



Sistema de Ensino Presencial Conectado

Superior em Tecnologia em Análise e desenvolvimento de sistemas

RODRIGO GOTARDO

**Sistema FAZENTECH**

NORTE SHOPPING – PR

2020

NORTE SHOPPING – PR

2020

RODRIGO GOTARDO

Utilizando tecnologias atuais e modernas para sistemas delivery

**Sistema FAZENTECH**

Trabalho de produção textual interdisciplinar individual apresentado à Universidade Pitágoras UNOPAR, como requisito parcial para a obtenção de média semestral nas disciplinas de: Análise Orientada a Objetos II, Banco de Dados II, Programação para WebI, Programação Orientada a Objetos e Educação Ambiental

Orientador: Vandelice Dalto.

|  |
| --- |
|  |

|  |
| --- |
|  |

**SUMÁRIO**

[1 INTRODUÇÃO 3](#_Toc50293845)

[2 DESENVOLVIMENTO. 4](#_Toc50293846)

[2.1 ANÁLISE ORIENTADA A OBJETOS II 4](#_Toc50293847)

[2.2 BANCO DE DADOS II 10](#_Toc50293848)

[2.3 PROGRAMAÇÃO PARA WEB I 15](#_Toc50293849)

[2.4 PROGRAMAÇÃO ORIENTADA A OBJETOS 22](#_Toc50293850)

[3 CONCLUSÃO 26](#_Toc50293851)

# INTRODUÇÃO

O crescente aumento na Produção Agrícola e Pecuária, vem aumentando exponencialmente em nosso país, fazendo assim com que muitos agricultores atotem vários métodos e tecnologias no plantio e na criação dos animais, sendo necessário a implementação de vária tecnologias desde, o planejamento, a compra da matéria prima, plantio, manutenção da plantação, colheita e venda do produto final no caso da agricultura e no caso da Pecuário na escolha do melhor animal, do controle de doenças, a sua produção, abate e venda do produto final, faz necessário todo um sistema para gerenciar esses ciclos, tornando assim o mais eficiente possível e mais sustentável para o meio ambiente.

O Sistema FazenTECH, visa atender todas essas etapas auxiliando os agricultores e pecuaristas, a gerenciar o seu negocio de forma eficiente, prática, rentável e ainda sustentável com soluções inovadoras para o homem do campo.

# DESENVOLVIMENTO.

## ANÁLISE ORIENTADA A OBJETOS II

A Análise Orientada a Objetos (OOA) é um processo de desenvolvimento de sistemas que utiliza o conceito de objetos que interagem entre si e, através dessa inte4ração, realizam tarefas computacionais.

O ponto de partida para a OOA é criar um modelo descritivo contendo informações do projeto. Podemos chamar este artefato de proposta técnica, já que é ela que vai conter detalhes do problema a ser solucionado, objetivo do projeto, casos de uso, requisitos funcionais e não funcionais, as atividades do sistema e soluções propostas.

Durante todo ciclo de vida do desenvolvimento (análise e programação), são gerados alguns diagramas que representam os objetos de análise, sendo eles: diagrama de caso de uso, diagrama de classe, diagrama de objeto, diagrama de componente, diagrama de implantação, diagrama de atividade, diagrama de estado, diagrama de colaboração e diagrama de sequência.

