



FUNDAÇÃO EDUCACIONAL DE FERNANDÓPOLIS
FACULDADES INTEGRADAS DE FERNANDÓPOLIS
DEPARTAMENTO DE SISTEMAS DE INFORMAÇÃO

José Pedro Bezerra Riva

S.I.G. PETSHOP

Sistema Integrado de Gerenciamento de Petshop

Fernandópolis - SP

2024



José Pedro Bezerra Riva

Projeto de Estágio Supervisionado apresentado às
Faculdades Integradas de Fernandópolis como
requisito parcial para a obtenção do título de bacharel
em Sistemas de Informação.

Orientador: Prof. Jefferson Antonio Ribeiro Passerini

Fernandópolis - SP

2024

DEDICATÓRIA

Dedicamos esse trabalho aos nossos professores, colegas de classe e a todos os profissionais que contribuíram para a nossa evolução acadêmica. Ao nosso orientador Jefferson Antonio Ribeiro Passerini pela paciência e incentivo para que tornássemos possível a conclusão desse projeto.

EPÍGRAFE

“É muito melhor arriscar coisas grandiosas,
alcançar triunfos e glórias, mesmo expondo-se
à derrota, do que formar filas com os pobres de
espírito que nem gozam muito e nem sofrem
muito, porque vivem nessa penumbra cinzenta
que não conhece nem vitória nem derrota.”
(THEODORE ROOSEVELT)

LISTA DE ILUSTRAÇÃO

Figura 1 - Diagrama de Atores – S.I.G. PETCHOP.....	17
Figura 2 - Diagrama de Contexto Geral - Módulo Funcionário.....	26
Figura 3 - Diagrama de Contexto Geral - Módulo Cliente.....	27
Figura 4 - DCU Individual Ator Funcionário - Realizar login.....	28
Figura 5 - DCU Individual Ator Funcionário - Cadastro.....	30
Figura 6 - DCU Individual Ator Funcionário - Consulta Pessoa.....	31
Figura 7 - DCU Individual Ator Funcionário - Altera Pessoa.....	33
Figura 8 - DCU Individual Ator Funcionário - Carregar Pessoa.....	34
Figura 9 - DCU Individual Ator Funcionário - Inativar.....	35
Figura 10 - Controller CadastrarFuncionario.....	43
Figura 11 - Controller AlterarFuncionario.....	44
Figura 12 - Controller PesquisarFuncionario.....	44
Figura 13 - DAO Classe Pessoa.....	46
Figura 14 - DAO Classe TipoFuncionario.....	47
Figura 15 - DAO TipoCliente.....	48
Figura 16 - DAO Animal.....	48
Figura 17 - DAO Serviço.....	49
Figura 18 - DAO TipoServiço.....	49
Figura 19 - DAO Agendamento.....	50
Figura 20 - DER.....	51
Figura 21-Tela Login.....	53
Figura 22 - Tela Cadastro Cliente.....	54
Figura 23 - Tela Cadastro Empregado.....	54
Figura 25 - Tela de Cadastro de Animais.....	56
Figura 26 - Tela de Listagem Animais.....	56
Figura 27 - Tela de Listagem De Clientes.....	57
Figura 28 - Listagem de Empregados.....	57
Figura 29 - Tela de Listagem de Serviços.....	58
Figura 30 - DCIHM - Cadastrar Cliente.....	59
Figura 31 - DCIHM - Listar Cliente.....	60
Figura 32 - DCIHM - Listar Serviço.....	61
Figura 33 - Sequência Login Cliente.....	62
Figura 34 - Login Empregado.....	63
Figura 35 - Cadastrar Pessoa(Cliente/Funcionário).....	64
Figura 36 - Sequência Listar Cliente/Funcionário.....	65



Figura 37 - Diagrama de implantação de Hardware.....	66
Figura 38 - Logo IntelliJ.....	68
Figura 39 - Logo Java.....	68
Figura 40 - Spring.....	69
Figura 41 - HTML5.....	69
Figura 42 - CSS.....	70
Figura 43 - VS CODE.....	70
Figura 44 - Astah UML.....	71

LISTA DE TABELAS

Tabela 1 - Lista de Casos de Uso - Pessoa Funcionário.....	18
Tabela 2 - Lista de Casos de Uso - Pessoa Cliente.....	20
Tabela - 3 Lista de Mensagens - Pessoa Funcionário.....	22
Tabela - 4 Lista de Mensagens - Pessoa Cliente.....	23
Tabela 5 - Caso de Uso Ator Funcionário - Realizar Login.....	27
Tabela 6 - CUI Ator Funcionário - Cadastro.....	28
Tabela 7 - CUI Ator Funcionário - Consulta Pessoa.....	30
Tabela 11 - Dicionário de atributos Classe Pessoa.....	37
Tabela 12 - Dicionário de atributos Classe Animal.....	37
Tabela 13 - Dicionário de atributos Classe Serviço.....	38
Tabela 14 - Dicionário de atributos ENUM Tipo Serviço.....	39
Tabela 16 - Dicionário de valores enumerados (ENUMs) TipoPessoa.....	40
3.5.1. Tabela Pessoa.....	49
3.5.2. Tabela Animal.....	50
3.5.3. Tabela Service.....	50

LISTA DE SIGLAS E ABREVIATURAS

AJAX – Asynchronous JavaScript and XML

BDO - Banco de Dados de Objeto

CSS - Cascading Style Sheets (Folha de Estilo em Cascata)

CUI – Caso de Uso Individual

DAO - Data Access Object (Objeto de Acesso a Dados)

DCU – Diagrama de Caso de Uso

DER – Diagrama de Entidades-Relacionamento

DOM – Document Object Model (Objeto Modelo do Documento)

EE – Enterprise Edition

FA – Fluxo Alternativo

FN – Fluxo Normal

GPS – Global Positioning System (Sistema de Posicionamento Global)

HTML - Hyper Text Markup Language (Linguagem de Marcação de Hipertexto)

HTTP – Hypertext Transfer Protocol (Protocolo de Transferência de Hipertexto).

IDE – Integrated Development Environment (Ambiente de Desenvolvimento Integrado)

JDK – Java Development Kit (Kit de Desenvolvimento Java)

JS - JavaScript

JSON – JavaScript Object Notation

JSP - Java Server Pages

MVC - Model View Controller (Modelo Visão Controle)

PHP - Hypertext Preprocessor (Processador de Hipertexto)

SDK – Software Development Kit (Kit de Desenvolvimento de Software)

SQL - Structured Query Language (Linguagem de Consulta Estruturada)

UML - Unified Modeling Language (Linguagem Unificada de Modelagem)

XAMPP – X(Sistema operacional) A(Apache) M(MariaDB) P(PHP) P(Pearl)

XML - Extensible Markup Language (Linguagem Extensível de Marcação)

SUMÁRIO

LISTA DE ILUSTRAÇÃO.....	5
LISTA DE SIGLAS E ABREVIATURAS.....	8
SUMÁRIO.....	9
1.INTRODUÇÃO.....	10
1.1. Levantamento de Informações.....	11
1.1.1 Descrição dos Objetivos Principais.....	12
1.1.2 Descrição do Sistema de Informação Atual.....	13
1.1.4 Ineditismo do Projeto.....	14
2.ANÁLISE ORIENTADA À OBJETOS.....	15
2.1 Lista de Atores.....	16
2.2. Lista de Casos de Uso.....	17
2.2.1 Lista de Mensagens.....	22
2.3. Diagrama de Contexto Geral.....	25
2.4. Diagramas de Caso de Uso Individual.....	26
2.4.1 DCU Individual Ator Funcionário - Realizar login.....	27
2.4.2 DCU Individual Ator Funcionário - Cadastro.....	28
2.4.3 DCU Individual Ator Funcionário - Consulta Pessoa.....	30
2.4.4 DCU Individual Ator Funcionário - Altera Pessoa.....	31
2.4.5 DCU Individual Ator Funcionário - Carregar Pessoa.....	33
2.4.6 DCU Individual Ator Funcionário - Inativar.....	34
3. PROJETO ORIENTADO À OBJETOS.....	35
3.1. Diagrama de Classes.....	36
3.2. Dicionário de Atributos das Classes.....	37
3.2.1. Classe Pessoa.....	38
3.2.2. Classe Animal.....	38
3.2.3 Classe Serviço.....	39
3.2.4 Dicionário de Valores Enumerados (ENUMs).....	40
3.2.5 ENUM TipoServico.....	40
3.2.6 ENUM TipoPessoa.....	41
3.3. Diagramas de Classes Camada Controller - DAO.....	41
3.3.2. Diagramas de Classes Camada DAO.....	44
3.4 Diagrama de Mapeamento Relacional.....	49
3.5 Projeto Físico do Banco de Dados.....	50
3.6 Especificação do Layout de Telas.....	52
3.6.1 - Tela Principal.....	52
3.6.2 - Tela de Cadastro de Cliente/Empregado.....	52
3.6.3 Tela De Cadastro de Serviços.....	54

3.6.4 Tela de Cadastro de Animais.....	54
3.6.5 Tela de Listagem de Animais.....	55
3.6.6 Tela De Listagem de Clientes.....	56
3.6.7 Tela de Listagem de Empregados.....	56
3.6.8 Tela de Listagem de Serviços.....	57
3.7 Diagrama de Classes - Interface Homem-Máquina.....	57
3.8 Diagrama de Sequencia - Modelo MVC.....	60
3.9 Diagrama de Implantação de Hardware e Software.....	64
3.10 Controle de Acesso e Segurança.....	65
4. METODOLOGIAS E TECNOLOGIAS USADAS.....	66
4.1 TECNOLOGIAS UTILIZADAS.....	66
5. CONSIDERAÇÕES FINAIS.....	70
6. REFERÊNCIAS BIBLIOGRÁFICAS.....	70

1.INTRODUÇÃO

A falta de um sistema integrado de gerenciamento em petshops pode gerar desperdício de recursos, perda de clientes, ineficiências no atendimento, e dificuldades na gestão de estoque e financeiras. O uso de um software web que possibilite o gerenciamento completo de um petshop é fundamental para garantir a eficiência e a transparência nas tarefas diárias. Com o uso de um software adequado, é possível monitorar o fluxo de entrada e saída de produtos (como brinquedos, rações, medicamentos e acessórios), agendar serviços (banho, tosa, consultas veterinárias) e gerenciar o histórico de atendimento dos pets, evitando assim o desperdícios e garantindo o uso otimizado dos recursos disponíveis.

Além disso, um sistema integrado de gerenciamento permite identificar rapidamente a necessidade de reposição de itens, promover uma gestão

eficiente, reduzir custos e melhorar o aproveitamento dos recursos disponíveis. Ferramentas analíticas e relatórios gerenciais fornecem insights valiosos que ajudam na tomada de decisões estratégicas, enquanto funcionalidades de lembretes e notificações melhoram o atendimento e a satisfação dos clientes.

Dessa forma, conclui-se que a implementação do Sistema Integrado de Gerenciamento de Petshop (S.I.G. Petshop) para o estabelecimento em questão se faz necessária, pois resultará em uma redução significativa de desperdícios, otimização do uso dos recursos e maior transparência na gestão, proporcionando um ambiente mais eficiente e sustentável para as atividades do petshop.

1.1. Levantamento de Informações

Através de uma análise detalhada da documentação fornecida pelo cliente, foi possível identificar os requisitos essenciais para o desenvolvimento do Sistema Integrado de Gerenciamento de Petshop (S.I.G. Petshop). Unindo as informações contidas nessa documentação com o conhecimento acadêmico disponibilizado pelo corpo docente e as orientações recebidas, a arquitetura do software foi cuidadosamente projetada, garantindo que o sistema atendesse inteiramente às necessidades específicas do estabelecimento.

Durante essa análise, ficou evidente a importância de fornecer um controle abrangente sobre o estoque, desde o cadastramento inicial dos produtos até as diversas movimentações de entrada e saída. Além disso, era essencial rastrear as solicitações de aquisição ou retirada de produtos realizadas pelos funcionários.

Com base nesse levantamento de informações, foi realizada uma análise detalhada e a modelagem conceitual do sistema utilizando a UML (Unified Modeling Language - Linguagem Unificada de Modelagem). Conforme amplamente reconhecido pela indústria de engenharia de software, a UML é

uma linguagem visual essencial para modelar sistemas complexos, como o S.I.G. Petshop, que são baseados no paradigma de orientação a objetos.

Assim, o sistema foi concebido não apenas para implementar as funcionalidades desejadas, mas também para garantir sua eficiência, usabilidade e conformidade com os padrões de qualidade estabelecidos pela indústria de desenvolvimento de software.

1.1.1 Descrição dos Objetivos Principais

O principal objetivo do Sistema Integrado de Gerenciamento de Petshop (S.I.G. Petshop) é fornecer uma plataforma web e mobile que assegure o controle eficaz do estoque, garantindo assim a disponibilidade adequada de produtos e materiais para atender às necessidades dos clientes e funcionários do estabelecimento.

Ao monitorar o estoque atual e estabelecer as quantidades mínimas desejadas para cada item, o sistema permite identificar os momentos oportunos para a reposição de estoque, evitando tanto a escassez de produtos essenciais quanto o excesso de compras de itens já disponíveis em quantidade suficiente.

Com registros minuciosos sobre os produtos, suas quantidades, custos e datas de aquisição, o S.I.G. Petshop viabiliza análises detalhadas sobre o consumo, identificação de tendências de demanda, projeção de custos futuros e embasamento de decisões gerenciais em dados concretos e atualizados em tempo real.

1.1.2 Descrição do Sistema de Informação Atual

Atualmente, o sistema de gerenciamento e cadastro funciona da seguinte maneira:

O cadastro de clientes e seus pets é realizado manualmente, utilizando formulários em papel ou planilhas eletrônicas descentralizadas. A busca e atualização de informações são demoradas e sujeitas a erros, dificultando o atendimento eficiente e personalizado.

O agendamento de serviços, como consultas veterinárias, banhos e tosas, é feito manualmente ou por meio de sistemas não integrados.

Os registros de serviços prestados são mantidos em documentos separados, sem uma visão consolidada do histórico do pet.

A comunicação entre os funcionários e clientes é ineficaz, resultando em falta de notificações e lembretes sobre compromissos agendados.

Controle de Estoque e Produtos:

O controle de estoque é realizado por meio de anotações em papel ou planilhas eletrônicas, sem integração com o sistema de cadastro de produtos.

A reposição de estoque é feita de maneira reativa, muitas vezes após a falta de produtos essenciais.

A divergência entre as quantidades registradas e as quantidades reais em estoque é comum, levando a perdas e desperdícios.

A gestão financeira é fragmentada, com registros de transações financeiras dispersos em diferentes sistemas e documentos.

A geração de relatórios financeiros e operacionais é demorada e sujeita a erros, dificultando a tomada de decisões informada.

1.1.3 Descrição dos Principais Problemas Existentes

Atualmente, o sistema de gerenciamento e cadastro no petshop apresenta várias falhas que comprometem a eficiência e a precisão na administração dos

recursos e informações. O processo é fragmentado e resulta em dificuldades de controle e rastreamento.

Essa abordagem atual apresenta várias deficiências, tais como falta de integração entre os processos de compra, dificuldades na gestão e controle de estoque, possibilidade de divergências entre as notas fiscais e os serviços prestados, além da ausência de um registro centralizado das transações financeiras e do histórico de compras.

Diante dessas falhas, se torna necessário implementar um Sistema Integrado de Gerenciamento de Petshop (S.I.G. Petshop), para substituir esse sistema atual falho. Este novo sistema proporcionará uma gestão mais eficiente, transparente e integrada dos serviços, otimizando o uso dos recursos disponíveis e garantindo um controle preciso sobre o funcionamento do petshop.

1.1.4 Ineditismo do Projeto

O projeto apresenta como ineditismo a classificação dos diferentes tipos de serviços oferecidos que possibilitam uma gestão mais organizada e específica. Métodos claramente definidos para cadastrar, pesquisar, alterar e localizar tanto pessoas quanto animais e serviços, centralização e integração de dados e a automação de Processos. Em suma, o novo sistema S.I.G. Petshop traz uma série de inovações que visam melhorar a eficiência, organização e gestão do petshop, substituindo o sistema antigo por uma solução mais moderna e integrada.

2. ANÁLISE ORIENTADA À OBJETOS

Análise Orientada a Objetos (OOA) se trata de metodologia fundamental no desenvolvimento de sistemas de software, proporcionando uma abordagem estruturada e intuitiva para compreender e modelar sistemas complexos. Este conceito baseia-se na ideia de decompor sistemas em "objetos", que são representações de entidades do mundo real ou conceitual, encapsulando dados e comportamentos relacionados. A análise orientada a objetos é um passo essencial na construção de sistemas eficientes, flexíveis e escaláveis.

Para o engenheiro de software americano, Grady Booch, um dos pioneiros da orientação a objetos, "A análise orientada a objetos é a atividade de examinar um problema, entender seus componentes e suas relações, e definir os objetos e as interações que resolvem esse problema." Essa definição salienta a importância de decompor o problema em partes menores e manejáveis, permitindo uma compreensão mais clara das necessidades e requisitos do sistema.

Craig Larman, cientista da computação e influente na área, enfatiza que "A análise orientada a objetos foca em descobrir e definir classes e objetos que formam a base para a solução do problema." Este processo envolve a identificação de classes, suas responsabilidades e as interações entre elas, facilitando a construção de um modelo de domínio que reflete a realidade do problema a ser resolvido.

A análise orientada a objetos também é bem exemplificada pela citação de Ivar Jacobson, que afirma: "A análise orientada a objetos ajuda a transformar os requisitos em um modelo estruturado, definindo claramente as responsabilidades dos objetos." Isso ressalta como a OOA contribui para a definição clara das funcionalidades e interações no sistema, garantindo que cada objeto tenha um papel específico e bem definido.

De forma resumida, a análise orientada a objetos é uma metodologia poderosa e essencial no desenvolvimento de sistemas de software modernos.

Ao focar na decomposição de problemas em objetos e suas interações, a OOA proporciona uma base sólida para a construção de sistemas robustos e escaláveis, promovendo a modularidade, a reutilização e a manutenção do código. Este paradigma não apenas transforma a maneira como os sistemas são concebidos, mas também melhora significativamente a qualidade e a eficiência do desenvolvimento de software.

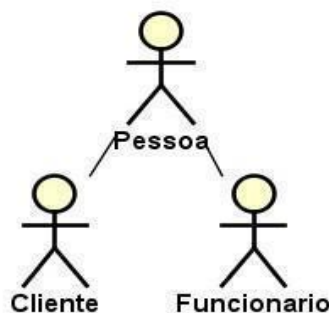
2.1 Lista de Atores

De acordo com Guedes (2011), a lista de atores é um dos componentes essenciais do diagrama de casos de uso, complementando os próprios casos de uso. Os atores representam os papéis desempenhados pelos diversos usuários que interagem com os serviços e funções do sistema. Em termos simples, um ator é qualquer entidade externa que se comunica com o software.

Atores são fundamentais para a definição de casos de uso, pois ajudam a identificar quem vai interagir com o sistema e de que forma essas interações ocorrerão. Segundo Góes (2014), "um ator é um usuário que interage com o sistema; é um objeto ou um conjunto de objetos que se comunica com o sistema e assim são chamados porque cada um desempenha um papel dentro do sistema". Esta definição enfatiza que um ator pode ser tanto uma pessoa quanto outro sistema ou dispositivo que se conecta ao sistema em questão.

A figura 1 abaixo demonstra todos os atores que interagem no sistema.

Figura 1 - Diagrama de Atores – S.I.G. PETSHOP



Fonte:O Autor

- **Ator Pessoa:** interage com as funcionalidades do sistema e realiza todas funções permitidas aos Atores Cliente e Usuário Funcionário que herdam todas as funcionalidades existentes no Ator Usuário. Segundo o Sbrocco (2014) o conceito de herança representa um mecanismo específico do paradigma orientado a objetos que permite criar classes a partir de outras já existentes, aproveitando as características da classe que será estendida.
- **Ator Cliente:** interage com todo o sistema através das seguintes funcionalidades: Realiza Login, Consulta produto, altera produto, Consulta Estoque.
- **Ator Funcionário:** interage com o sistema através das seguintes funcionalidades: altera estoque, Consulta Fornecedor, Consulta Funcionário, Altera Funcionário, Cadastro de Materiais, Realiza Acerto de Estoque, Geração Relatórios, Aprovação de Solicitações.

2.2. Lista de Casos de Uso

Os casos de uso representam uma técnica essencial para identificar os requisitos funcionais de um sistema. Eles são utilizados para descrever as

interações típicas entre os usuários e o sistema, oferecendo uma narrativa sobre como o sistema é empregado (FOWLER, 2005).

Segundo Guedes (2011), os casos de uso podem ser divididos em primários, que abordam processos essenciais focados nos requisitos funcionais do software, e secundários, que tratam de processos periféricos do sistema. A documentação detalhada dos casos de uso serve como base para outros diagramas, como o diagrama de sequência.

O principal objetivo dessa documentação é proporcionar ao cliente um relatório que explique o comportamento esperado de cada caso de uso e suas funcionalidades.

Nas tabelas de número 1 a 3 abaixo, serão apresentadas listas individuais de casos de uso agrupados por funcionalidades dos perfis de Usuário Pessoa, Usuário Cliente e Usuário Funcionário em relação ao sistema.

A tabela 1 exibe a lista de casos de usos entre o Ator Pessoa Funcionário e o Sistema.

Tabela 1 - Lista de Casos de Uso - Pessoa Funcionário

Pessoa Funcionário				
Nº	Descrição	Caso de Uso	Entrada	Saída
01	Usuário realiza login no Software	Autenticação	Credenciais de login	Msg: 01 Acesso ao sistema
02	Consultar/ Localizar/ Buscar	Gestão de Animais	Parâmetros de busca (nome do animal, espécie, raça)	Msg: 02
03	Alterar Animais	Alterar Animais	Dados atualizados dos animais	Msg: 09
04	Consultar/Localizar/Buscar Serviços	Gestão de Serviços	Parâmetros de busca (tipo de serviço, data)	Msg:03
05	Alterar Serviços	Alterar Serviços	Dados atualizados dos serviços	Msg:09
06	Consultar/Localizar/Buscar Funcionário	Gestão de Funcionários	Parâmetros de busca (nome,	Msg: 04 Informações detalhadas

			cargo, contato)	do funcionário
07	Cadastro de Animais	Gestão de Animais	Informações do animal (nome, espécie, raça, idade, dono)	Msg: 06 Confirmação do cadastro do animal
08	Cadastro de Serviços	Gestão de Serviços	Informações do serviço (tipo, valor, data)	Msg: 07 Confirmação do cadastro do serviço
09	Aprovação de Solicitações	Gestão de Solicitações	Solicitação (pedido de compra, folga)	Msg: 08

Fonte: O Autor

A tabela 2 exibe a lista de casos de usos entre o Ator Pessoa Cliente e o Sistema.

Tabela 2 - Lista de Casos de Uso - Pessoa Cliente

Pessoa Cliente				
Nº	Descrição	Caso de Uso	Entrada	Saída
01	Usuário realiza login no Software	Autenticação	Credenciais de login	Msg: 01 Acesso ao sistema
02	Consultar Animais	Gestão de Animais	Parâmetros de busca (nome do animal, espécie, raça))	Msg 02:Lista de animais correspondentes aos parâmetros de busca
03	Agendar Serviço	Agendamento de Serviços	Informações do agendamento (tipo de serviço, data, hora)	Msg: 03 Confirmação do agendamento
04	Consultar Serviços	Gestão de Serviços	Parâmetros de busca (tipo de serviço, data)	Msg: 04 Lista de serviços correspondentes aos parâmetros de busca

05	Status Compra	Gestão de Compras	Parâmetros de busca (compra, pedido)	Msg: 05 Informações detalhadas da compra
06	Alterar Dados Pessoais	Alterar Dados Pessoais	Dados atualizados do cliente (nome, endereço, contato)	Msg: 06 Mensagem de confirmação da alteração

Fonte: O Autor

2.2.1 Lista de Mensagens

A tabela 3 demonstra as mensagens que o sistema retornará em resposta às entradas requisitadas pelos usuários nas tabelas de Caso de Uso acima demonstradas.

Tabela - 3 Lista de Mensagens - Pessoa Funcionário.

Nº Mensagem	Mensagem
Msg 01	"Login realizado com sucesso. Bem-vindo(a)" / "Login ou senha inválidos!"
Msg 02	"Consulta realizada com sucesso" / "Nenhum pet encontrado"
Msg 03	"Serviços atualizados disponíveis para consulta. " / "Não há informações disponíveis sobre os serviços no momento."
Msg 04	"Aqui estão as informações detalhadas sobre seus funcionários." / "Nenhum funcionário foi encontrado com base nos critérios de pesquisa."
Msg 05	"As informações do funcionário foram atualizadas com sucesso." / "Erro ao tentar atualizar as informações do funcionário. Tente novamente."
Msg 06	"Novo Pet cadastrado com sucesso no sistema." / "Erro ao tentar cadastrar o novo Pet. Verifique os dados e tente novamente."
Msg 07	"Novo Serviço cadastrado com sucesso no sistema." / "Erro ao tentar cadastrar o novo Serviço. Verifique os dados e tente novamente."
Msg 08	"A solicitação foi aprovada com sucesso. Os procedimentos serão iniciados." / "A solicitação foi reprovada. Entre em contato para mais informações."
Msg 09	"Dados alterados com sucesso" / "Os dados não pode ser alterados"

Fonte:O Autor

Tabela - 4 Lista de Mensagens - Pessoa Cliente.

Nº Mensagem	Mensagem
Msg 01	“Login realizado com sucesso. Bem-vindo(a)” / “Login ou senha inválidos!”
Msg 02	"Aqui estão as informações detalhadas sobre os Pets." / "Erro ao processar as informações. ”
Msg 03	"Seu agendamento foi confirmado. Acompanhe o status." / "Houve um problema ao processar sua solicitação. Tente novamente mais tarde."
Msg 04	"Aqui estão os serviços solicitados." / "Não há disponibilidade para os serviços solicitados no momento."
Msg 05	"Você recebeu uma notificação sobre o status da sua solicitação. Confira agora." / "Nenhuma notificação nova no momento."
Msg 06	“Dados alterados com sucesso!” / “Erro ao registrar novos dados.”

Fonte:O Autor

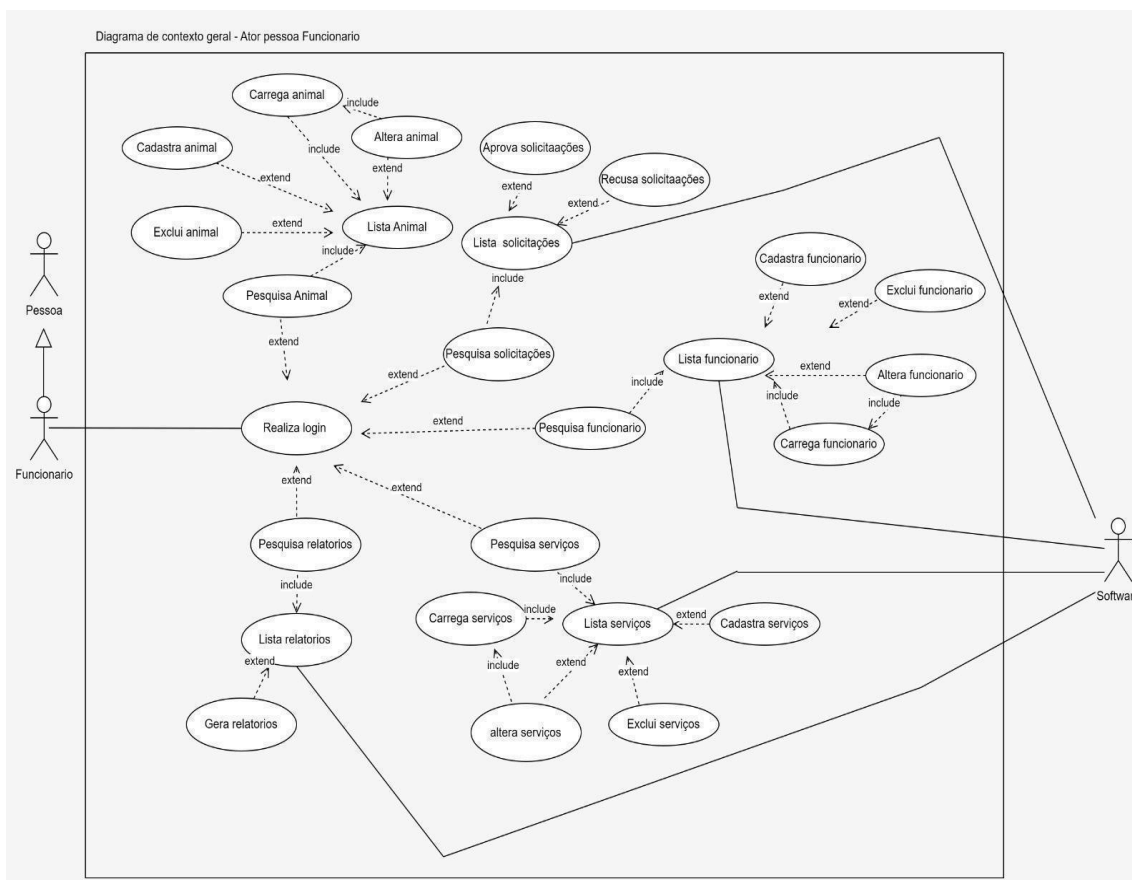
2.3. Diagrama de Contexto Geral

O Diagrama de contexto geral se trata de um diagrama de casos de uso que representa de forma geral todos os atores e casos de uso apresentando o sistema como um todo (SBROCCO, 2014).

A seguir os diagramas serão apresentados em módulos representando a interação de cada ator com o sistema.

A figura abaixo representa o Diagrama de Contexto Geral no módulo Pessoa Funcionário.

Figura 2 - Diagrama de Contexto Geral - Módulo Funcionário



Fonte: O Autor

Figura 3 - Diagrama de Contexto Geral - Módulo Cliente

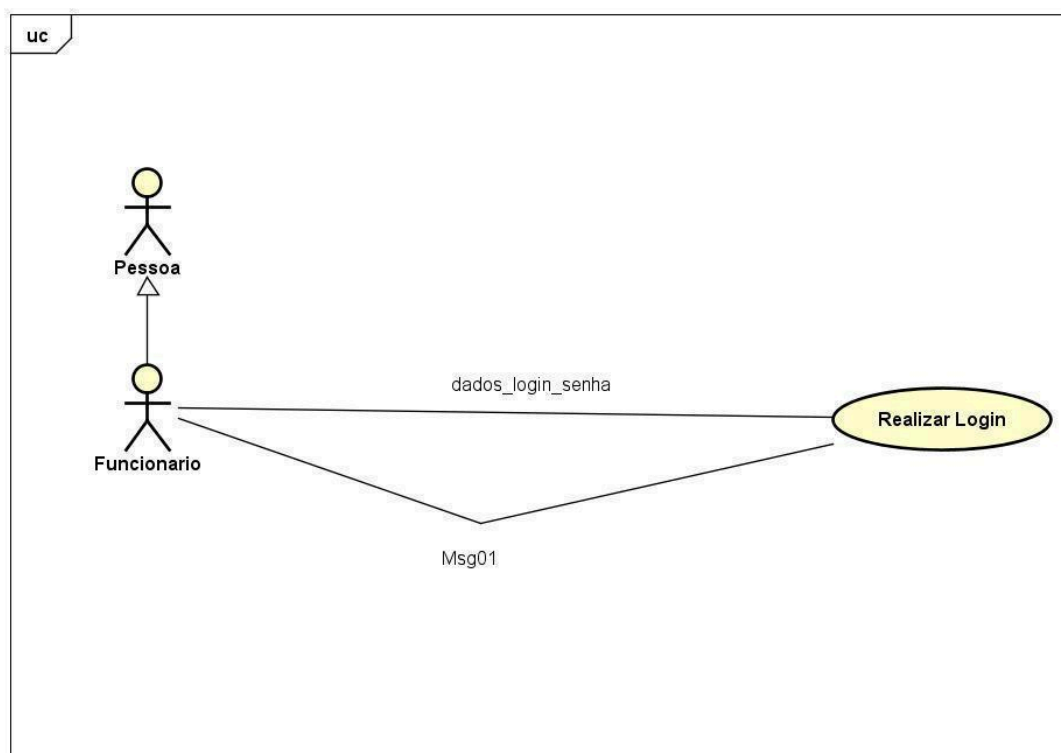


O Diagrama de caso de uso individual mostra de forma individualizada as interações entre um dos atores e o sistema, representa cada funcionalidade do sistema permitindo uma visualização destacada de como cada ator irá desempenhar suas funções dentro do sistema (SBROCCO, 2014).

2.4.1 DCU Individual Ator Funcionário - Realizar login

A figura 4 abaixo representa o Caso de Uso Individual Realizar Login no módulo Ator Funcionário.

Figura 4 - DCU Individual Ator Funcionário - Realizar login



Fonte:O Autor

A tabela 5 descreve todas as informações referentes ao Caso de Uso Individual do módulo Pessoa Funcionário – Realizar Login, exibe seus pré-requisitos e o fluxo das informações.

Tabela 5 - Caso de Uso Ator Funcionário - Realizar Login

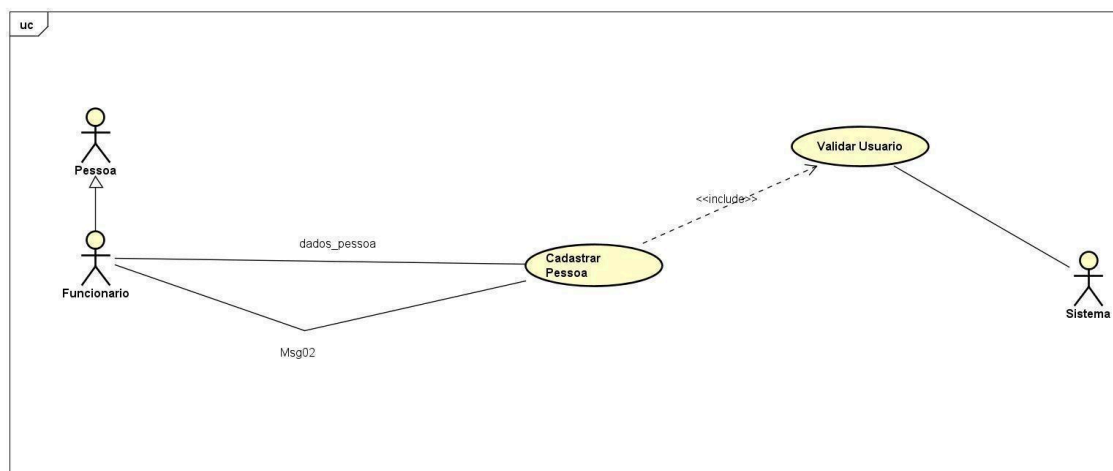
Ator Principal: Funcionário
Descrição: Funcionário informa login e senha. O sistema valida e permite o acesso
Pré-requisito: Funcionário não estar logado.
Fluxo Normal: 1 - Funcionário Loga no sistema. 2 - Insere as informações de login na página de login. 3 - Sistema valida as informações de login. 4 - Sistema exibe página principal. 5 - Fim
Fluxo Alternativo: 3.1 -Informações de login erradas o sistema retorna uma mensagem. 3.1.1 - Retorna para o item 2.
Dados: Usuário e senha.

Fonte:O Autor

2.4.2 DCU Individual Ator Funcionário - Cadastro

A figura 5 abaixo representa o Caso de Uso Individual Cadastro no modulo Cadastro

Figura 5 - DCU Individual Ator Funcionário - Cadastro



Fonte:O Autor

A tabela 6 descreve todas as informações referentes ao Caso de Uso Individual Cadastrar Funcionário do módulo Pessoa Funcionário, exibe seus pré-requisitos e o fluxo das informações.

Tabela 6 - CUI Ator Funcionário - Cadastro

Ator Principal: Funcionário
Descrição: Funcionário informa os dados a serem cadastrados e o sistema retorna o resultado..
Pré-requisito: Novos dados a serem cadastrados no sistema.
Fluxo Normal: 1 - Funcionário Informa os dados 2 - Sistema válida. 3 - Sistema retorna o resultado. 4 - Fim

Fluxo Alternativo:

2.1 - Parâmetros insuficientes ou errados

2.1.1 - Sistema retorna mensagem

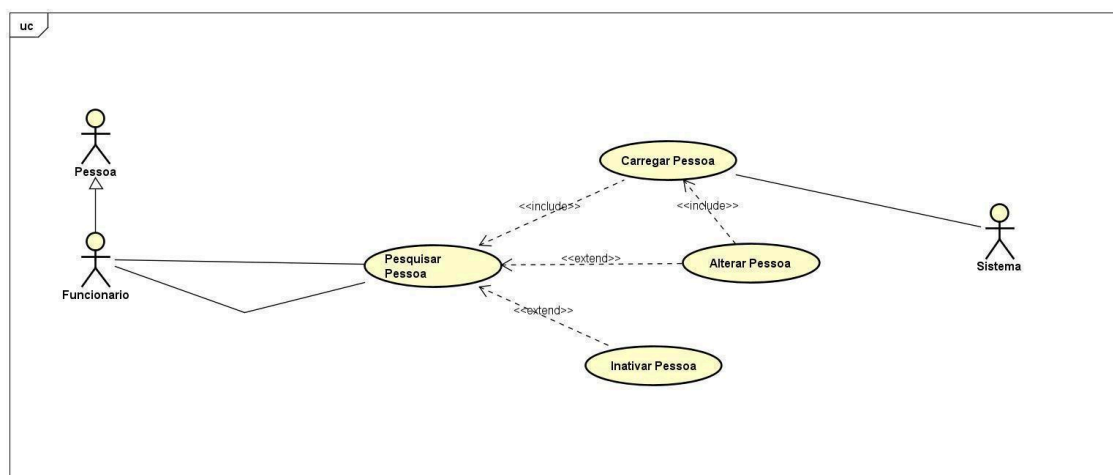
Dados: Nome, cpf, id, endereço..

Fonte:O Autor

2.4.3 DCU Individual Ator Funcionário - Consulta Pessoa

A figura 6 abaixo representa o Caso de Uso Individual Consultar Pessoa no módulo Pessoa.

Figura 6 - DCU Individual Ator Funcionário - Consulta Pessoa



Fonte:O Autor

A tabela 7 descreve todas as informações referentes ao Caso de Uso Individual Consultar Funcionário no módulo Pessoa Funcionário, exibe seus pré-requisitos e o fluxo das informações.

Tabela 7 - CUI Ator Funcionário - Consulta Pessoa

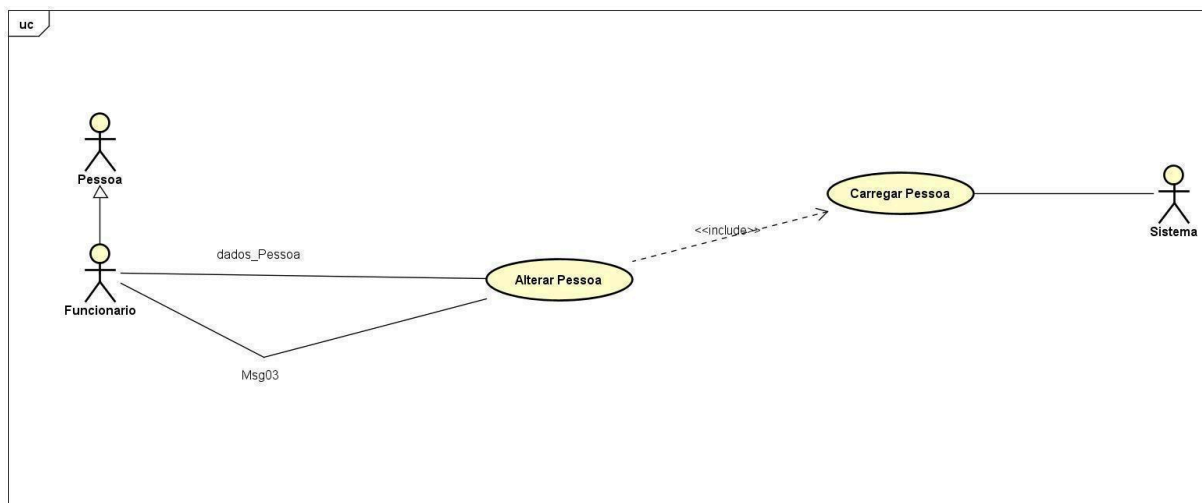
Ator Principal: Funcionário
Descrição: Funcionário faz a busca por alguma pessoa
Pré-requisito: Ter algum cliente ou funcionário cadastrado.
Fluxo normal: 1 - Após login funcionário clica no botão “Consultar Pessoa” 2 – Sistema direciona funcionário para tela de dados das Pessoas. 3 – Fim.
Fluxo Alternativo: 2.1 - Caso os dados do sistema estiverem vazios retorna uma mensagem. 2.2 - Caso os dados não coincidirem o sistema retorna uma mensagem.
Dados: nome, cpf, id.

Fonte:O Autor

2.4.4 DCU Individual Ator Funcionário - Altera Pessoa

A 7 figura abaixo representa o Caso de Uso Individual Altera Pessoa no modulo Pessoa.

Figura 7 - DCU Individual Ator Funcionário - Altera Pessoa



Fonte:O Autor

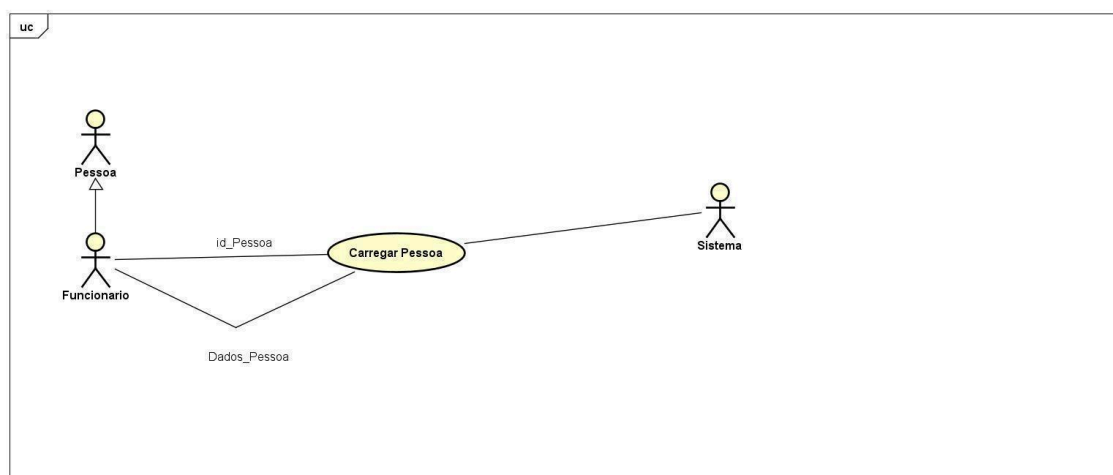
Ator Principal: Funcionário
Descrição: Funcionário alterando dados da pessoa..
Pré-requisito: Ter uma pessoa cadastrada.
Fluxo Normal: 1 - Funcionário informa dados da pessoa 2 - Sistema carrega e valida os dados 3 - Fim.
Fluxo Alternativo: 2.1 - Sistema retorna mensagem caso não tiver fornecedor.
Dados: Nome, cpf, email, senha

Fonte:O Autor

2.4.5 DCU Individual Ator Funcionário - Carregar Pessoa

A figura 8 abaixo representa o Caso de Uso Individual Carregar Pessoa no modulo Pessoa.

Figura 8 - DCU Individual Ator Funcionário - Carregar Pessoa



Fonte:O Autor

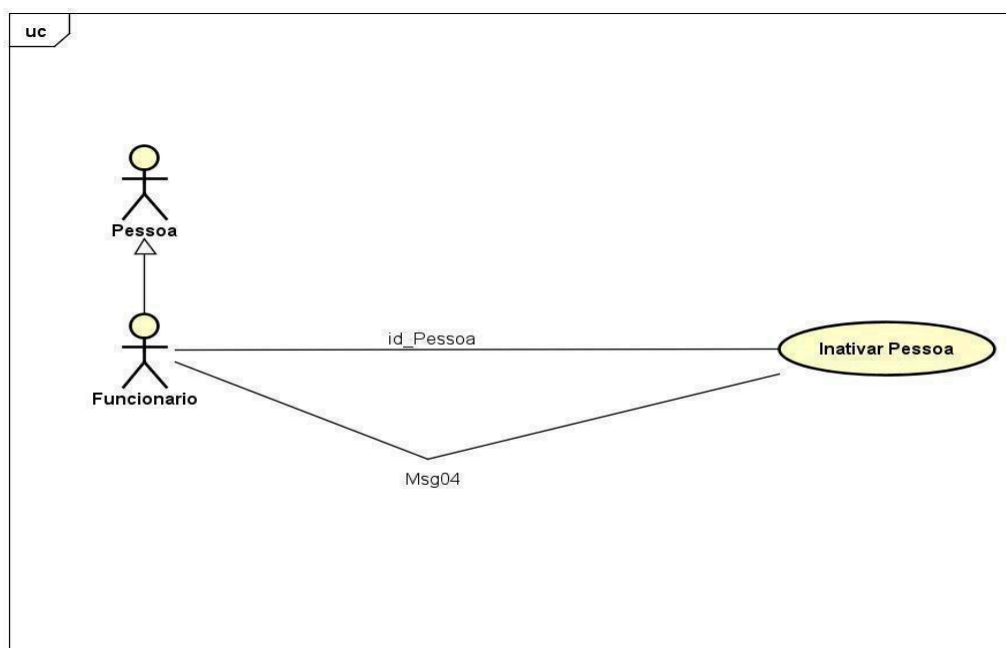
Ator Principal: Funcionário
Descrição: Funcionário carregando dados da pessoa.
Pré-requisito: Ter alguma pessoa cadastrada.
Fluxo Normal: 1 - Funcionário informa dados do pessoa 2 - Sistema retorna a pessoa atualizada.. 3 - Fim
Fluxo Alternativo: 2.1 Parâmetros errados ou sistema sem pessoa retorna mensagem.
Dados: Nome, cpf

Fonte:O Autor

2.4.6 DCU Individual Ator Funcionário - Inativar

A 9 figura abaixo representa o Caso de Uso Individual Inativar no módulo Pessoa.

Figura 9 - DCU Individual Ator Funcionário - Inativar



Fonte:O Autor

Ator Principal: Funcionário.

Descrição: Inativar pessoa.

Pré-requisito: Ter alguma pessoa cadastrada.

Fluxo Normal:

- 1 - Funcionário informa o id da pessoa a ser inativada
- 2 - Sistema retorna.
- 3 - Fim

Fluxo Alternativo:

2.1 - Caso não tenha a pessoa com os dados informados sistema retorna mensagem..

Dados: id

Fonte: O Autor

3. PROJETO ORIENTADO À OBJETOS

Um objeto é uma entidade que possui atributos (características), comportamentos, responsabilidades e se relaciona com ele mesmo ou com outros objetos por meio de mensagens. Podem ser concretos ou conceituais (GÓES, 2014).

Guedes (2014) explica que através de classificação e abstração conseguimos definir classes, ou seja, grupo de objetos, sendo que cada objeto é um exemplo de um determinado grupo, possuindo as mesmas características e os mesmos comportamentos de qualquer objeto do grupo em questão.

De acordo com Sbrocco (2014) existem diversas vantagens na utilização do paradigma orientado a objetos, tais como proporcionar maior facilidade de manutenção dos sistemas, maior índice de reaproveitamento do código, e menor tempo para desenvolvimento dos programas em virtude dos códigos serem menores pela utilização do mecanismo de herança existente na orientação a objetos.

No paradigma orientado a objetos é preciso escolher o que é importante e o que pode ser omitido, seguindo-se um ponto de vista, decompondo o sistema através da visão funcional (quem vai utilizar, interagir com o sistema), visão dos objetos (foco e busca da estrutura dos problemas e não apenas dos dados) e visão dos dados (os dados podem ser utilizados para mais de uma função) (SBROCCO, 2014).

Esse capítulo irá tratar da fase de projeto orientado a objetos demonstrando através de diagramas a formação das classes, seus atributos e métodos, e a interação e as atividades que serão executadas dentro do sistema.

3.1. Diagrama de Classes

Classe é a descrição de um conjunto de objetos que compartilham os mesmos atributos, operações e relacionamentos, podem ser utilizadas para representar elementos de software, hardware e qualquer coisa que possa ser conceituada (SBROCCO, 2014).

O diagrama de classe deve ser detalhado para poder descrever com precisão o sistema a ser construído. Os sistemas orientados a objetos organizam-se ao redor de classes de objetos, que representam tanto os elementos do domínio do problema, incorporados ao modelo, como os elementos propostos para a implementação da solução (SBROCCO, 2014).

De acordo com Guedes (2014) é um dos mais importantes e mais utilizado da UML, e seu principal enfoque é permitir a visualização das classes que irão compor o sistema com seus respectivos atributos e métodos.

O diagrama de classes pode ser utilizado na fase de análise para produzir o modelo conceitual por meio do qual representamos as classes e seus atributos, sem levar em conta a implementação, com ênfase no domínio do problema, na fase de projeto é utilizada para enfoque na solução do problema, evidenciando a implementação, como os tipos de dados dos atributos, e suas operações e métodos (GÓES, 2014).

A rigor em todas as classes deve haver um método `get`¹⁷ e um `set`¹⁸ para cada atributo da classe, porém não é muito prático fazer essa representação no diagrama de classes, por deixa-lo muito extenso quando alguma classe possuir muitos atributos (GUEDES, 2014). Com base nesta argumentação será representada no diagrama de classes apenas a classe Pessoa com os métodos `get` e `set` de seus atributos. A figura 16 demonstra o diagrama de

classes a ser utilizado durante a fase de projeto com enfoque na solução do problema identificado durante a fase de análise.

3.2. Dicionário de Atributos das Classes

Dicionário de Atributos das Classes Os atributos representam características de uma classe que costumam variar de objeto para objeto, e que permitem diferenciar um objeto de outro da mesma classe (GUEDES, 2014).

Somente os atributos que são de interesse do sistema devem ser descritos em uma classe, e cada um deles ainda pode ser de um tipo diferente. O valor correspondente a todos os atributos definidos é o que chamamos de estado do objeto (SBROCCO, 2014).

De acordo com Guedes (2014) os atributos são apresentados na segunda divisão da classe e contêm, normalmente, duas informações: o nome que o identifica e eventualmente o tipo de dado que o atributo armazena, como, por exemplo, integer, float ou character.

Góes (2014) nos explica que os atributos também possuem sinais indicadores que indicam sua visibilidade, a qual define por quem uma propriedade desse atributo pode ser utilizada. Os sinais indicadores são “+ (público)”, “- (privado)”, “# (protegido)” e “~(pacote)”.

O dicionário de atributos das classes irá representar essa estrutura dos atributos, e além de conter, o nome do atributo irá descrever de forma sucinta a descrição do atributo.

As tabelas abaixo irão apresentar os atributos das classes identificadas no desenvolvimento do sistema.

A tabela 11 representa o dicionário dos atributos da Classe Pessoa.

3.2.1. Classe Pessoa

Tabela 11 - Dicionário de atributos Classe Pessoa.

CLASSE: Pessoa	
ATRIBUTO	DESCRIÇÃO
Id	Código da pessoa (chave primária)
nome	Nome da Pessoa
cpf	Dados regionais
email	Email da pessoa
createdAt	Quando foi Criado
tipoPessoa	Tipo Pessoa

Fonte: O Autor

3.2.2. Classe Animal

A tabela 12 representa o dicionário dos atributos da Classe Animal.

Tabela 12 - Dicionário de atributos Classe Animal.

CLASSE: Animal	
ATRIBUTO	DESCRIÇÃO
Id	Código do animal (chave primária)

nome	Nome do animal
tipo	Tipo do animal
raca	Raça do animal

Fonte:O Autor

3.2.3 Classe Serviço

A tabela 13 representa o dicionário dos atributos da Classe Serviço.

Tabela 13 - Dicionário de atributos Classe Serviço.

CLASSE: Serviço	
ATRIBUTO	DESCRIÇÃO
idServico	Código do serviço (chave primária)
valor_total	Valor total do serviço
descricao	Descrição do serviço
idEmpregado	Identificação do empregado
idCliente	Identificação do Cliente
tipoServico	Tipo de Serviço

Fonte:O Autor

3.2.4 Dicionário de Valores Enumerados (ENUMs)

ENUMs (abreviação de "enumerations" ou "enumerações") são tipos de dados que consistem em um conjunto de constantes nomeadas, representando valores possíveis para uma variável. