Fundação Educacional do Município de Assis
Instituto Municipal de Ensino Superior de Assis
Campus "José Santilli Sobrinho"

DANILO APARECIDO PORTO

SISTEMA PARA GESTÃO DE ACADEMIAS

Assis/SP
2017

Fundação Educacional do Município de Assis
Instituto Municipal de Ensino Superior de Assis
Campus "José Santilli Sobrinho"

DANILO APARECIDO PORTO

SISTEMA PARA GESTÃO DE ACADEMIAS

Projeto de pesquisa apresentado ao curso de Análise e Desenvolvimento de Sistemas do Instituto Municipal de Ensino Superior de Assis – IMESA e a Fundação Educacional do Município de Assis – FEMA, como requisito parcial à obtenção do Certificado de Conclusão.

Orientando (a): Danilo Aparecido Porto
Orientador (a): Alex Sandro Romeo de Souza Poletto

Assis/SP
2017

FICHA CATALOGRÁFICA

PORTO, Danilo Aparecido.
Sistema para Gestão de Academias /Danilo Aparecido Porto. Fundação Educacional do Município de Assis –FEMA – Assis, 2017.
45 páginas.

1. Sistema para gestão de academias. 2. Java. 3.JavaFX. 4.MySQL.

CDD:
Biblioteca da FEMA

SISTEMA PARA GESTÃO DE ACADEMIAS

DANILO APARECIDO PORTO

Trabalho de Conclusão de Curso apresentado ao Instituto Municipal de Ensino Superior de Assis, como requisito do Curso de Graduação de Análise e Desenvolvimento de Sistemas, avaliado pela seguinte comissão examinadora:

Orientador: Alex Sandro Romeo de Souza Poletto

Examinador: Domingos de Carvalho Villela Júnior

Assis/SP
2017

DEDICATÓRIA

Dedico este trabalho a todos os que estiveram ao meu lado durante essa jornada, aos que me apoiaram e me deram forças para chegar até o final.

AGRADECIMENTOS

Agradeço a Deus pela oportunidade de me graduar e pela capacidade de absorver os conhecimentos passados em sala, ao professor Alex por aceitar a ideia proposta para o trabalho e orientar nas dificuldades que apareceram dura o seu desenvolvimento, agradeço a todos que sempre estiveram do meu lado, amigos e família.

RESUMO

O projeto Sistema para Gestão de Academias tem como proposta trazer melhorias para o atendimento e gestão em qualquer academia, controlando a entrada e saída de clientes, autenticar usuários e clientes, gerenciar as vendas, cadastros de modalidades, avaliação física dos clientes, cadastro de treinos, cadastro de produtos e emitir relatórios. Para o desenvolvimento do sistema foi utilizado a linguagem de programação Java com a IDE Eclipse, MySQL como Banco de Dados e UML para a documentação do sistema.

Palavras-chave: Java, eclipse MySQL, UML.

ABSTRACT

The System for Management of Fitness Centers project has as the aim to bring improvements to the customer service and management in any gym, controlling the checking in and checking out of clients, authenticate users and clients, manage sales, registrater working out routines, clients physical evaluation, products and reports. For the development of the system it was used the Java programming language with the Eclipse IDE, MySQL as Database and UML for the system documentation.

Keywords: Java, eclipse MySQL, UML.

LISTA DE ILUSTRAÇÕES

LISTA DE TABELAS

SUMÁRIO

1. INTRODUÇÃO

Hoje em dia toda empresa de todos os portes e de qualquer seguimento necessitam de gestão, para atingir o ápice de seu crescimento e sucesso almejado.

Segundo Peter Drucker ,“Gerenciamento é substituir músculos por pensamentos, folclore e superstição por conhecimento, e força por cooperação”, sendo assim a melhor forma para gerir uma empresa é através de um software que substituirá toda papelada que antes era necessário para manter os dados de clientes e/ou funcionários, obrigações como pagamentos de contas, cobranças de inadimplentes e vendas armazenadas em arquivarias, com o sistema não é mais necessário, os dados serão guardados virtualmente melhorando muito o layout e organização de uma empresa. (Digital, 2016)

Mas o que é um software?

Software é uma série de instruções de código, escrito em uma determinada linguagem de programação, que são compreendidas e executadas pelo computador a fim de uma atividade especifica. (PACIEVITCH, 2016).

Para uma academia onde os fluxos de clientes são frequentes, vendas e cobranças são volumosas, o modo de gerir não pode ser arcaico. A melhor forma de agilizar o atendimento e o gerenciamento implementar um sistema para auxiliar na gestão.

1.1. OBJETIVOS

Esse projeto tem como objetivo desenvolver um sistema gerenciador de academias, com a finalidade de controlar a entrada de alunos, realizar matrículas, cadastrar modalidade, cadastro de alunos, cadastro de produtos, vendas, avaliação física, dentre outras rotinas, possibilitando melhorar o atendimento aos clientes, agilizar o fluxo na recepção, tornar o gerenciamento da academia digital, diminuindo os custos com materiais arcaicos como papeis usados para várias tarefas que visa o armazenamento de informações vitais para o controle do negócio.

1.2. PÚBLICO ALVO

Academias que ainda não possuam um software de gestão ou que já possui um, mais que não cumpri com as necessidades da mesma.

1.3. JUSTIFICATIVA

Esse sistema irá permitir maior agilidade no principal ponto de uma academia (recepção) onde o fluxo de pessoas é constante e o atendimento precisa ser rápido.

1.4. MOTIVAÇÃO

A maior motivação é poder por em pratica todo o conhecimento abstraído durante o período do curso e com as dificuldades ao decorrer do desenvolvimento do trabalho, buscar mais conceitos e informações relacionados às tecnologias do trabalho, possibilitando uma qualificação para o mercado de trabalho. Além disso, com o aumento de pessoas em busca de saúde física e bem-estar, as academias estão com mais clientes, necessitando de softwares que darão suporte no gerenciamento e agilidade no atendimento, sendo assim esse sistema pode ganhar mercado com esse pequeno nicho atendendo aos principais requisitos para um bom funcionamento de uma academia.

2. METODOLOGIA DE DESENVOLVIMENTO

O método proposto para o desenvolvimento do sistema é o modelo cascata, que sugere uma abordagem sequencial para o desenvolvimento do software, onde se inicia pelo levantamento dos requisitos, logo após é feita a modelagem do sistema através da UML (Unified Modeling Language) e em seguida a codificação utilizando o paradigma de Programação Orientado a Objetos.

2.1. UML (UNIFIED MODELIG LANGUEM – LINGUAGEM UNIFICADA DE MODELAGEM)

A UML é uma linguagem de modelagem que permite representar os eventos e problemas de um software em diagramas de forma gráfica. Essa modelagem é feita antes da construção do sistema, pois através da análise é levantado as necessidades e só então usamos os modelos de UML para entender melhor o funcionamento do sistema

2.1.1. DIAGRAMA DE CASO DE USO

O Diagrama de Caso de Uso modela os principais funcionamentos do sistema do ponto de vista do usuário. Esse modelo não se atenta a detalhes.

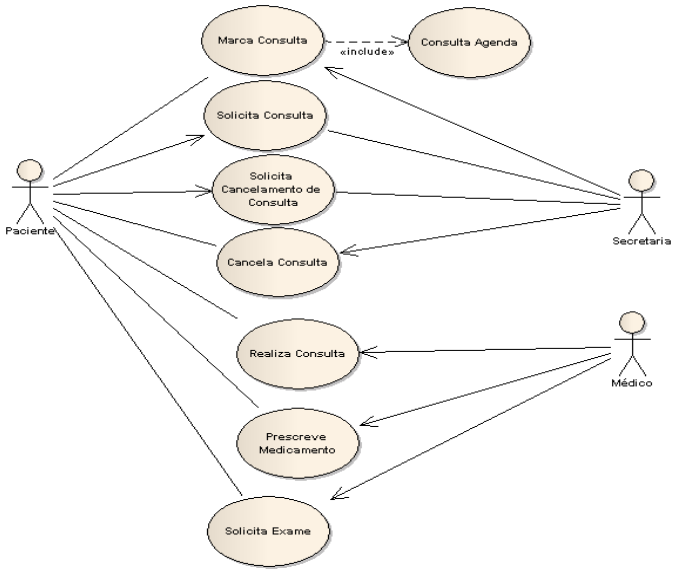

Figura 1: Exemplo Caso de Uso
Fonte: (RIBEIRO,2015)

2.1.2. DIAGRAMA DE CLASSE

Esse diagrama tem por objetivo mostrar as Classes que vão compor o sistema, elas se dividem em três partes: A primeira parte superior indica o nome da classe, a segunda mostra os atributos da classe e a parte inferior apresenta os métodos, mas também nos permite visualizar os relacionamentos entre cada classe.