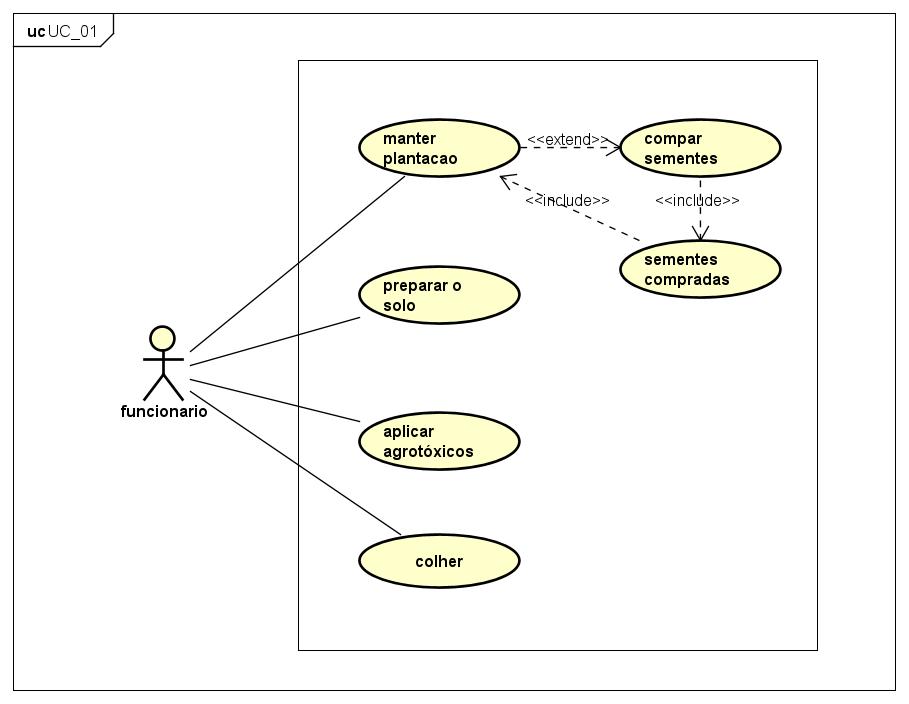
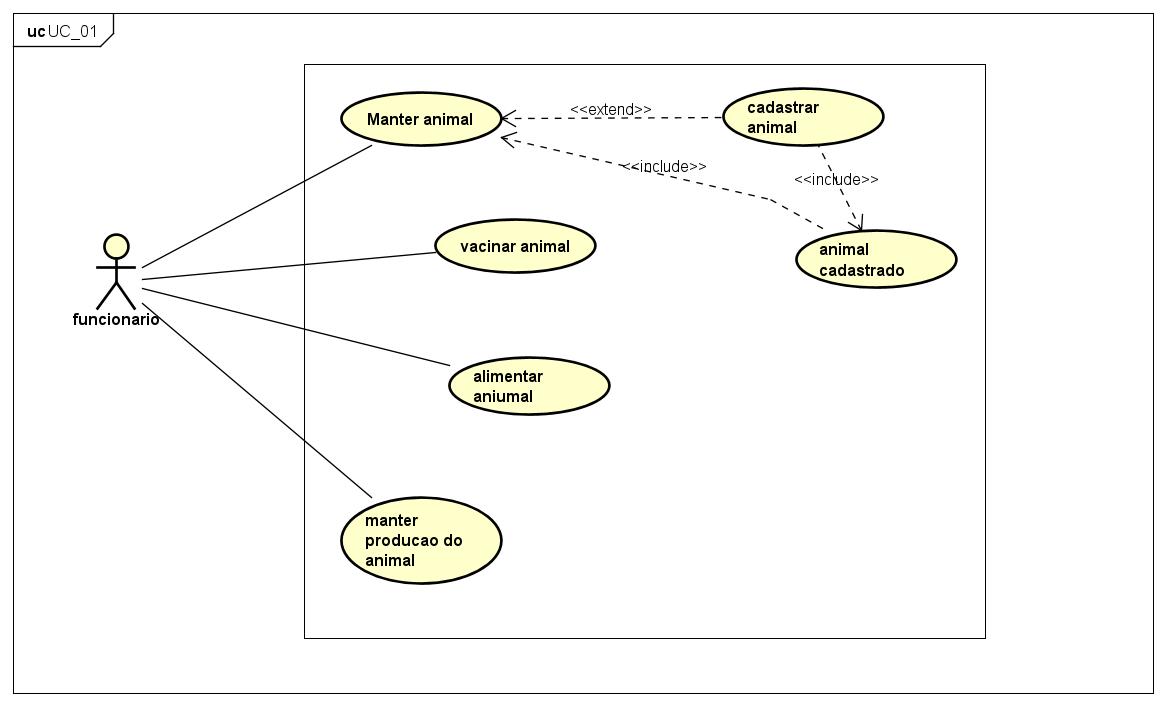
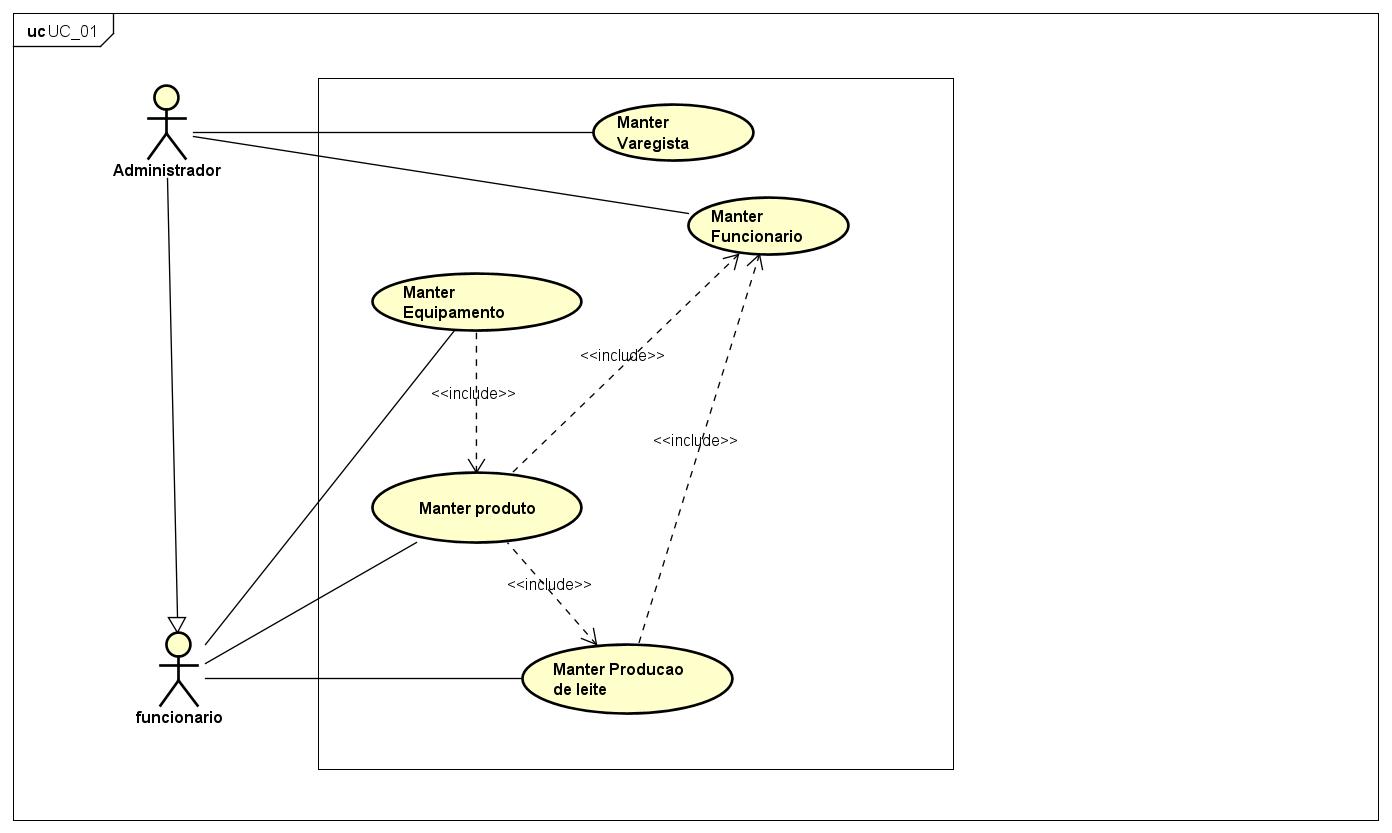
Analisando o estudo de caso apresentado para o PTI FazenTECH, e utilizando ferramentas CASE de modelagem juntamente com a UML, para realizar as atividades propostas para Análise de Sistemas, com os requisitos da criação animal e o planejamento de diversas culturas na fazenda.

Para desenvolver o sistema da FazenTech, na criação animal e no planejamento do plantio apresentamos os o Modelo de Casos de Uso, o Modedlo de Classes e o Diagrama de Máquina e Estados para a Classe “Plantio

1. Modelo de Use Cases:

O Diagrama de Use Case tem o objetivo de auxiliar a comunicação entre os analistas e o cliente, um Diagrama de Use Case descreve um cenário que mostra as funcionalidades do sistema do ponto de vista do usuário.

O Use Case é apresentado para representar os detalhes que compõem a sua estrutura, as notações usadas para representar graficamente o diagrama e usamos alguns exemplos a partir do estudo de caso proposto para solucionar o problema apresentado.



