	**P1.1.** O usuário escolhe a opção “Pacientes”. **P1.2.** O sistema mostra uma tela que possui duas guias que podem ser escolhidas pelo usuário, sendo primeiramente exibida a que possui a listagem das pacientes cadastrados (intitulada Pacientes). **P1.3.** O usuário escolhe a paciente que pretende editar e clica no botão “Editar”. **P1.4.** O sistema exibe o formulário para visualizar todas as informações da paciente e também permitir a edição das mesmas, caso seja necessário. **P1.5.** O usuário realiza as atualizações necessárias e ao término clica no botão

	"ATUALIZAR". **<E1>** **P1.6.** O sistema exibe uma mensagem perguntando se o usuário realmente deseja fazer a atualização, oferecendo as opções SIM ou NÃO. **P1.7**. O usuário clica sobre a opção SIM. **P1.8.** O sistema exibe uma mensagem informando que os dados da paciente foram atualizados.
Fluxo de Exceção	**E1.** Caso o usuário não preencha todos os campos necessários do formulário, os seguintes passos serão executados: **E1.1** O sistema exibe uma mensagem informando os campos que não foram preenchidos. **E1.2** O sistema retorna para o passo 1.4 do Fluxo Principal. **E2.** Caso o usuário informe o número do cartão do SUS já inserido no sistema, os seguintes passos serão executados: **E2.1.** O sistema exibe uma mensagem informando que o dado já está inserido no sistema. **E2.2** O sistema retorna para o passo 1.4 do Fluxo Principal.
Pós-condições	Ao término desta ação, a edição da paciente deve ter sido efetuada com sucesso.

Fonte: Autoria própria.

Tabela 12 - Caso de Uso Excluir Paciente.

Nome do caso de uso	Excluir Paciente
Objetivo	O usuário exclui a paciente, caso necessário.
Ator principal	Enfermeiro Administrador ou Comum.
Pré-condições	O usuário deve ter efetuado o *login*. A paciente escolhida pelo usuário deve estar previamente cadastrada.
Fluxo Principal	**P1.1.** O usuário escolhe a opção "Pacientes". **P1.2.** O sistema mostra uma tela que possui duas guias que podem ser escolhidas pelo usuário, sendo primeiramente exibida a que possui a listagem das pacientes cadastrados (intitulada Pacientes). **P1.3.** O usuário escolhe a paciente que pretende excluir e clica no botão "Excluir".

	P1.4. O sistema exibe uma mensagem perguntando se o usuário realmente deseja fazer a exclusão, oferecendo as opções SIM ou NÃO. **P1.5**. O usuário clica sobre a opção SIM. **P1.6.** O sistema exibe uma mensagem informando que a paciente foi excluída.
Fluxo de Exceção	Não existe Fluxo de Exceção.
Pós-condições	Ao término desta ação, a exclusão da paciente deve ter sido efetuada com sucesso.

Fonte: Autoria própria.

Tabela 13 - Caso de Uso Listar Pacientes.

Nome do caso de uso	Listar Pacientes
Objetivo	O usuário lista as pacientes já cadastradas no sistema.
Ator principal	Enfermeiro Administrador ou Comum.
Pré-condições	O usuário deve ter efetuado o *login*.
Fluxo Principal	**P1.1.** O usuário escolhe a opção "Pacientes". **P1.2.** O sistema mostra uma tela que possui duas guias que podem ser escolhidas pelo usuário, sendo primeiramente exibida a que possui a listagem das pacientes cadastradas (intitulada Pacientes).
Fluxo de Exceção	Não existe Fluxo de Exceção.
Pós-condições	Ao término desta ação, o usuário poderá ter acesso a todas as pacientes cadastradas no sistema.

Fonte: Autoria própria.

Tabela 14 - Caso de Uso Buscar Paciente.

Nome do caso de uso	Buscar Enfermeiro
Objetivo	O usuário busca por uma paciente já cadastrada no sistema.
Ator principal	Enfermeiro Administrador ou Comum.
Pré-condições	O usuário deve ter efetuado o *login*.
Fluxo Principal	**P1.1.** O usuário escolhe a opção "Pacientes". **P1.2.** O sistema mostra uma tela que possui duas guias que podem ser escolhidas pelo usuário, sendo primeiramente exibida a que possui a

	listagem das pacientes cadastrados (intitulada Pacientes). Esta tela também mostra um campo de busca de paciente a partir do CPF, nome ou RG. **P1.3**. O usuário digita a informação da paciente que deseja buscar, conforme do tipo da escolha da busca. <**E1**> **P1.4**. O sistema exibe na listagem a paciente encontrada.
Fluxo de Exceção	**<E.1>** A paciente não consta no banco de dados do sistema. **<E.1.1>** O sistema mostra uma mensagem "PACIENTE NÃO ENCONTRADO!".
Pós-condições	Ao término desta ação, o usuário terá acesso às informações da paciente – caso ela esteja cadastrada.

Fonte: Autoria própria.

Tabela 15 - Caso de uso Cadastrar Exame.

Nome do caso de uso	Cadastrar Exame
Objetivo	O usuário cadastra o exame de acordo com as informações que são passadas.
Ator principal	Enfermeiro Administrador ou Comum.
Pré-condições	O usuário deve ter efetuado o *login* no sistema. A paciente do exame deve estar previamente cadastrada no sistema.
Fluxo Principal	**P1.1.** O usuário escolhe a opção "Pacientes". **P1.2.** O sistema mostra uma tela que possui duas guias que podem ser escolhidas pelo usuário, sendo primeiramente exibida a guia que possui a listagem das pacientes cadastradas (intitulada Pacientes). **P1.3.** O usuário escolher a paciente e clica sobre o botão "Exames". **P1.4.** O sistema mostra um menu de opções de Exames **P1.5.** O usuário escolhe o menu do exame que deseja cadastrar e clica sobre ele. **P1.6.** O sistema mostra uma tela que possui duas guias que podem ser escolhidas pelo usuário, sendo primeiramente exibida a guia (intitulada pelo nome do exame escolhido) que possui a listagem do tipo do exame. A

	outra guia (Novo "Nome do Exame") apresenta o formulário de cadastro do exame escolhido. **P1.7.** O usuário clica sobre guia de novo exame; **P1.8.** O sistema exibe o formulário do exame a ser preenchido. **P1.9.** O usuário preenche o formulário com as informações exigidas e, ao término do preenchimento, clica sobre o botão "CADASTRAR". **P1.10.** O sistema exibe uma mensagem informando se o exame foi cadastrado.
Fluxo de Exceção	Não existe Fluxo de Exceção.
Pós-condições	O cadastro da exame deve ser efetuado com sucesso.

Fonte: Autoria própria.

Tabela 16 - Caso de uso Editar Exame.

Nome do caso de uso	Editar Exame
Objetivo	O usuário edita as informações de cadastro de um exame quando necessário.
Ator principal	Enfermeiro Administrador ou Comum
Pré-condições	O usuário deve ter efetuado o *login*. A paciente escolhida pelo usuário deve estar previamente cadastrada.
Fluxo Principal	**P1.1.** O usuário escolhe a opção "Pacientes". **P1.2.** O sistema mostra uma tela que possui duas guias que podem ser escolhidas pelo usuário, sendo primeiramente exibida a guia que possui a listagem das pacientes cadastradas (intitulada Pacientes). **P1.3.** O usuário escolher a paciente e clica sobre o botão "Exames". **P1.4.** O sistema mostra um menu de opções de Exames **P1.5.** O usuário escolhe o menu do tipo de exame que deseja editar e clica sobre ele. **P1.6.** O sistema mostra uma tela que possui duas guias que podem ser escolhidas pelo usuário, sendo primeiramente exibida a guia (intitulada pelo nome do exame escolhido) que possui a listagem do tipo do exame.

	P1.7. O usuário escolhe o exame que deseja fazer a edição e clica no botão “Editar”. **P1.8.** O sistema exibe o formulário do exame com todas as suas informações. **P1.9.** O usuário atualiza as informações necessárias e clica sobre o botão “ATUALIZAR”. **P1.10.** O sistema exibe uma mensagem perguntando se o usuário realmente deseja fazer a edição, oferecendo as opções SIM ou NÃO. **P1.11.** O usuário escolhe a opção SIM. **P1.12.** O sistema exibe uma mensagem informando se o exame foi atualizado.
Fluxo de Exceção	Não existe Fluxo de Exceção.
Pós-condições	Ao término desta ação, a edição do exame deve ter sido efetuada com sucesso.

Fonte: Autoria própria.

Tabela 17 - Caso de Uso Excluir Exame.

Nome do caso de uso	Excluir Exame
Objetivo	O usuário exclui o exame, caso necessário.
Ator principal	Enfermeiro Administrador ou Comum.
Pré-condições	O usuário deve ter efetuado o *login*. A paciente escolhida pelo usuário deve estar previamente cadastrada.
Fluxo Principal	**P1.1.** O usuário escolhe a opção “Pacientes”. **P1.2.** O sistema mostra uma tela que possui duas guias que podem ser escolhidas pelo usuário, sendo primeiramente exibida a guia que possui a listagem das pacientes cadastradas (intitulada Pacientes). **P1.3.** O usuário escolher a paciente e clica sobre o botão “Exames”. **P1.4.** O sistema mostra um menu de opções de Exames **P1.5.** O usuário escolhe o menu do tipo de exame que deseja excluir e clica sobre ele. **P1.6.** O sistema mostra uma tela que possui duas guias que podem ser escolhidas pelo usuário, sendo

	primeiramente exibida a guia (intitulada pelo nome do exame escolhido) que possui a listagem do tipo do exame. **P1.7.** O usuário escolhe o exame que deseja fazer a exclusão e clica no botão "Excluir". **P1.8.** O sistema exibe uma mensagem perguntando se o usuário realmente deseja fazer a exclusão, oferecendo as opções SIM ou NÃO. **P1.9.** O usuário escolhe a opção SIM. **P1.10.** O sistema exibe uma mensagem informando se o exame foi excluído.
Fluxo de Exceção	Não existe Fluxo de Exceção.
Pós-condições	Ao término desta ação, a exclusão do exame deve ter sido efetuada com sucesso.

Fonte: Autoria própria.

Tabela 18 - Caso de Uso Listar Exames.

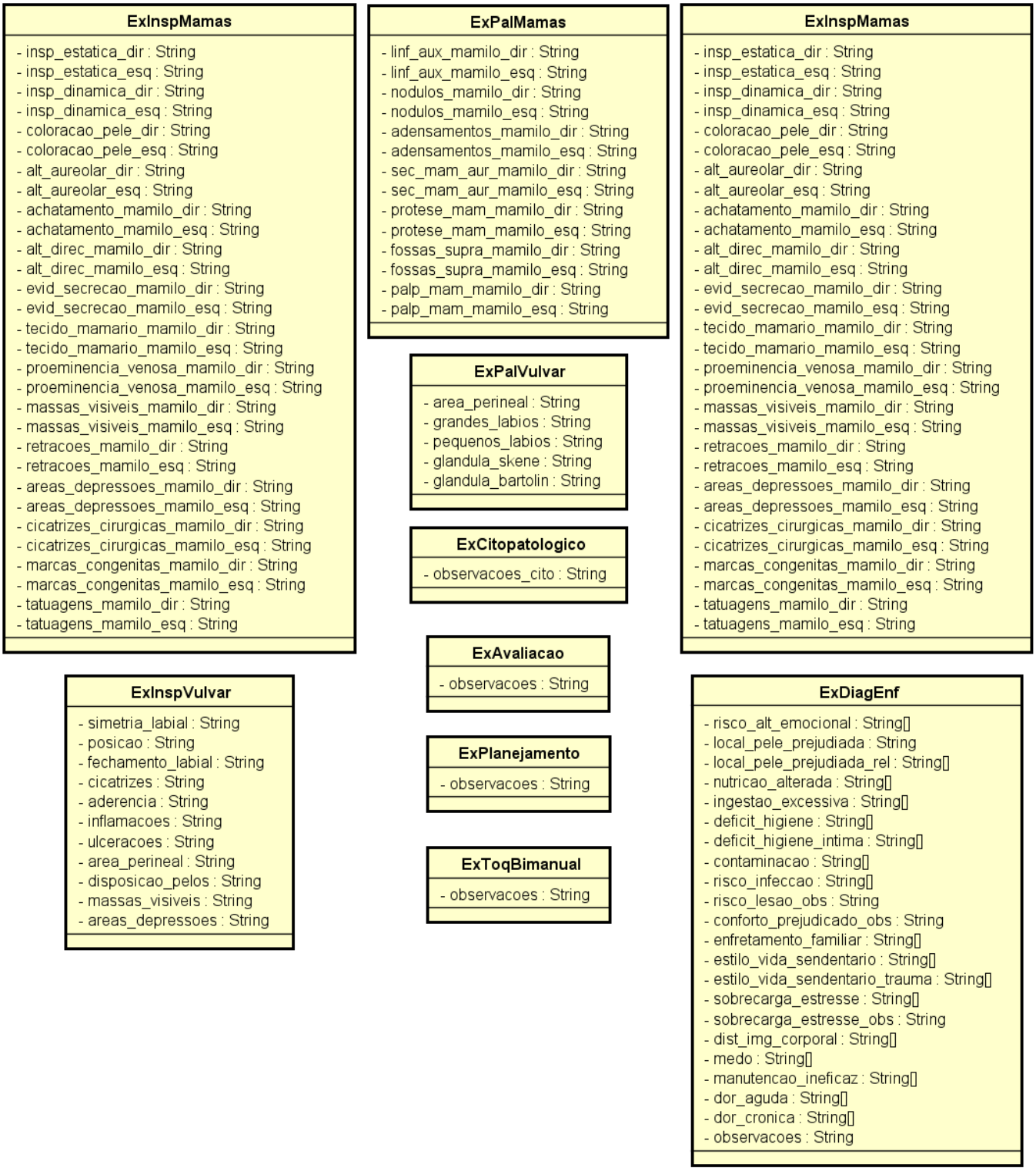
Nome do caso de uso	Listar Exames
Objetivo	O usuário lista os exames já cadastrados no sistema.
Ator principal	Enfermeiro Administrador ou Comum.
Pré-condições	O usuário deve ter efetuado o *login*.
Fluxo Principal	**P1.1.** O usuário escolhe a opção "Pacientes". **P1.2.** O sistema mostra uma tela que possui duas guias que podem ser escolhidas pelo usuário, sendo primeiramente exibida a guia que possui a listagem das pacientes cadastradas (intitulada Pacientes). **P1.3.** O usuário escolher a paciente e clica sobre o botão "Exames". **P1.4.** O sistema mostra um menu de opções de Exames **P1.5.** O usuário escolhe o menu do tipo de exame que deseja excluir e clica sobre ele. **P1.6.** O sistema mostra uma tela que possui duas guias que podem ser escolhidas pelo usuário, sendo primeiramente exibida a guia (intitulada pelo nome do exame escolhido) que possui a listagem do tipo do exame.

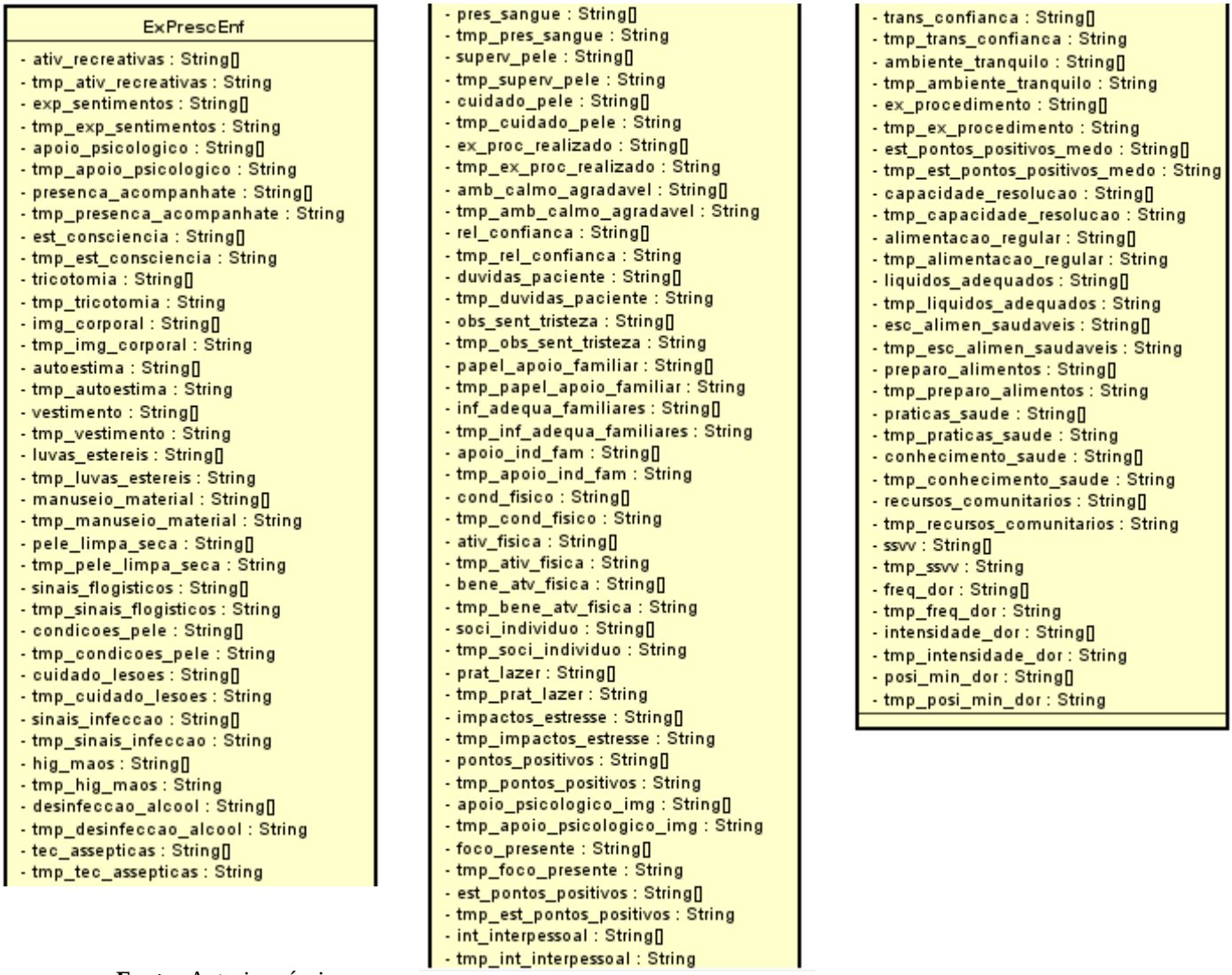
Fluxo de Exceção	Não existe Fluxo de Exceção.
Pós-condições	Ao término desta ação, o usuário poderá ter acesso a todos os exames da paciente, de acordo com tipo escolhido, cadastrados no sistema.

Fonte: Autoria própria.

APÊNDICE B – ESPECIFICAÇÃO DAS CLASSES DO DIAGRAMA DE CLASSES

Figura 26 – Classes detalhadas.

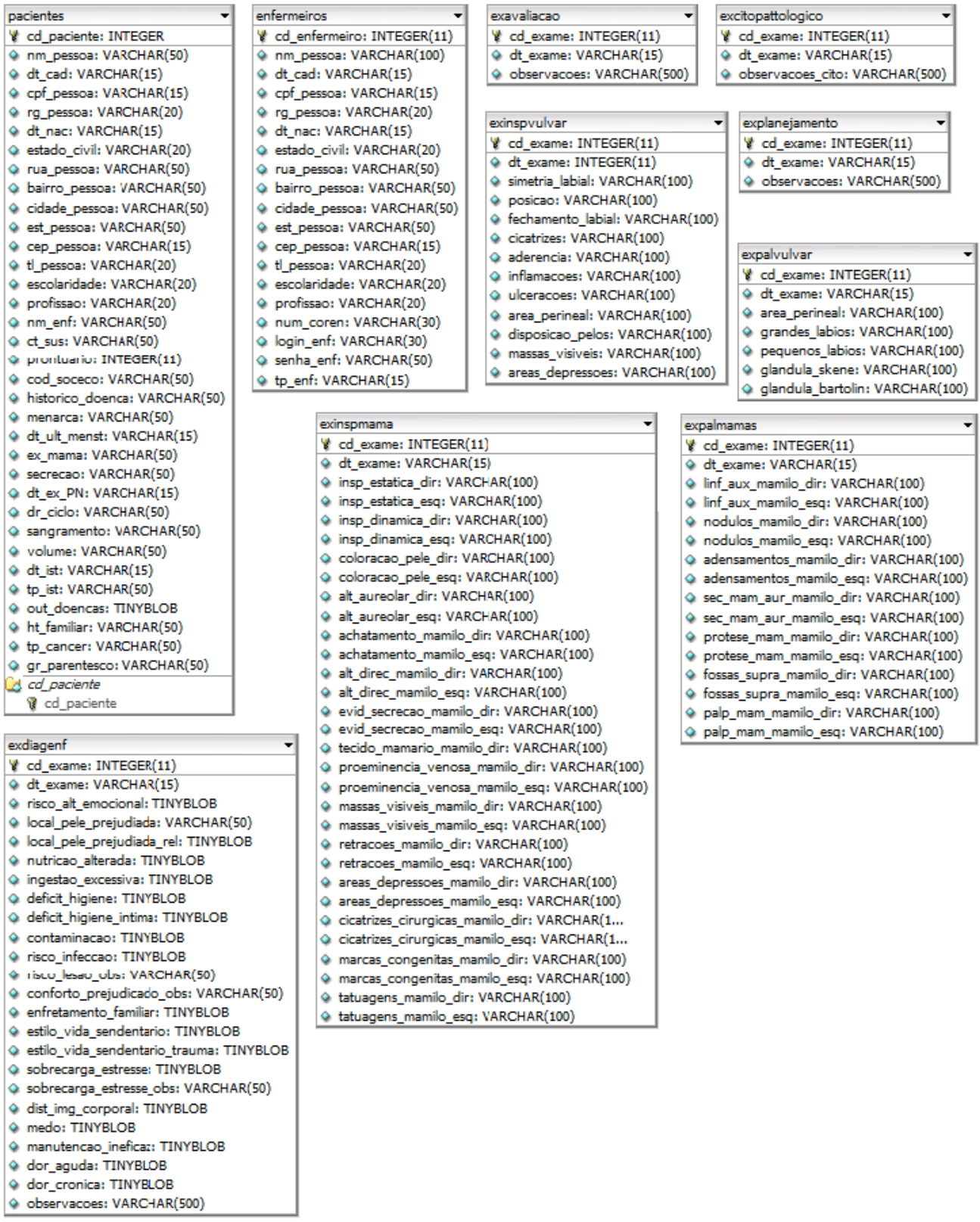


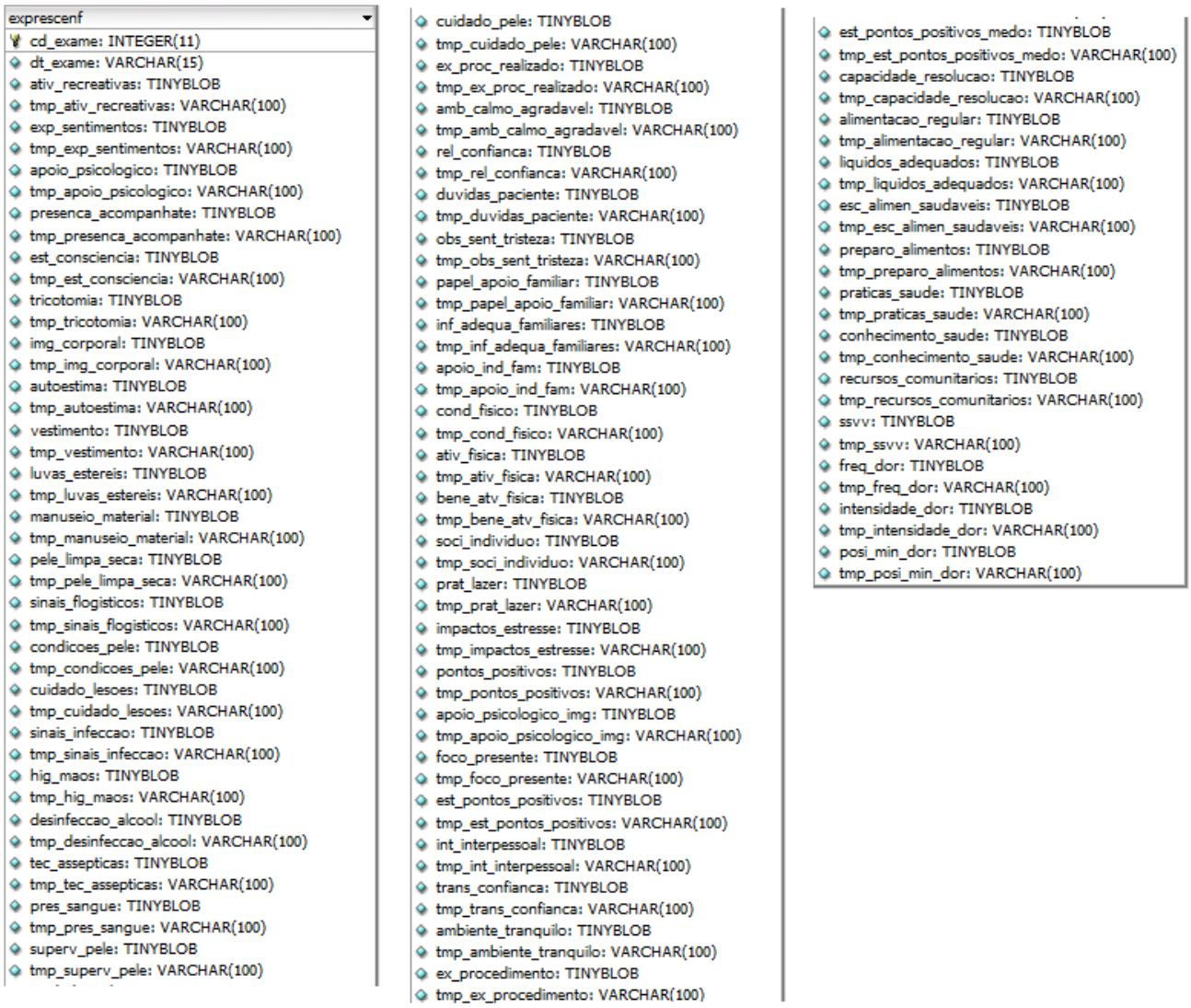


Fonte: Autoria própria.

APÊNDICE C – ESPECIFICAÇÃO DAS TABELAS DO DIAGRAMA DE ENTIDADE E RELACIONAMENTO

Figura 27 – Tabelas detalhadas.





Fonte: Autoria própria.