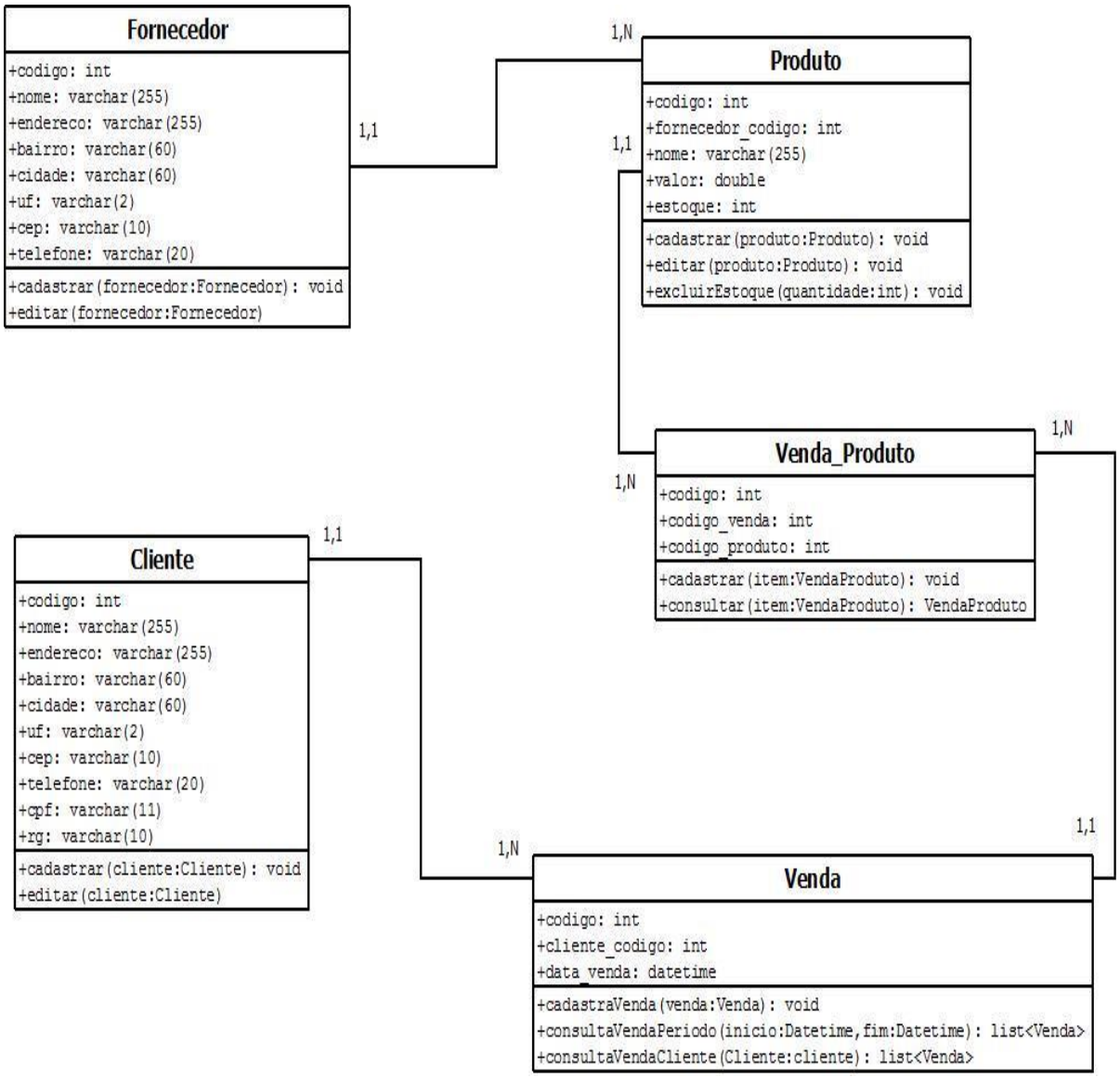

Figura 2: Exemplo Diagrama de classe
Fonte: (PROFISSIONAISTI**,** 2011**)**

2.1.3. DIAGRAMA DE SEQUÊNCIA

O Diagrama de Sequência usa como base um diagrama de caso de uso que possa ser mais complexo do que os demais, onde seu objetivo é mostrar os detalhes, interação e troca de mensagens entre os objetos que compões o respectivo caso de uso seguindo uma ordem cronológica.

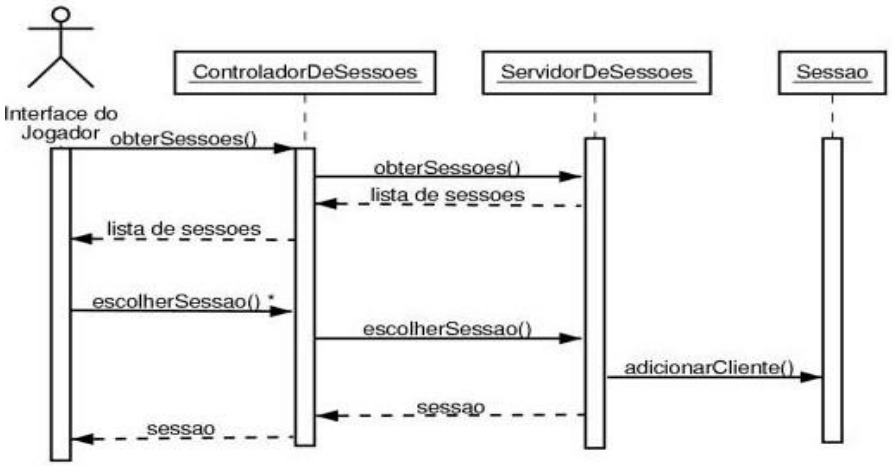

Figura 3: Exemplo Diagrama de Sequência
Fonte: (PROFISSIONAISTI **,** 2011**)**

2.1.4. DIAGRAMA DE ATIVIDADE

O Diagrama de Atividade demonstra o comportamento de processos e permite mapear as atividades que são compostas por várias ações e tais ações podem seguir vários caminhos dependendo da decisão tomada. **(**PROFISSIONAISTI **,** 2011**)**

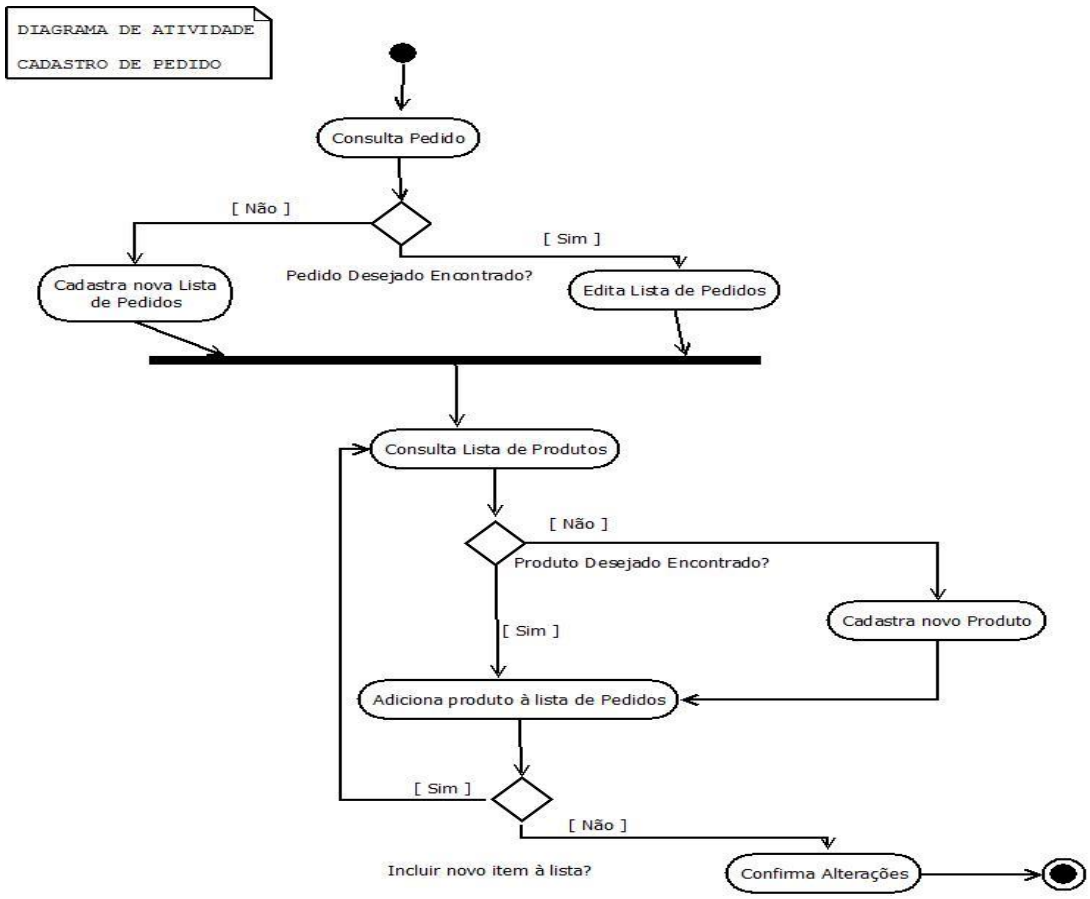

Figura 4: Exemplo Diagrama de Atividade
Fonte: (PROFISSIONAISTI **, 2011)**

2.1.5. Diagrama EAP

A EAP traz uma visão organizada das atividades executáveis do projeto, onde segue uma hierarquia dessas atividades, começando da atividade maior para suas subdivisões em atividades menores.

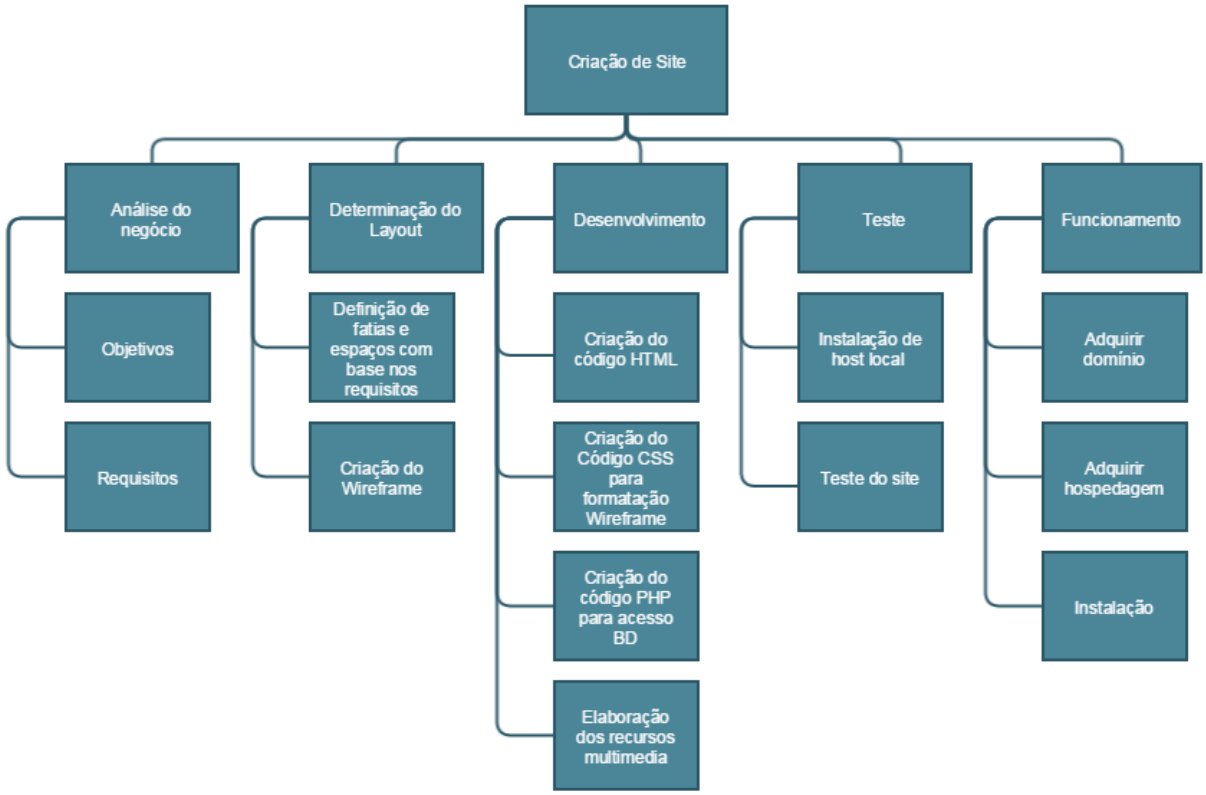

Figura 5: Exemplo de Diagrama EAP
Fonte: (MARCOS, 2014**)**

2.2. POO (PROGRAMAÇÃO ORIENTADA A OBJETOS)

Segundo Rumbaugh (1996), a orientação a objetos trata-se de uma maneira de pensar nos problemas utilizando modelos organizados a partir de conceitos do mundo real, que sendo o principal componente o objeto, que combinam dados e comportamentos. A programação orientada a objetos possui 4 pontos importantes.

2.2.1. Abstração

A abstração é o mapeamento do objeto do mundo real para um objeto abstrato (uma classe), que por obrigação possui uma única identidade para que não haja conflitos. Como no mundo real essas classes possuem atributos que as definem, por exemplo, um gato pode ter “raça”, “cor”, “tamanho” e esse gato pode ter comportamentos (Métodos) “Andar()”, “Miar()” e toda instancia de gato, possuirá as mesma característica que a classe gato tem.

2.2.2. Encapsulamento

O encapsulamento vem como uma forma de segurança, onde visa esconder os atributos para que outras classes não tenham acesso direto a elas, na maioria das linguagens de programação orientada a objetos reserva-se uma palavra “private” que define o encapsulamento. Mas aí vem a dúvida, como acessar essas propriedades “escondidas”?
Essas linguagens têm como padrão a criação de métodos assessores chamados de getters e setters que lhe permitirá acessar a propriedade da classe de forma indireta, um retornando valor e o outro atribuindo valor ao atributo.

2.2.3. Herança

A herança é ponto muito interessante da POO, pois aqui se dá o conceito de reuso de código, esse paradigma diminui consideravelmente o número de linha de código.
Como na vida real, a herança na programação cede características de um indivíduo para o outro e o próximo herdeiro obtém as características de todos os seus ancestrais e assim sucessivamente.

2.2.4. Polimorfismo

Na programação orientada a objetos o polimorfismo é a capacidade de fazer uma “certa coisa” de várias formas diferentes, tratando-se de POO estamos falando da chamada de métodos, que podem agir de diferentes formas dependendo do objeto que o chamou, esse é o polimorfismo de sobreposição, onde só é possível através da herança e com a sobreposição (@Override) dos métodos da classe pai na classe filha usando a mesma assinatura. Outra forma de polimorfismo é o polimorfismo de sobrecarga, onde há vários métodos com o mesmo nome, na mesma classe, com assinaturas diferentes se comportando de formas diferentes. Segundo o Guanabara esses são os dois tipos de polimorfismo mais usados. (GUANABARA, 2016).

2.3. FERRAMENTAS NECESSÁRIAS PARA O DESENVOLVIMENTO DO PROJETO

2.3.1. FreeMind

FreeMind é um software Open Source para a criação de mapas mentais. Os mapas mentais são uteis para mostrar ideias onde somente os principais pontos são ressaltados.

2.3.2. Astah Community

Astah Community é uma ferramenta que possibilita criar diagramas fundamentais em UML de forma gráfica, facilitando o desenvolvimento com essa metodologia, por ser Community não possui todos os recursos liberados. Somente a verão Profissional possui todos os recursos.

2.4. DBDESIGNERFORK

DBDesignerFork é uma ferramenta para modelagem de lógica de banco de dados relacional, essa ferramenta também possibilita a criação de modelos a partir de banco de dados já existentes chamado de engenharia reversa. (ANTÔNIO, 2015).

2.4.1. JAVA

Java é uma linguagem de programação orientada a objetos, diferente das demais linguagens como C que é compilada tendo seu código fonte traduzido direto para linguagem de máquina gerando um código executável que “amarra” a aplicação ao sistema operacional ou plataforma que foi gerado. Em Java suas aplicações independem de plataforma, pois todo código fonte passa pelo compilador Java (Java Compiler - Javac) transformando o código de alto nível em bytecode. O compilador Java envia o bytecode para a máquina virtual Java ou Java Virtual Machine (JVM) que interpreta esse código para os mais diversos sistemas operacionais.

2.4.2. IDE ECLIPSE

Criada em Java pela IBM a IDE Eclipse é uma ferramenta editor de texto Open Source que ajuda no desenvolvimento de software principalmente na linguagem de programação Java, mas suporta outras linguagens por meio de plug-ins.

2.4.3. JavaFX

JavaFX é uma tecnologia para criação de aplicações visuais ricas para o cliente, ou seja, elementos 2D, 3D, formulários, gráficos, etc. Sendo desenvolvido em Java possibilita ao programador utilizar as APIs disponíveis na linguagem, APIs de coleções, I/O entre outras.

A estrutura de cena.

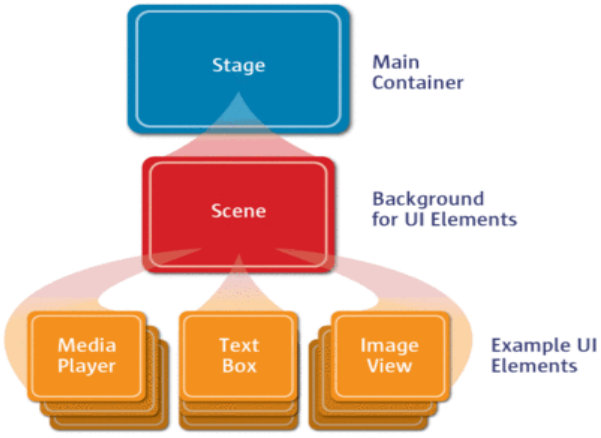

Figura 6: Estrutura de Cena JavaFX

Fonte: (https://www.google.com.br/search?q=javafx+estrutura+de+cena)

Fazendo uma analogia, JavaFX funciona como uma peça de teatro, onde o Stage se assemelha com o palco principal, é a base para que toda a cena funcione, o Scene seria a cena contendo todo o cenário e em cada cenário de JavaFX pode conter nós e cada nó pode ter vários "filhos" e deve ter no máximo um pai "root" (BorderPane, GridPane, AnchorPane, Button), por exemplo, cria-se um layout BorderPane (root) e dentro desse layout ponha-se um GridPane (filho) e no Grid em cada célula botões e campos de textos (filhos de Grid). (TOSIN, 2017).

2.4.4. Scene Builder

O Scene Builder é um editor externo "arrasta" e "solta" voltado para a criação de janelas em JavaFX, ele é capaz de gerar arquivos FXML onde a API do JavaFX as interpreta e constrói as telas;

2.4.5. MYSQL

MySQL é sistema gerenciador de banco de dados relacional que utiliza como base a linguagem SQL (Structure Query Language) linguagem a qual acessa, insere e gerencia o banco o conteúdo armazendo. (PORTALEDUCACAO, 2008)

2.4.6. TIBCO Jaspersoft Studio

Jaspersoft Studio é um editor de relatórios baseado em Eclipse onde é possível, criar e executar modelos de relatórios, fazer consultas a fontes de dados como JDBC, JavaBeans, JSON e XML. Através dessa ferramenta podemos gerar documentos como PDF para impressão, HTML, XML, Jason entre outros. (jaspersoft, 2017).

2.4.7. JasperReports

JaperReport é uma biblioteca que gera relatórios em arquivos XML para a linguagem Java, sendo capaz de usar dados provenientes de qualquer fonte de dados. (jaspersoft, 2017).

3. DOCUMENTANDO O SISTEMA

Neste capitulo será realizado o levantamento das funcionalidades do sistema utilizando a UML.

3.1. LEVANTAMENTO DE REQUISITOS

Antes de iniciar o desenvolvimento deve-se levantar quais funcionalidades o sistema terá, abstraindo para o projeto

3.1.1. Técnica para o levantamento dos requisitos

A técnica utilizada para o levantamento dos requisitos foi a de observação e Brainstorming ou “Tempestade de ideia” que consiste em levantar ideias e incorporar ao projeto.

3.1.2. Análise dos requisitos

O sistema deve automatizar toda a parte de cadastros de visitantes, clientes, funcionários, fornecedores, produtos, tipo de produtos, avaliação física, modalidades, matriculas, efetuar vendas, controlar estoque, fornecer consultas e gerar relatórios das vendas, das matriculas, de clientes, de visitantes, de produtos, de avaliações físicas. Os clientes ao entrar na academia faram um check-in para a confirmação de que não há pendencias.
Por medida de segurança o software ao ser iniciado solicitará *login* ao usuário.

3.1.2.1. Descrição do Problema

Por mais inacreditável que seja, ainda existe academias que não são automatizadas e as fichas para o controle dos processos fazem parte do cotidiano dessas empresas, isso é um fator que atrapalha o desenvolvimento do negócio, pois não se pode ter informações precisas e cruciais para uma tomada de decisão no ponto de vista empresarial, onde os relatórios gerado pelo sistema pode mostrar informações de pontos importantes da

academia, por exemplo, um produto que foi comprado algum tempo e não vende, o proprietário pode reduzir o preço para que o produto fique mais atrativo para o cliente.

3.1.2.2. Resultado esperado

Espera-se que o sistema cumpra com todas as necessidades levantadas, e agilize os cadastros e, e que torne mais seguras e confiáveis os atendimentos das academias onde ocorrem os processos mais importe desse seguimento.

3.1.3. Eventos do Sistema

- Manter clientes
- Manter funcionários
- Manter fornecedores
- Manter produtos
- Manter tipo
- Efetuar vendas
- Manter modalidades
- Manter treinos
- Efetuar avaliação física
- Cadastrar matrículas
- Consultar clientes
- Consultar vendas
- Consultar produtos
- Consultar avaliação física
- Consultar matrículas
- Consultar treino
- Controlar o estoque
- Relatórios de matriculas
- Relatório de vendas
- Relatório de produtos
- Relatório de clientes
- Autenticar usuário

- Check-in cliente
- Cadastra *login*

3.2. ESTRUTURA ANALITICA DO PROJETO

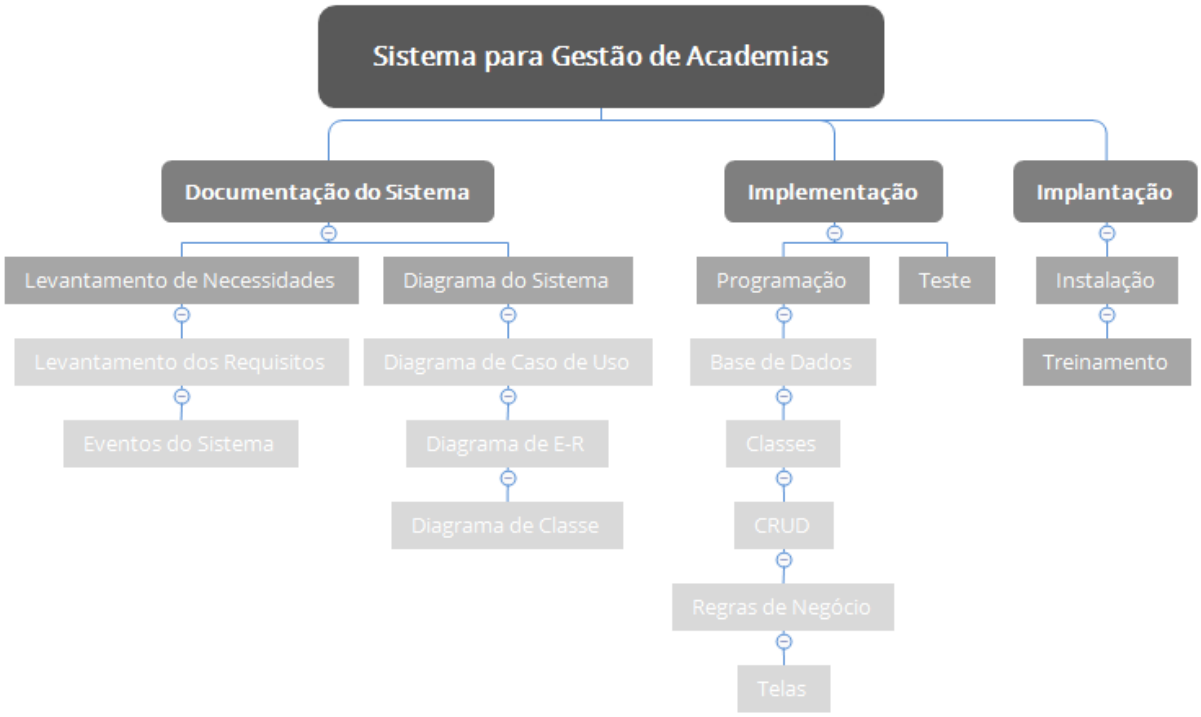

Figura 7: EAP

3.3. DIAGRAMAS DE CASO DE USO

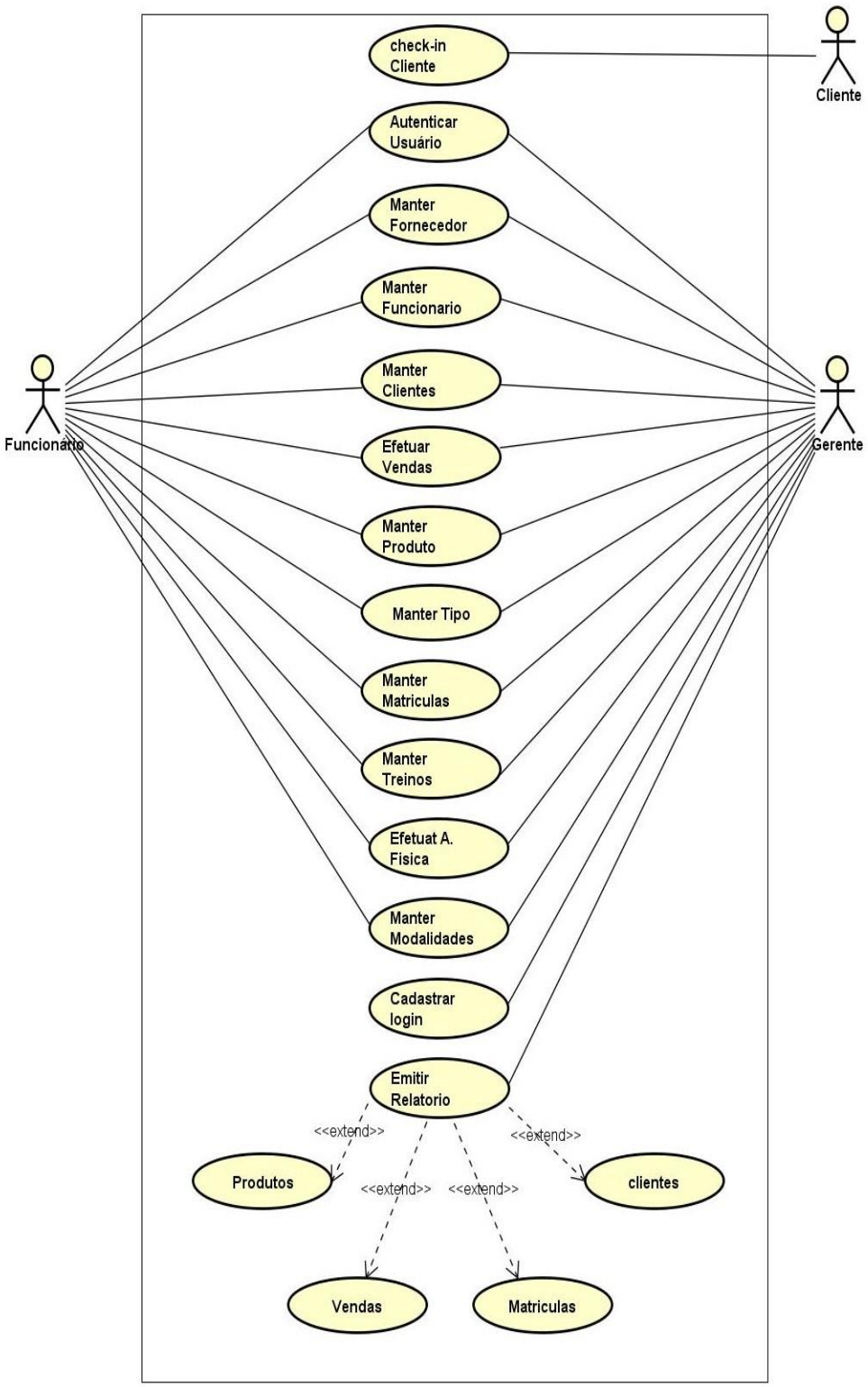

Figura 8: Diagrama de Caso de Uso Geral

3.3.1. Detalhamentos dos Casos de Uso

- **Autenticar Usuário**

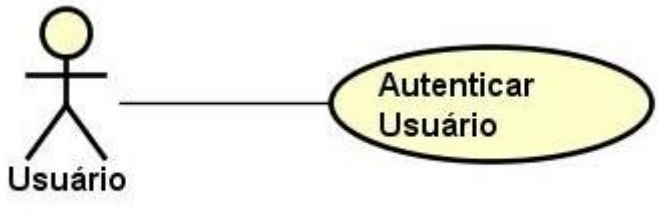

Figura 9:Caso de Uso Autenticar Usuário.

Atores Funcionário, Gerente.
Fluxo Principal **1.** O ator inicia o sistema. **2.** O sistema exibe a tela de autenticação do sistema. **3.** O ator insere seu *login* e senha. **4.** O ator prime o botão entrar. **[A1]** **5.** O sistema verifica *login* e senha. **[E1]** **6.** O sistema abre a tela principal.
Fluxo de Alternativo **A1.** Cancelar a operação. **a.** O ator cancela a operação. **b.** Os campos são limpos para outra operação.
Fluxo de Exceção **E1.** O ator errar um dos campos. **a.** O ator insere algum dado errado em uns dos campos. **b.** O sistema exibe uma mensagem de erro, mas não exibe qual campo errou por medida de segurança. **c.** O sistema limpa os campos.

Tabela 1: Caso de Uso Autenticar Usuário.

- **Check-in Cliente**

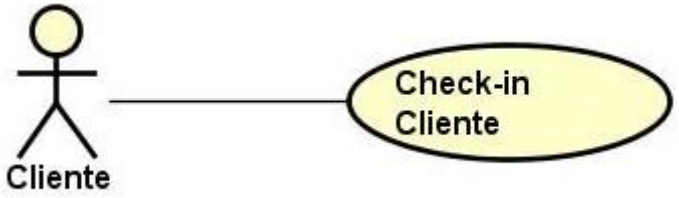

Figura 10: Caso de Uso Check-in Cliente.

Atores Cliente.
Fluxo Principal **1.** O cliente insere sua senha no sistema. **2.** O sistema verifica a senha. **[A1]** **3.** O sistema verifica a situação do cliente. **[E1]** **4.** O sistema libera o acesso do usuário a academia.
Fluxo Alternativo **A1.** Senha invalida. **a.** O cliente insere no sistema uma senha invalida. **b.** O sistema exibe uma mensagem de senha invalida. **c.** O sistema limpa o campo para uma nova digitação.
Fluxo de Exceção **E1.** Cliente com mensalidade vencida. **a.** O sistema acusa vencimento de mensalidade. **b.** O sistema bloqueia a entrada do cliente a academia. **c.** O sistema dá a opção do pagamento da mensalidade.

Tabela 2:Caso de Uso Check-in Cliente.

- **Cadastrar Login**

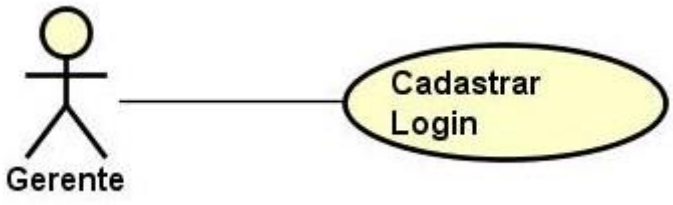

Figura 11: Caso de Uso Cadastrar *Login*.

Ator Gerente.
Pré-Condições • O ator deve se autenticar no sistema com seu *login* e senha já, que já foi inserido.
Fluxo Principal **1.** O Gerente abre tela de cadastro de *Login* no menu principal. **2.** O Gerente seleciona qual é o funcionário. **[E1]** **3.** O funcionário digita seu *login* e senha. **[E2]** **4.** O gerente confirma a operação. **[A1]** **5.** O sistema exibe uma mensagem de confirmação de cadastro.
Fluxo Alternativo **A1.** Cancelar a operação. **a.** O Gerente opta por pressionar o botão de cancelar. **b.** O sistema pergunta se é isso que o ator pretende. **c.** O gerente confirma. **d.** O sistema retorna ao primeiro passo do fluxo principal.
Fluxo de Exceção. **E1.** O nome já possui *login* e senha. **a.** O sistema verifica se o funcionário já está cadastrado com *login* e senha. **b.** O sistema notifica a redundância e finaliza a operação. **E2.** *Login* e senha já existente. a. O sistema faz a verificação se há as mesmas informações já cadastradas. b. O sistema notifica a redundância. O sistema solicita nova senha e *login* ao funcionário.

Tabela 3: Caso de Uso Cadastrar *Login*.

- **Manter Cliente**

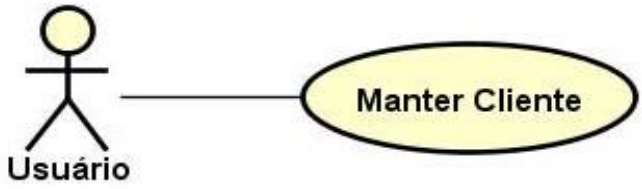

Figura 12: Caso de Uso Manter Cliente.

Ator Usuário.
Pré-condições • O usuário deverá fazer *login* no sistema.
Fluxo Principal **1.** O usuário abre a tela de cadastro de cliente no menu principal. **2.** O usuário insere os dados. **3.** O usuário pressiona o botão salvar. **[A1]** **4.** O sistema verifica as obrigatoriedades de campos. **[E1]** **5.** O sistema exibe uma mensagem de sucesso.
Fluxo Alternativo **A1.** Cancelar a operação. **a.** O usuário opta por pressionar o botão de cancelar. **b.** O sistema exibe uma mensagem de confirmação. **c.** O usuário confirma a opção. **[A2]** **d.** O sistema limpa os campos. **A2. Cancelar confirmação.** **a.** O usuário opta por pressionar o botão de cancelar da mensagem. **b.** O sistema permanece com os dados que o usuário digitou.
Fluxo de Exceção **E1.** Campos nulos. **a.** O sistema notifica que campo não pode ser nulo e pede ao usuário a inserção dos dados. **b.** O usuário completa o campo. **c.** O sistema retorna ao **Fluxo Principal 1**.

Tabela 4: Caso de Uso Manter Cliente.

- **Manter Funcionário**

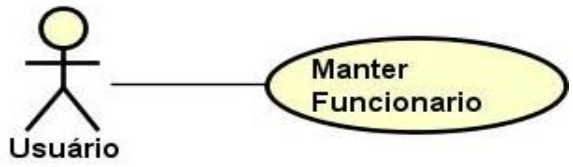

Figura 13: Caso de Uso Manter Funcionário.

Ator Usuário.
Pré-condições • O ator deverá ter feito *login* no sistema.
Fluxo Principal **1.** O usuário abre a tela de cadastro de funcionário no menu principal. **2.** O usuário insere os dados. **3.** O usuário pressiona o botão de salvar. **[A1]** **4.** O sistema verifica as obrigatoriedades de campos. **[E1]** **5.** O sistema exibe mensagem de sucesso.
Fluxo Alternativo **A1.** Cancelar operação **a.** O usuário opta por pressionar o botão de cancelar. **b.** O sistema exibe uma mensagem de confirmação. **c.** O usuário confirma. **[A2]** **d.** O sistema limpa os campos. **A2.** Cancelar confirmação. **a.** O usuário opta por cancelar confirmação. **b.** O sistema permanece com os dados inseridos pelo usuário.
Fluxo de Exceção **E1.** Campos nulos. **a.** O sistema notifica que o campo não pode ser nulo e pede ao usuário a inserção dos dados. **b.** O usuário completa o campo. **c.** O sistema retorna ao **Fluxo Principal 1**.

Tabela 5: Caso de Uso Manter Funcionário.

- **Manter Fornecedores**

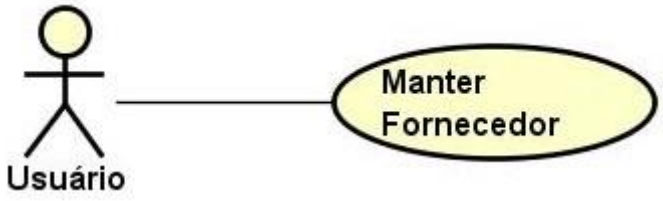

Figura 14: Caso de Uso Manter Fornecedor.

Ator Usuário.
Pré-condições O usuário deverá ter feito *login* no sistema.
Fluxo Principal **1.** O usuário abre a tela de cadastro de fornecedor no menu principal. **2.** O usuário insere os dados. **3.** O usuário pressiona o botão de salvar. **[A1]** **4.** O sistema verifica as obrigatoriedades de campos. **[E1]** **5.** O sistema exibe mensagem de sucesso.
Fluxo Alternativo **A1.** Cancelar operação **a.** O usuário opta por pressionar o botão de cancelar. **b.** O sistema exibe uma mensagem de confirmação. **c.** O usuário confirma. **[A2]** **d.** O sistema limpa os campos. **A2.** Cancelar confirmação. **a.** O usuário opta por cancelar confirmação. **b.** O sistema permanece com os dados inseridos pelo usuário.
Fluxo de Exceção **E1.** Campos nulos. **a.** O sistema notifica que o campo não pode ser nulo e pede ao usuário a inserção dos dados. **b.** O usuário completa o campo. **c.** O sistema retorna ao **Fluxo Principal 1.**

Tabela 6: Caso de Uso Manter Fornecedor.

- **Manter Tipo.**

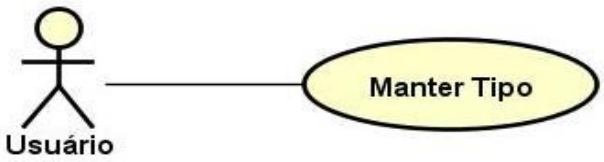

Figura 15: Caso de Uso Manter Tipo.

Ator Usuário.
Pré-condições • O usuário deverá ter feito *login* no sistema.
Objetivo • O objetivo é cadastrar e dar aos produtos divisões em tipo.
Fluxo Principal **1.** O usuário abre a tela de cadastro de tipo no menu principal. **2.** O usuário insere os dados. **3.** O usuário pressiona o botão de salvar. **[A1]** **4.** O sistema verifica as obrigatoriedades de campos. **[E1]** **5.** O sistema exibe mensagem de sucesso.
Fluxo Alternativo **A1.** Cancelar operação **a.** O usuário opta por pressionar o botão de cancelar. **b.** O sistema exibe uma mensagem de confirmação. **c.** O usuário confirma. **[A2]** **d.** O sistema limpa os campos. **A2.** Cancelar confirmação. **a.** O usuário opta por cancelar confirmação. **b.** O sistema permanece com os dados inseridos pelo usuário.
Fluxo de Exceção **E1.** Campos nulos. **a.** O sistema notifica que o campo não pode ser nulo e pede ao usuário a inserção dos dados. **b.** O usuário completa o campo. **c.** O sistema retorna ao **Fluxo Principal 1**.

Tabela 7: Caso de Uso Manter Tipo.

- **Manter Produto.**

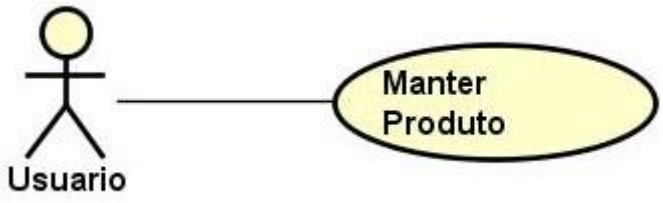

Figura 16: Caso de Uso Manter Produto.

Ator Usuário.
Pré-condições O ator deverá ter feito *login* no sistema.
Fluxo Principal **1.** O usuário abre a tela de cadastro de produto no menu principal. **2.** O usuário insere os dados do produto e o tipo. **3.** O usuário pressiona o botão de salvar. **[A1]** **4.** O sistema verifica as obrigatoriedades de campos. **[E1]** **5.** O sistema exibe mensagem de sucesso.
Fluxo Alternativo **A1.** Cancelar operação **a.** O usuário opta por pressionar o botão de cancelar. **b.** O sistema exibe uma mensagem de confirmação. **c.** O usuário confirma. **[A2]** **d.** O sistema limpa os campos. **A2.** Cancelar confirmação. **a.** O usuário opta por cancelar confirmação. **b.** O sistema permanece com os dados inseridos pelo usuário.
Fluxo de Exceção **E1.** Campos nulos. **a.** O sistema notifica que o campo não pode ser nulo e pede ao usuário a inserção dos dados. **b.** O usuário completa o campo. **c.** O sistema retorna ao **Fluxo Principal 1**.

Tabela 8: Caso de Uso Manter Produto.

- **Efetuar Venda.**

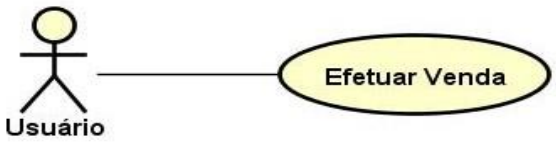

Figura 17: Caso de Uso Manter Produto.

Ator Usuário
Pré-condições • O usuário deverá fazer *login* no sistema.
Fluxo Principal **1.** O usuário abre a tela de vendas no menu principal. **2.** O usuário insere produto, quantidade e o nome do cliente. **3.** O usuário escolha a forma de pagamento a desejo do cliente. **[A1] [A2]** **4.** O usuário pressiona o botão salvar. **[A3]** **5.** O sistema confere se a quantidade de produto tem em estoque. **[E1]** **6.** O sistema exibe mensagem de sucesso.
Fluxo Alternativo **A1.** Pagamento à vista. **a.** O usuário seleciona a opção à vista na tela de vendas. **b.** O sistema retorna ao **Fluxo Principal 4.** **A2.** Pagamento a prazo. **a.** O usuário seleciona a quantidade de parcelas permitida pelo sistema. **b.** O sistema retorna ao **Fluxo Principal 4.** **A3.** Cancelar operação. **a.** O usuário opta por pressionar o botão cancelar. **b.** O sistema exibe mensagem de confirmação. **c.** O sistema limpa os campos.
Fluxo de Exceção **E1.** Quantidade insuficiente. **a.** O usuário insere quantidade acima do que o estoque possui. **b.** O sistema notifica falha na quantidade e pede correção. **c.** O sistema retorna ao **Fluxo Principal 5.**

Tabela 9: Caso de Uso Efetuar Venda.

- **Manter Modalidades.**

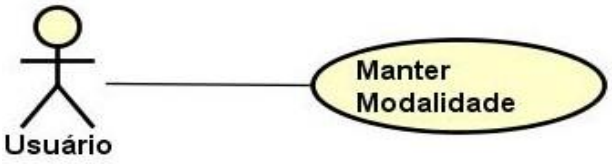

Figura 18: Caso de Uso Manter Modalidade.

Ator Usuário
Pré-condições • O usuário deverá ter feito *login* no sistema.
Fluxo Principal **1.** O usuário abre a tela de cadastro de modalidade no menu principal. **2.** O usuário insere os dados para o cadastro. **3.** O usuário pressiona o botão de salvar. **[A1]** **4.** O sistema verifica as obrigatoriedades de campos. **[E1]** **5.** O sistema exibe mensagem de sucesso.
Fluxo Alternativo **A1.** Cancelar operação **a.** O usuário opta por pressionar o botão de cancelar. **b.** O sistema exibe uma mensagem de confirmação. **c.** O usuário confirma. **[A2]** **d.** O sistema limpa os campos. **A2.** Cancelar confirmação. **a.** O usuário opta por cancelar confirmação. **b.** O sistema permanece com os dados inseridos pelo usuário.
Fluxo de Exceção **E1.** Campos nulos. **a.** O sistema notifica que o campo não pode ser nulo e pede ao usuário a inserção dos dados. **b.** O usuário completa o campo. **c.** O sistema retorna ao **Fluxo Principal 1**.

Tabela 10: Caso de Uso Manter Modalidade.

- **Manter Treinos**

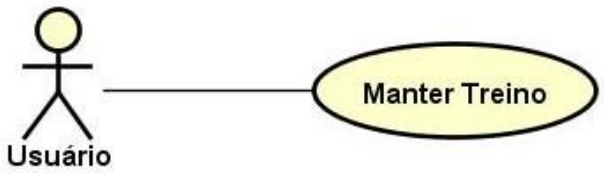

Figura 19: Caso de Uso Manter Treino.

Ator Usuário.
Pré-condições • O usuário deverá ter feito *login* no sistema.
Fluxo Principal **1.** O usuário abre a tela de cadastro de treinos no menu principal. **2.** O usuário preenche os campos. **3.** O usuário pressiona o botão de salvar. **[A1]** **4.** O sistema verifica as obrigatoriedades de campos. **[E1]** **5.** O sistema exibe mensagem de sucesso.
Fluxo Alternativo **A1.** Cancelar operação **a.** O usuário opta por pressionar o botão de cancelar. **a.** O sistema exibe uma mensagem de confirmação. **b.** O usuário confirma. **[A2]** **c.** O sistema limpa os campos. **A2.** Cancelar confirmação. **a.** O usuário opta por cancelar confirmação. **b.** O sistema permanece com os dados inseridos pelo usuário.
Fluxo de Exceção **E1.** Campos nulos. **a.** O sistema notifica que o campo não pode ser nulo e pede ao usuário a inserção dos dados. **b.** O usuário completa o campo. **c.** O sistema retorna ao **Fluxo Principal 1**.

Tabela 11: Caso de Uso Manter Treino.

- **Efetuar Avaliação Física**

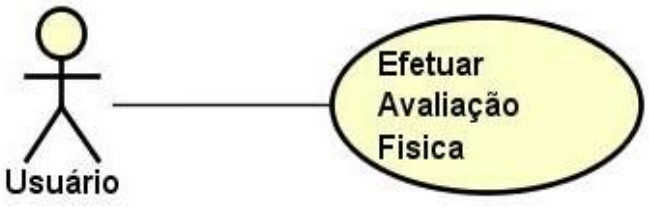

Figura 20: Caso de Uso Efetuar Avaliação Física.

Ator Usuário.
Pré-condições • O usuário deverá ter feito *login* no sistema.
Fluxo Principal **1.** O usuário abre a tela de cadastro de avaliação física no menu principal **2.** O usuário insere os dados de acordo com os aspectos do cliente. **3.** O usuário pressiona o botão de salvar. **[A1]** **4.** O sistema exibe mensagem de sucesso.
Fluxo Alternativo **A1.** Cancelar operação. **a.** O usuário opta por pressionar o botão de cancelar. **b.** O sistema exibe uma mensagem de confirmação. **c.** O usuário confirma. **[A2]** **d.** O sistema limpa os campos. **A2.** Cancelar confirmação. **a.** O usuário opta por cancelar confirmação. **b.** O sistema permanece com os dados inseridos pelo usuário.

Tabela 12: Caso de Uso Efetuar Avaliação Física.

- **Cadastrar matriculas**

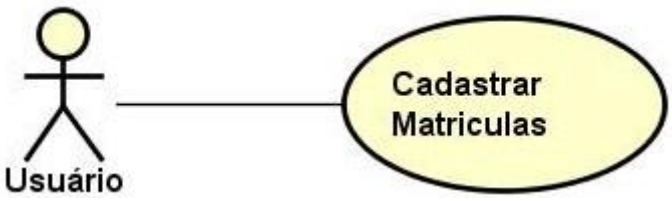

Figura 21: Caso de Uso Cadastrar Matriculas.

Ator Usuário.
Pré-condições • O usuário deverá ter feito *login* no sistema.
Fluxo Principal **1.** O usuário abre a tela de cadastro de matriculas no menu principal. **2.** O usuário insere os dados. **3.** O usuário pressiona o botão de salvar. **[A1] [E1]** **4.** O sistema exibe mensagem de sucesso.
Fluxo Alternativo **A1.** Cancelar operação. **a.** O usuário opta por pressionar o botão de cancelar. **b.** O sistema exibe uma mensagem de confirmação. **c.** O usuário confirma. **[A2]** **d.** O sistema limpa os campos. **A2.** Cancelar confirmação. **a.** O usuário opta por cancelar confirmação. **b.** O sistema permanece com os dados inseridos pelo usuário. **Fluxo de Exceção** **E1.** Campos Nulos **a.** O sistema verifica se o usuário inseriu dados em todos os campos obrigatórios. **b.** O sistema exibe mensagem de erro. **c.** O sistema retorna ao **Fluxo Principal 2**.

Tabela 13: Caso de Uso Cadastrar Matricula.

- **Consultar Cliente**

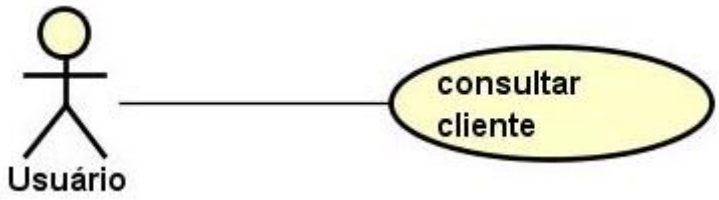

Figura 22: Caso de Uso Consultar Cliente.

Ator Usuário
Fluxo Principal **1.** O usuário abre a tela de consulta de cliente no menu principal. **2.** O usuário insere o nome do cliente. **3.** O usuário pressiona o botão de pesquisar. **4.** O sistema verifica se há um cliente com tal nome. **[E1]** **5.** O sistema retorna os dados do cliente e sua situação. **6.** O usuário escolhe qual das operações deseja realizar, alterar **[A1]**, excluir **[A2]**.
Fluxo Alternativo **A1.** Altera campos. **a.** O usuário pressiona o botão alterar que abre outra tela para a alteração dos dados. **b.** O usuário altera os campos desejados. **c.** O usuário pressiona o botão salvar. **d.** O sistema exibe mensagem de sucesso. **A2.** Excluir cliente **a.** O usuário pressiona o botão excluir. **[E2]** **b.** O sistema abre outra tela com os dados do cliente para confirmação. **c.** O usuário pressiona novamente o botão excluir. **[A2.1]** **d.** O sistema exibe mensagem de confirmação. **e.** O usuário confirma a operação. Os dados do cliente são excluídos permanentemente do sistema.

A2.1. Cancelar operação.

- **a.** O usuário pressiona o botão cancelar.
- **b.** O sistema exibe mensagem de confirmação.
- **c.** O usuário confirma.
- **d.** O sistema retorna ao **Fluxo Alternativo A1.**

Fluxo de Exceção

E1. Nome inválido.

- **a.** O sistema exibe mensagem de erro acusando nome invalido.
- **b.** E retorna ao retorna ao **Fluxo Alternativo A1.**

E2. Pendencia.

- **a.** O sistema verifica se o cliente tem alguma pendencia.
- **b.** O sistema exibe mensagem de erro acusando pendencia.
- **c.** O sistema só permite a exclusão de cliente regularizado com a academia.

Tabela 14: Caso de Uso Consultar Cliente.

3.4. DIAGRAMA DE CLASSE

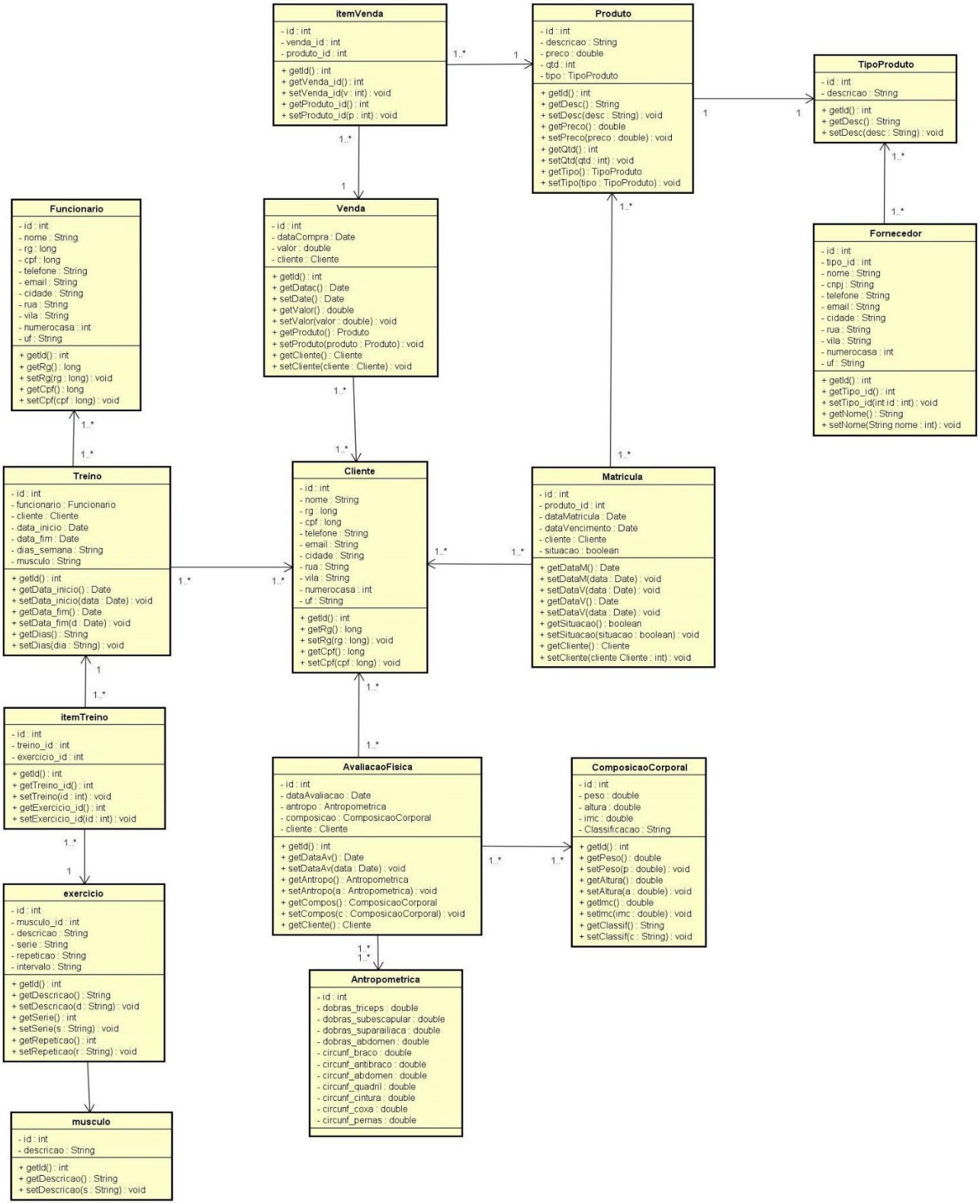

Figura 23: Diagrama de Classe.

3.5. DIAGRAMA DE ENTIDADE-RELACIONAMENTO

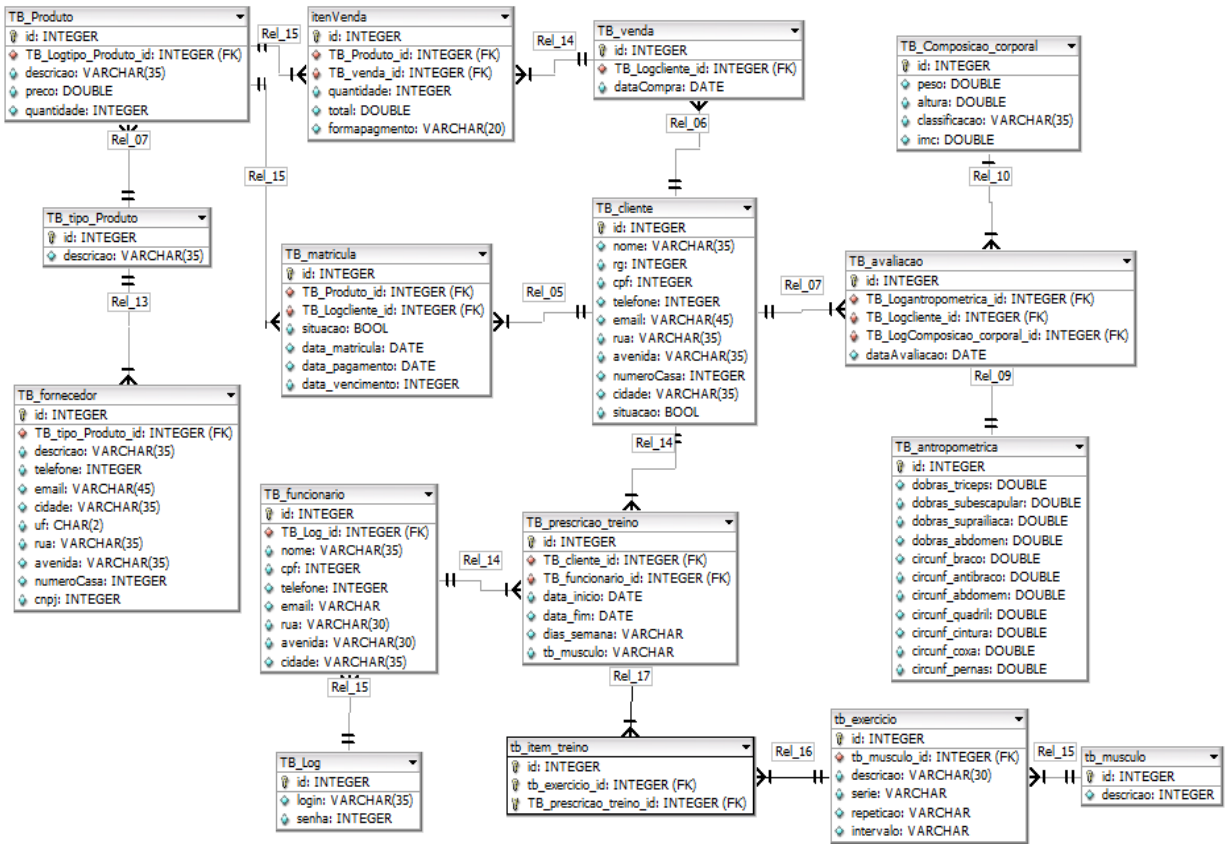

Figura 24: Diagrama de Entidade e Relacionamento.

4. CONCLUSÃO

Este sistema trará maior conforto e agilidade as recepções das academias, pois facilita todas as atividades contidas nos processos de uma recepção, tais como venda de produtos, realização de matriculas nas respectivas modalidades, controle do pagamento e validade das matriculas, avaliações físicas, criação treinos de acordo com o musculo desejado entre outras, com isso os gestores terão maior controle sobre a receita de seu negócio e mais informações de seus clientes.

E com a utilização de JavaFX as interfaces visuais se tronam mais atraentes aos olhos, sendo assim, o potencial de ganhar uma fatia de mercado aumenta, considerando todos os benefício do sistema a qualquer academia.

5. REFERÊNCIAS

ANTÔNIO, Marcos. **DBdesigner: uma ferramenta gratuita para a modelagem de banco de dados.**
Disponível em: < http://www.devmedia.com.br/dbdesigner-uma-ferramenta-gratuita-para-modelagem-de-dados-artigo-sql-magazine-35/6840 >
Acesso em: 12 nov. 2016.

CARVALHO, Claudio R. **Java Orientado a Objetos.**
Disponível em: < https://www.youtube.com/user/excriptvideo/playlists>
Acesso em: 13 jun. 2016.

DEITEL, Paul & DEITEL, Harvey. **Java como programar.** 8. Ed. São Paulo: Editora PEARSON, 2010.

DRUCKER, Peter. **Citações sobre Gestão Empresarial**.
Disponível em: https://pensador.uol.com.br/citacoes_sobre_gestao_empresarial/
Acesso em: 29 set. 2016.

FreeMind.
Disponível em: < https://pt.wikipedia.org/wiki/Freemind>
Acesso em: 12 nov. 2016.

GUANABARA, Gustavo. **Curso de Java para iniciantes.**
Disponível em: <https://www.youtube.com/user/cursosemvideo/playlists>
Acesso em: 10 set. 2016.

GUANABARA, Gustavo. **Curso de banco de dados MySQL.**
Disponível em: <https://www.youtube.com/user/cursosemvideo/playlists>
Acesso em: 10 set. 2016.

GUANABARA, Gustavo. **História do Java.**
Disponível em: < https://www.youtube.com/watch?v=sTX0UEplF54>
Acesso em: 12 nov. 2016.

GUANABARA, Gustavo. **Curso POO teoria #13ª – Conceito de Polimorfismo (Parte 2),** 2016.
Disponível em <https://www.youtube.com/watch?v=hYek1xqWzgs>
Acesso em: 31 jul. 2017

GUEDES, Gilleanes T. A. **UML 2: GUIA PRÁTICO,** 2. ed. São Paulo: Editora Novatec, 2007.

JASPERSOFT. **JasperResport library**.
Disponível em: < http://community.jaspersoft.com/project/jasperreports-library >
Acesso em: 8 nov. 2016

LANHELLAS, Ronaldo. **JasperReport: Relatórios em Java com iReport.**
Disponível em:< http://www.devmedia.com.br/jasperreport-relatorios-em-java-com-ireport/31075>
Acesso em: 12 nov. 2016.

MARCOS. **Diagrama de estrutura analítica do projeto**.
Disponível em: < http://partilho.com.br/aprender/eap/diagrama-de-estrutura-analitica-do-projeto-criacao-de-site/ >
Acesso em: 18 abr. 2017.

MARTINS, Marcelo. **Relatórios em Java – JasperReport e iReport.**
Disponível em:< http://www.k19.com.br/artigos/relatorios-em-java-jasperreports-e-irepor/ >
Acesso em: 12 nov. 2016.

PACIEVITCH, Yuri. **Software.**
Disponível em:< http://www.infoescola.com/informatica/software/>
Acesso em: 29 set. 2016.

PortalEducacao. **O que é MySQL?**
Disponível em:< https://www.portaleducacao.com.br/informatica/artigos/4398/o-que-e>
Acesso em:12 nov. 2016.

PROFISSIONAISTI. **Os principais diagramas da UML.**
Disponível em: <https://www.profissionaisti.com.br/2011/07/os-principais-diagramas-da-uml-resumo-rapido/ >
Acessado em: 18 abr. 2017.

RIBEIRO, Leandro. **O que é UML e diagramas de caso de uso**.
Disponível em: < http://www.devmedia.com.br/o-que-e-uml-e-diagramas-de-caso-de-uso-introducao-pratica-a-uml/23408 >
Acesso em: 18 abr. 2017.

SEABRA, Bruno. **O que é Astah Comunnity?**
Disponível em: <http://www.startupsstars.com/2015/10/o-que-e-o-astah-posttecnico-por-bruno-seabra/>
Acesso em: 12 nov. 2016.

TOSIN, Calos. **Java Avançado – JavaFX**.
Disponível em: < http://www2.softblue.com.br >
Acesso em: 16 mar. 2017.

VINÍCIUS, Thiago. **Conhecendo o Eclipse – Uma apresentação detalhada da IDE.**
Disponível em: < http://www.devmedia.com.br/conhecendo-o-eclipse-uma-apresentacao-detalhada-da-ide/25589>
Acesso em: 12 nov. 2016.