**Figura 1: Diagrama de Caso de Uso**

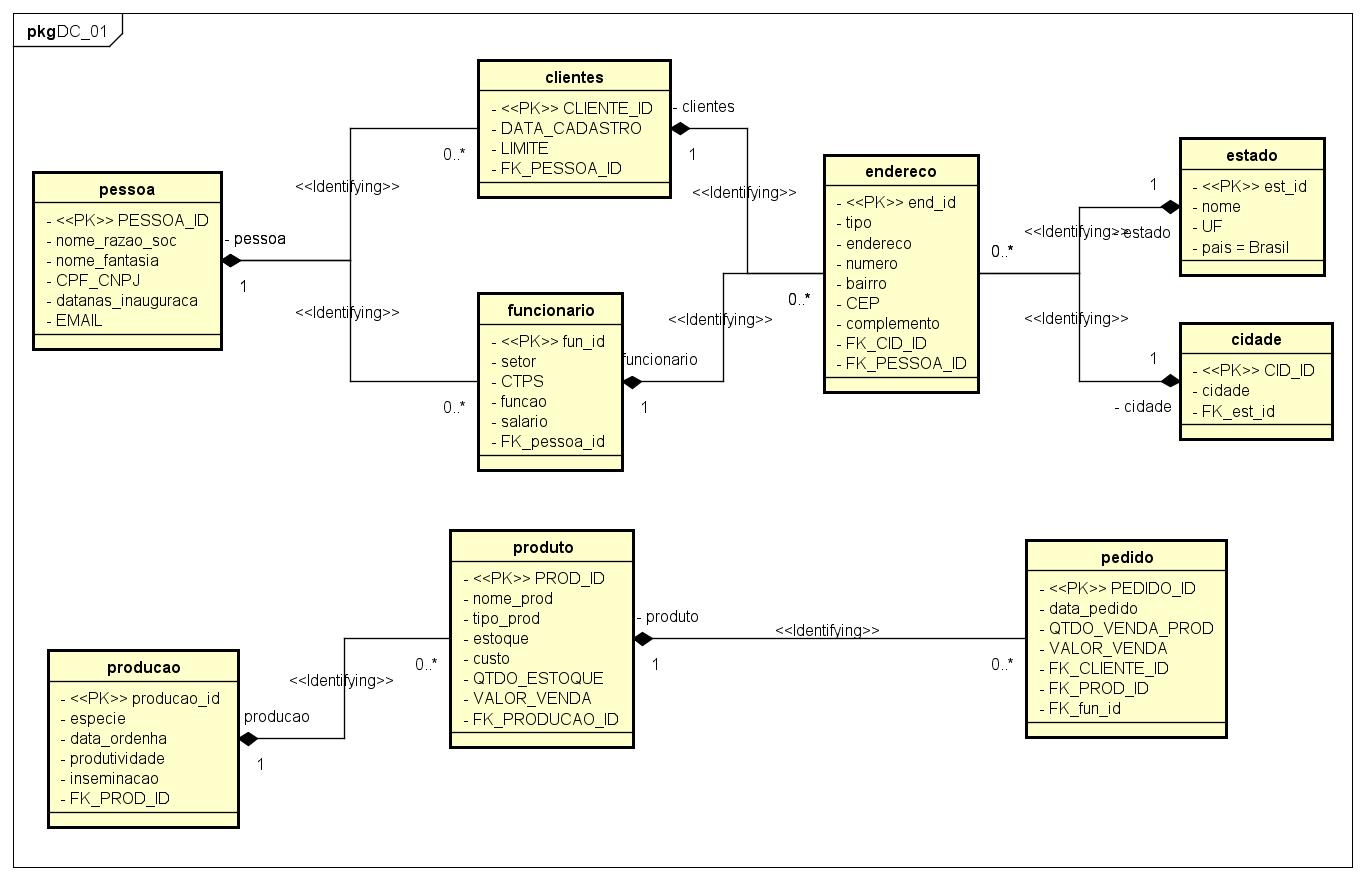
1. Modelo de Classes:

Um diagrama de classes modela a estrutura estática de um sistema.

Em um processo de desenvolvimento iterativo, a análise precede o projeto, entretanto, a análise e o projeto podem acontecer simultaneamente. Segundo Bezerra (2007, p 176), “os modelos especificados na análise esclarecem o problema a ser resolvido. No entanto, as perspectivas do sistema fornecidas por esses modelos não são suficientes para se ter uma visa o completa do sistema para que a implementação comece”.

O Diagrama de Classes é uma representação da estrutura e relações das classes que servem de modelo para objetos. Podemos afirmar de maneira simples que será um conjunto de objetos com a mesma característica, assim saberemos identificar objetos e agrupá-los, de forma a encontrar suas respectivas classes. Uma classe é representada por um retângulo com três divisões.

**Figura 2: Diagrame de Classe**

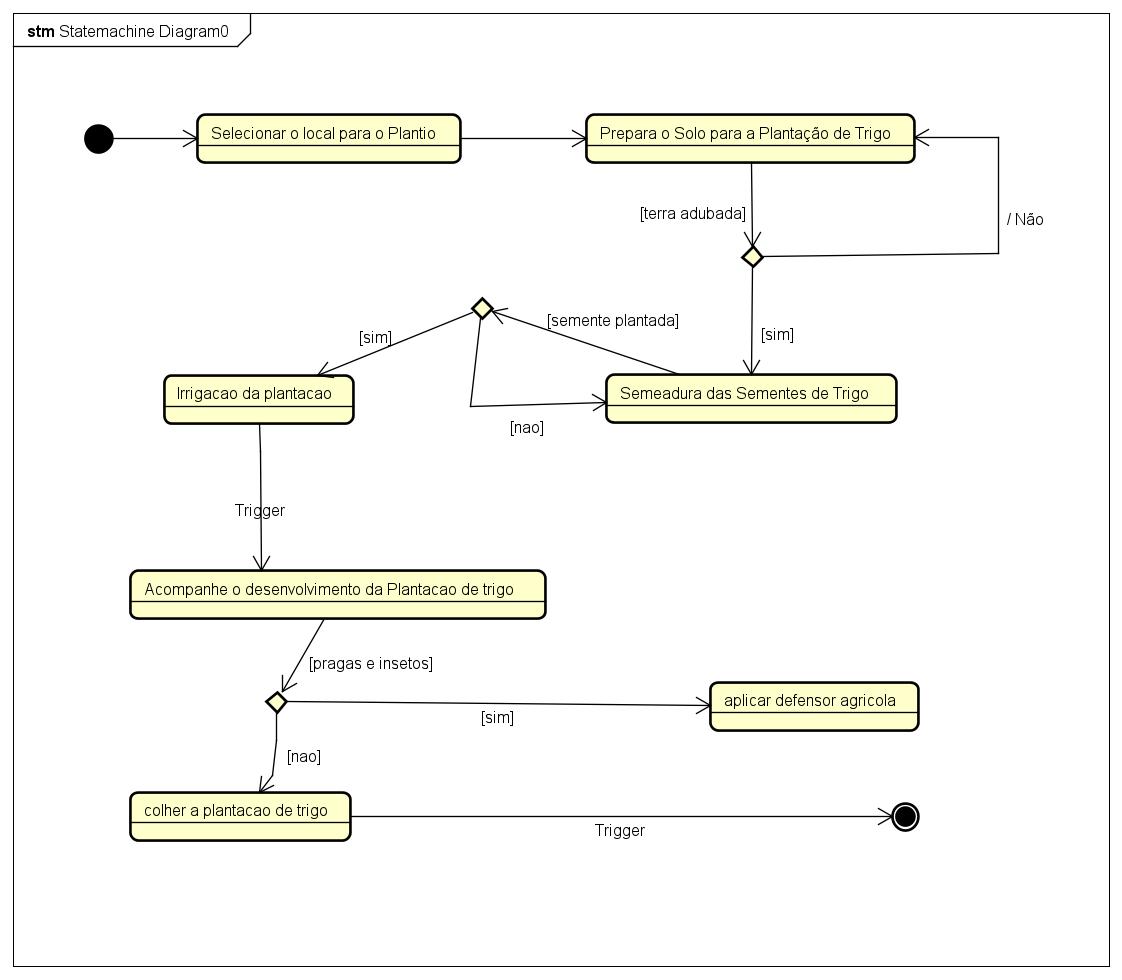
****

1. O Diagrama de Máquina de Estados:

São usados ​​para capturar o comportamento de um sistema de

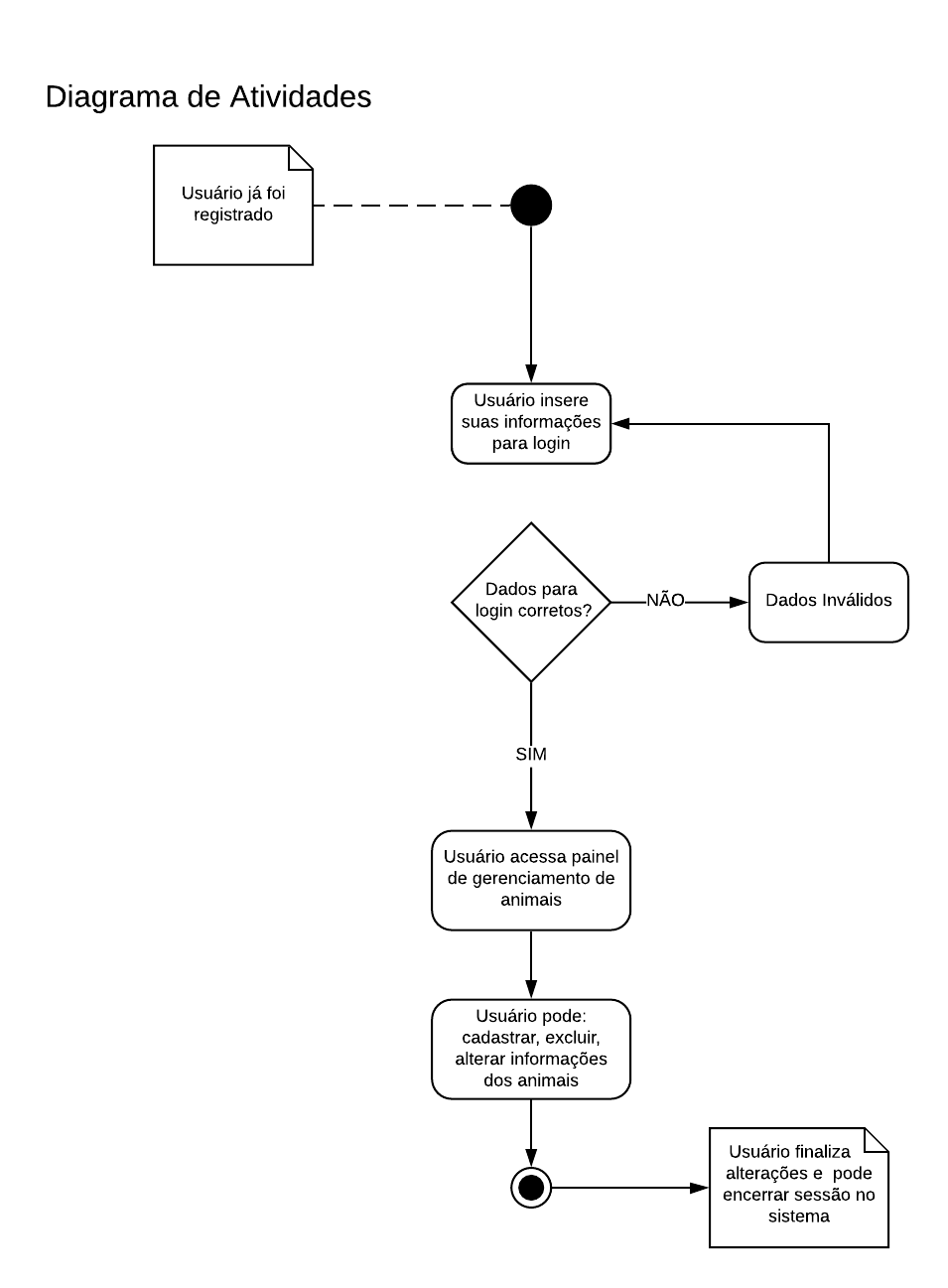
São utilizados para modelar o comportamento dinâmico de um elemento de modelo e, mais especificamente, os apectos direcionados a eventos do comportameento do sistema.

Os diagramas de máquina de estado UML podem ser usados ​​para modelar o comportamento de uma classe, um subsistema, um pacote ou até mesmo um sistema inteiro. É também denominado diagrama de estatísticas ou de transição de estados.

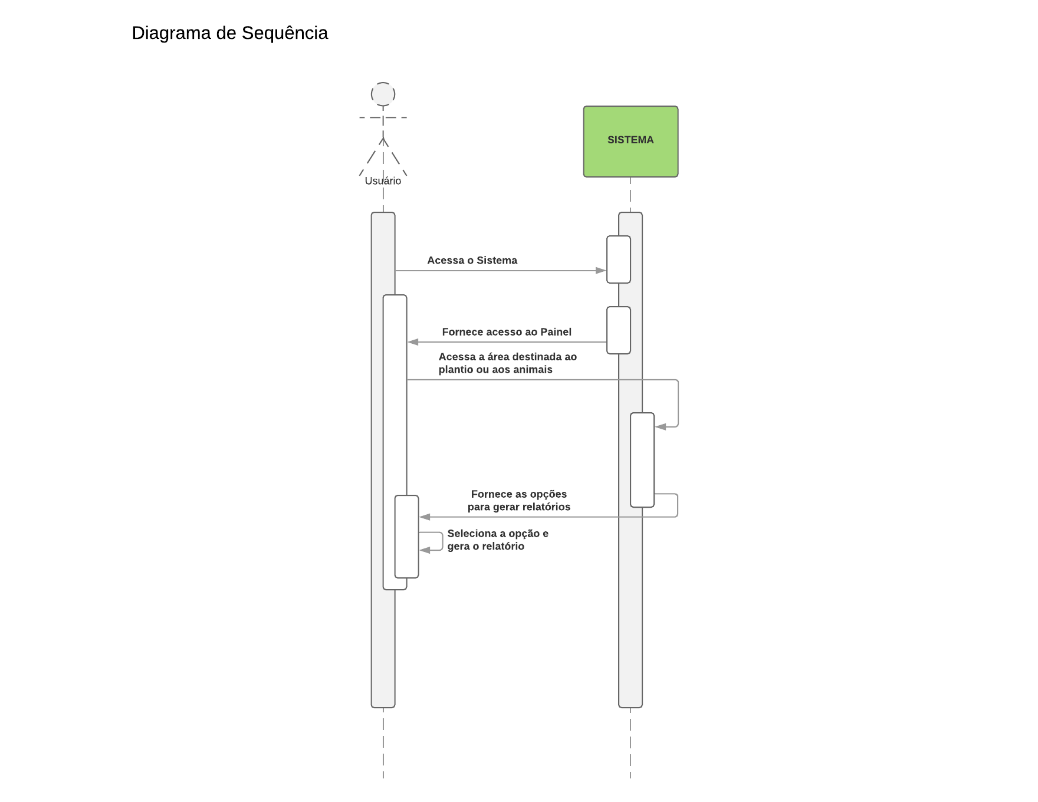
O elemento modelado muitas vezes é uma instancia de uma classe, no entanto, poode se usar esse diagrama para modelar o comportamento de um case de uso, o comportamento de um dado durante uma transação ou mesmo o comportamento de um sistema completo, de acordo com Bezerra (2007, p.287),”o Diagrama de eMáquina de Estados, permite descrever o ciclo de vida de objetos de uma classe, os eventos que causam a transição de um estado para outro e a realização de operações resultantes”. E para complementar, segundo Booch, Rumbaugh e Jacobson (2006, o.288), “uma Máquina de Estados é um comportamento que especiofica as sequências de estados pelas quais um objeto passa durante seu tempo de via em resposta a eventos, juntamente com suas respostas e esses ****eventos”.

**Figura 3: Diagrama de Máquina de Estados**

1. Diagrama de Atividades:

**Figura 4: Diagrama de atividades**

1. Diagrama de Sequência:

Os diagramas de sequência descrevem as interações entre as classes em termos de troca de mensagens ao longo do tempo. Eles também são chamados de diagramas de eventos. Um diagrama de sequência é uma boa maneira de visualizar e validar vários cenários de tempo de execução. Isso pode ajudar a prever como um sistema se comportará e a descobrir as responsabilidades que uma classe pode precisar ter no processo de modelagem de um novo sistema.

**Figura 5: Diagrama de sequencia**

## BANCO DE DADOS II

Banco de dados, também chamado de banco de dados eletrônico, qualquer coleção de dados, ou informação, que é especialmente organizada para rápida pesquisa e recuperação por um computador. Os bancos de dados são estruturados para facilitar o armazenamento, recuperação, modificação e exclusão de dados em conjunto com várias operações de processamento de dados. Um sistema de gerenciamento de banco de dados (DBMS) extrai informações do banco de dados em resposta à consulta.

Um banco de dados é armazenado como um arquivo ou um conjunto de arquivos. As informações nesses arquivos podem ser divididas em registros, cada um dos quais consiste em um ou mais campos. Os campos são as unidades básicas de armazenamento de dados e cada campo geralmente contém informações pertencentes a um aspecto ou atributo da entidade descrita pelo banco de dados. Os registros também são organizados em tabelas que incluem informações sobre as relações entre seus vários campos. Embora o banco de dados seja aplicado livremente a qualquer coleção de informações em arquivos de computador, um banco de dados no sentido estrito fornece recursos de referência cruzada. Usando palavras-chave e vários comandos de classificação, os usuários podem rapidamente pesquisar, reorganizar, agrupar e selecionar os campos em muitos registros para recuperar ou criar relatórios sobre agregados de dados específicos.

Os registros e arquivos do banco de dados devem ser organizados para permitir a recuperação das informações. As consultas são a principal forma de os usuários recuperarem informações do banco de dados. O poder de um SGBD vem de sua capacidade de definir novos relacionamentos a partir dos básicos fornecidos pelas tabelas e de usá-los para obter respostas a consultas.

Os muitos usuários de um grande banco de dados devem ser capazes de manipular as informações contidas nele rapidamente a qualquer momento. Além disso, grandes empresas e outras organizações tendem a criar muitos arquivos independentes contendo dados relacionados e até mesmo sobrepostos, e suas atividades de processamento de dados frequentemente requerem a vinculação de dados de vários arquivos. Vários tipos diferentes de SGBD foram desenvolvidos para oferecer suporte a esses requisitos: simples, hierárquico, de rede, relacional e orientado a objetos.

Para o FazenTECH utilizamos o MySQL para criar um script contendo comandos DDL, DML e DQL, conforme será demostrado abaixo:

|  |
| --- |
| CREATE DATABASE `fazenda-bd`;  CREATE TABLE `Compras` (  `id` int(11) NOT NULL,  `nome` varchar(255) COLLATE utf8\_unicode\_ci DEFAULT NULL,  `produto` varchar(255) COLLATE utf8\_unicode\_ci DEFAULT NULL,  `qtd` int(11) DEFAULT NULL,  `dt\_compra` date DEFAULT NULL  ) ENGINE=InnoDB DEFAULT CHARSET=utf8 COLLATE=utf8\_unicode\_ci;  INSERT INTO `Compras` (`id`, `nome`, `produto`, `qtd`, `dt\_compra`) VALUES  (1, 'Fazenda Sao Francisco', 'sementes de girassol', 900, '2020-08-09'),  (2, 'Comunidade Rural Apaga Fogo', 'sementes de girassol', 1200, '2020-07-14');  CREATE TABLE `Equipamentos` (  `id` int(11) NOT NULL,  `nome` varchar(255) COLLATE utf8\_unicode\_ci DEFAULT NULL,  `tipo` varchar(255) COLLATE utf8\_unicode\_ci DEFAULT NULL  ) ENGINE=InnoDB DEFAULT CHARSET=utf8 COLLATE=utf8\_unicode\_ci;  INSERT INTO `Equipamentos` (`id`, `nome`, `tipo`) VALUES  (1, 'colheitadeira', 'motorizado'),  (2, 'ceifadora', 'motorizado');  CREATE TABLE `Funcionarios` (  `id` int(11) NOT NULL,  `nome` varchar(255) COLLATE utf8\_unicode\_ci DEFAULT NULL,  `cpf` varchar(11) COLLATE utf8\_unicode\_ci DEFAULT NULL,  `salario` varchar(15) COLLATE utf8\_unicode\_ci DEFAULT NULL  ) ENGINE=InnoDB DEFAULT CHARSET=utf8 COLLATE=utf8\_unicode\_ci;  INSERT INTO `Funcionarios` (`id`, `nome`, `cpf`, `salario`) VALUES  (1, 'Roberto Carlos', '53698702134', '1040'),  (2, 'Raul Seixas', '40236902349', '1040'),  (3, 'Elvis Presley', '43749314539', '2650'),  (4, 'Hebert Viana', '33793454294', '2650');  CREATE TABLE `Produc\_Leite` (  `id` int(11) NOT NULL,  `especie` varchar(255) COLLATE utf8\_unicode\_ci DEFAULT NULL,  `data\_ordenha` date DEFAULT NULL,  `temp\_leite` int(11) DEFAULT NULL,  `produtividade` int(11) DEFAULT NULL,  `inseminacao` varchar(3) COLLATE utf8\_unicode\_ci DEFAULT NULL,  `est\_parto` date DEFAULT NULL,  `secagem` date DEFAULT NULL,  `mm\_rumina` int(11) DEFAULT NULL  ) ENGINE=InnoDB DEFAULT CHARSET=utf8 COLLATE=utf8\_unicode\_ci;  INSERT INTO `Produc\_Leite` (`id`, `especie`, `data\_ordenha`, `temp\_leite`, `produtividade`, `inseminacao`, `est\_parto`, `secagem`, `mm\_rumina`) VALUES  (1, 'marina', '2020-07-13', 33, 1500, 'nao', '2020-09-25', '2021-04-29', 3600),  (2, 'leiteira', '2020-05-22', 39, 2600, 'nao', '2021-02-13', '2021-10-15', 3600);  CREATE TABLE `Produtos` (  `id` int(11) NOT NULL,  `nome` varchar(255) COLLATE utf8\_unicode\_ci NOT NULL,  `tipo` varchar(255) COLLATE utf8\_unicode\_ci DEFAULT NULL,  `qtd\_estoque` int(11) DEFAULT NULL,  `preco` float DEFAULT NULL  ) ENGINE=InnoDB DEFAULT CHARSET=utf8 COLLATE=utf8\_unicode\_ci;  INSERT INTO `Produtos` (`id`, `nome`, `tipo`, `qtd\_estoque`, `preco`) VALUES  (1, 'sementes de girassol', 'sementes', 500, 20),  (2, 'enxada', 'material', 300, 70);  CREATE TABLE `Varejistas` (  `id` int(11) NOT NULL,  `nome` varchar(255) COLLATE utf8\_unicode\_ci DEFAULT NULL,  `ult\_compra` date DEFAULT NULL  ) ENGINE=InnoDB DEFAULT CHARSET=utf8 COLLATE=utf8\_unicode\_ci;  INSERT INTO `Varejistas` (`id`, `nome`, `ult\_compra`) VALUES  (1, 'Fazenda Sao Francisco', '2020-01-23'),  (2, 'Comunidade Rural Apaga Fogo', '2020-04-15');  ALTER TABLE `Compras`  ADD PRIMARY KEY (`id`);  ALTER TABLE `Equipamentos`  ADD PRIMARY KEY (`id`);  ALTER TABLE `Funcionarios`  ADD PRIMARY KEY (`id`);  ALTER TABLE `Produc\_Leite`  ADD PRIMARY KEY (`id`);  ALTER TABLE `Produtos`  ADD PRIMARY KEY (`id`);  ALTER TABLE `Varejistas`  ADD PRIMARY KEY (`id`);  ALTER TABLE `Compras`  MODIFY `id` int(11) NOT NULL AUTO\_INCREMENT, AUTO\_INCREMENT=3;  ALTER TABLE `Equipamentos`  MODIFY `id` int(11) NOT NULL AUTO\_INCREMENT, AUTO\_INCREMENT=3;  ALTER TABLE `Funcionarios`  MODIFY `id` int(11) NOT NULL AUTO\_INCREMENT, AUTO\_INCREMENT=5;  ALTER TABLE `Produc\_Leite`  MODIFY `id` int(11) NOT NULL AUTO\_INCREMENT, AUTO\_INCREMENT=3;  ALTER TABLE `Produtos`  MODIFY `id` int(11) NOT NULL AUTO\_INCREMENT, AUTO\_INCREMENT=3;  ALTER TABLE `Varejistas`  MODIFY `id` int(11) NOT NULL AUTO\_INCREMENT, AUTO\_INCREMENT=3;  COMMIT;  [SELECT](https://br140.hostgator.com.br:2083/cpsess6136946034/3rdparty/phpMyAdmin/url.php?url=https://dev.mysql.com/doc/refman/5.5/en/select.html) A.nome, B.nome FROM Compras as A INNER JOIN Varejistas as B on A.nome = B.nome;  [SELECT](https://br140.hostgator.com.br:2083/cpsess6136946034/3rdparty/phpMyAdmin/url.php?url=https://dev.mysql.com/doc/refman/5.5/en/select.html) [COUNT](https://br140.hostgator.com.br:2083/cpsess6136946034/3rdparty/phpMyAdmin/url.php?url=https://dev.mysql.com/doc/refman/5.5/en/group-by-functions.html#function_count)(\*) FROM Funcionarios;  [SELECT](https://br140.hostgator.com.br:2083/cpsess6136946034/3rdparty/phpMyAdmin/url.php?url=https://dev.mysql.com/doc/refman/5.5/en/select.html) [COUNT](https://br140.hostgator.com.br:2083/cpsess6136946034/3rdparty/phpMyAdmin/url.php?url=https://dev.mysql.com/doc/refman/5.5/en/group-by-functions.html#function_count)(DISTINCT nome) FROM Compras;  [SELECT](https://br140.hostgator.com.br:2083/cpsess6136946034/3rdparty/phpMyAdmin/url.php?url=https://dev.mysql.com/doc/refman/5.5/en/select.html) [MAX](https://br140.hostgator.com.br:2083/cpsess6136946034/3rdparty/phpMyAdmin/url.php?url=https://dev.mysql.com/doc/refman/5.5/en/group-by-functions.html#function_max)(salario) FROM Funcionarios; |

**Quadro 1: Script SQL**

## PROGRAMAÇÃO PARA WEB I

Para minimizar as perdas na produção leite que ocorrem na fazenda de Lúcia foi idealizado um sistema de controle de produção utilizando HTML e CSS.

Apresentamos então parte da interface de um sistema simples:



**Figura 6: Tela inicial Sistema FazenTECH**

  
**Figura 7: Tela Cadastro de Animais**

Código HTML+CSS do Cadastro de Animais:

|  |
| --- |
| <!DOCTYPE html>  <html lang="en">  <head>  <meta charset="utf-8">  <meta http-equiv="X-UA-Compatible" content="IE=edge">  <meta name="viewport" content="width=device-width, initial-scale=1">  <meta name="description" content="">  <meta name="author" content="">  <title>Sistema FazenTECH</title>  <link href="css/site.css" rel="stylesheet">  <link rel="stylesheet" type="text/css" href="css/demo.css" />  <link rel="stylesheet" type="text/css" href="css/style.css" />  <link rel="stylesheet" type="text/css" href="css/animate-custom.css" />  <!-- Bootstrap Core CSS -->  <link href="css/bootstrap.min.css" rel="stylesheet">  <!-- Custom CSS -->  <link href="css/simple-sidebar.css" rel="stylesheet">  <!-- HTML5 Shim and Respond.js IE8 support of HTML5 elements and media queries -->  <!-- WARNING: Respond.js doesn't work if you view the page via file:// -->  <!--[if lt IE 9]>  <script src="https://oss.maxcdn.com/libs/html5shiv/3.7.0/html5shiv.js"></script>  <script src="https://oss.maxcdn.com/libs/respond.js/1.4.2/respond.min.js"></script>  <![endif]-->  </head>  <body>  <div id="wrapper">  <!-- Sidebar -->  <div id="sidebar-wrapper">  <ul class="sidebar-nav">  <li class="sidebar-brand">  <a href="index.html">  Sistema FazenTECH  </a>  </li>  <li>  <a href="cadastroa.html">Cadastro Animais</a>  </li>  <li>  <a href="cadastroc.html">Cadastro Clientes</a>  </li>  <li>  <a href="cadastrop.html">Cadastro Produtos</a>  </li>  <li>  <a href="cadastrof.html">Cadastro Funcionarios</a>  </li>  <li>  <a href="cadastroe.html">Cadastro Equipamentos</a>  </li>  <li>  <a href="cadastrov.html">Cadastro Varejistas</a>  </li>      </ul>    </div>  <!-- /#sidebar-wrapper -->  <!-- Page Content -->  <div id="page-content-wrapper">  <a href="#menu-toggle" class="btn btn-default" id="menu-toggle">Esconder/Mostrar Menu</a>  <div class="container-fluid">  <div class="row">  <div class="col-lg-12">    <!--Título-->  <h1>Cadastro de Animais</h1>  <div class="form-group"><form id="w0" action="novo\_animal.html" method="post">  <button type="submit" class="btn btn-success">Novo Animal</button>  </form>  </div>        <div class="col-lg-12">    <div class="table-responsive">    <table class="table table-bordered table-hover">  <thead>  <tr class="warning">    <th>Código Numérico</th>  <th>Qtd leite/dia</th>  <th>Data da última ordenha</th>  <th>Ordenhado por</th>  <th>Horário da última ordenha</th>  <th>Doente</th>  <th>Tratamento</th>  <th>Dias para tratamento</th>  <th>AÇÕES</th>  </tr>  </thead>  <tbody>  <tr>  <td>000563843</td>  <td>23 litros</td>  <td>06/09/2020</td>  <td>Tonho da Lua</td>  <td>5:15</td>  <td>Não</td>  <td>Não se aplica</td>  <td>Não se aplica</td>  <td><a href="editara.html">Editar</a> / <a href="excluira.html">Excluir</a></td>  </tr>    <tr>  <td>100566543</td>  <td>21 litros</td>  <td>06/09/2020</td>  <td>Germano</td>  <td>5:10</td>  <td>Sim</td>  <td>Aceturato de dimenazene</td>  <td>5</td>  <td><a href="editara.html">Editar</a> / <a href="excluira.html">Excluir</a></td>  </tr>    <tr>  <td>000563379</td>  <td>20 litros</td>  <td>06/09/2020</td>  <td>Quaresma</td>  <td>5:15</td>  <td>Não</td>  <td>Não se aplica</td>  <td>Não se aplica</td>  <td><a href="editara.html">Editar</a> / <a href="excluira.html">Excluir</a></td>  </tr>    <tr>  <td>100563843</td>  <td>18 litros</td>  <td>26/08/2020</td>  <td>Zé Banha</td>  <td>6:30</td>  <td>Sim</td>  <td>Enfrent Ourofino</td>  <td>1</td>  <td><a href="editara.html">Editar</a> / <a href="excluira.html">Excluir</a></td>  </tr>    <tr>  <td>100325843</td>  <td>10 litros</td>  <td>25/08/2020</td>  <td>Tonho da Lua</td>  <td>5:30</td>  <td>Sim</td>  <td>Vacina Poli-Star</td>  <td>31</td>  <td><a href="editara.html">Editar</a> / <a href="excluira.html">Excluir</a></td>  </tr>  </tbody>  <nav aria-label="...">  <ul class="pagination">  <li class="page-item disabled">  <a class="page-link" href="#" tabindex="-1">Anterior</a>  </li>  <li class="page-item active">  <a class="page-link" href="#">1 <span class="sr-only">(atual)</span></a>  </li>  <li class="page-item"><a class="page-link" href="#">2</a></li>  <li class="page-item"><a class="page-link" href="#">3</a></li>  <li class="page-item">  <a class="page-link" href="#">Próximo</a>  </li>  </ul>  </nav>  </table>    </div>  </div>          </div>      </div>        </div>  <!-- /#page-content-wrapper -->  </div>  <!-- /#wrapper -->  <!-- jQuery -->  <script src="js/jquery.js"></script>  <!-- Bootstrap Core JavaScript -->  <script src="js/bootstrap.min.js"></script>  <!-- Menu Toggle Script -->  <script>  $("#menu-toggle").click(function(e) {  e.preventDefault();  $("#wrapper").toggleClass("toggled");  });  </script>  </body>  </html> |

**Quadro 2: HTML + CSS**

## PROGRAMAÇÃO ORIENTADA A OBJETOS

Como o nome sugere, Programação orientada a objetos ou POO se refere a linguagens que usam objetos na programação. A programação orientada a objetos tem como objetivo implementar entidades do mundo real como herança, ocultação, polimorfismo, etc. na programação. O principal objetivo da POO é vincular os dados e as funções que operam neles de modo que nenhuma outra parte do código possa acessar esses dados, exceto aquela função.

Existem alguns conceitos quando se fala de Programação Orientada a Objetos:

**Polimorfismo**: polimorfismo se refere à capacidade das linguagens de programação POOs de diferenciar entidades com o mesmo nome de forma eficiente. Isso é feito pelo Java com a ajuda da assinatura e declaração dessas entidades.

**Herança**: a herança é um pilar importante da POO (Programação Orientada a Objetos). É o mecanismo em java pelo qual uma classe pode herdar os recursos (campos e métodos) de outra classe.

**Encapsulamento**: O encapsulamento é definido como o agrupamento de dados em uma única unidade. É o mecanismo que une o código e os dados que ele manipula. Outra forma de pensar sobre o encapsulamento é que ele é um escudo protetor que evita que os dados sejam acessados ​​pelo código fora desse escudo.

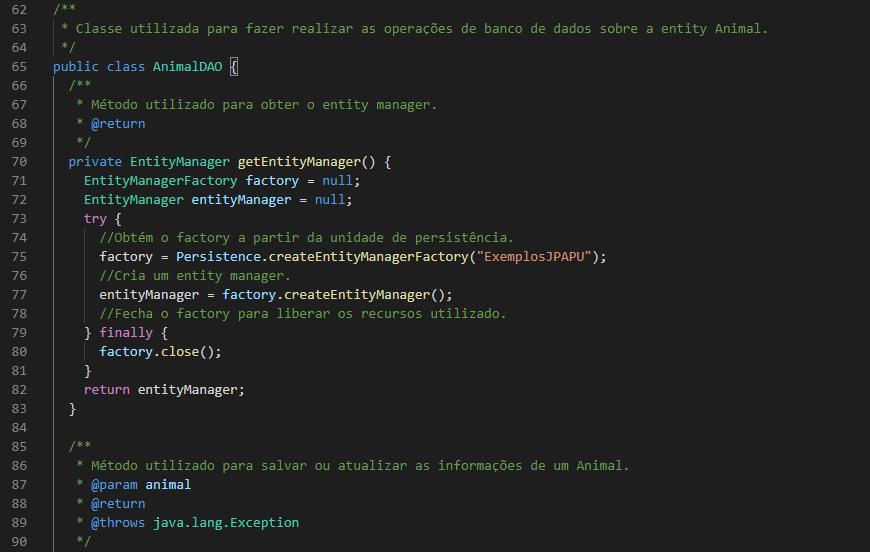
**Abstração**: A abstração de dados é a propriedade em virtude da qual apenas os detalhes essenciais são exibidos para o usuário. As unidades triviais ou não essenciais não são exibidas para o usuário. Ex: um carro é visto como um carro ao invés de seus componentes individuais.

A abstração de dados também pode ser definida como o processo de identificar apenas as características necessárias de um objeto, ignorando os detalhes irrelevantes. As propriedades e comportamentos de um objeto o diferenciam de outros objetos de tipo semelhante e também ajudam na classificação / agrupamento dos objetos.

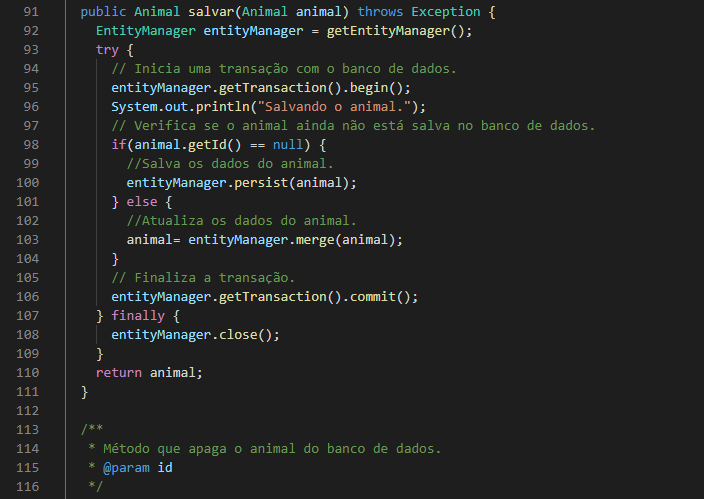
**Método**: um método é uma coleção de instruções que executam alguma tarefa específica e retornam o resultado ao chamador. Um método pode realizar alguma tarefa específica sem retornar nada. Os métodos nos permitem reutilizar o código sem redigitá-lo. Em Java, todo método deve fazer parte de alguma classe diferente de linguagens como C, C ++ e Python.

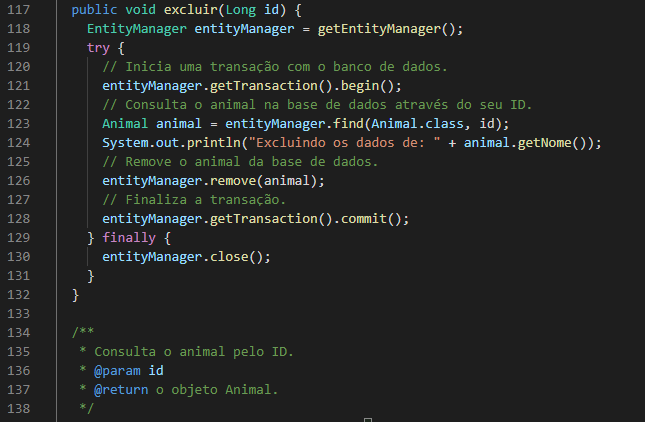
**Objeto**: É uma unidade básica de Programação Orientada a Objetos e representa as entidades da vida real. Um programa Java típico cria muitos objetos, que, como você sabe, interagem chamando métodos.

Para o Sistema FazenTECH foi construído um Banco de Dados denominado “fazenda-db”, para armazenar as informações mais importantes e criado uma estrutura de classes com seus respectivos métodos para cadastro e manutenção dos dados no FazenTECH:

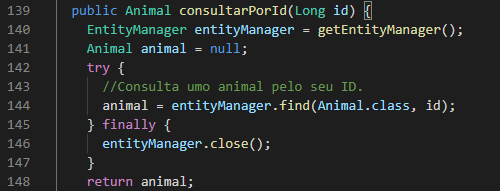


**Figura 8: Classe AnimalDAO – parte 1**

**Figura 9: Classe AnimalDAO – parte 2**

****

**Figura 9: Classe AnimalDAO – parte 3**



**Figura 10: Classe AnimalDAO – parte 4**

# CONCLUSÃO

Com a criação do trabalho proposto, assimilar alguns conceitos antes só conhecidos na teoria acadêmica.

A Programação Orientada a Objetos segue uma estrutura onde é possível reaproveitar código e dar segurança a nossa aplicação.

Quando abordamos desenvolvimento e modelagem de sistemas, podemos ser apoiados pela UML, com seus diagramas que nos permite representar várias ocasiões possíveis de nosso sistema, e também fazer a coleta dos requisitos.

Foi possível entendermos que adotar um modelo de processo de desenvolvimento de software é uma tarefa indispensável para termos um bom gerenciamento de cada etapa que venham a ocorrer em nosso trabalho.

###### REFERÊNCIAS

PERINI, Luís Cláudio; HISATOMI, Marco Ikuro; BERTO, Wagner Luiz. **Engenharia de Software**. São Paulo: Pearson Prentice Hall, 2009.

FREITAS, Veronice. **Programação Web II**. São Paulo: Pearson Education do Brasil, 2013.

PRESSMAN, R.; MAXIM, B. **Engenharia de software: uma abordagem profissional**. 8 ed. Porto Alegre: AMGH, 2016. Disponível em < https://integrada.minhabiblioteca.com.br/books/9788580555349> Acesso: em 06 set. 2020.

SANTANA, Gisele Alves. **Linguagens de programação e estruturas de dados**. Londrina: Editora e Distribuidora Educacional S.A. 2018.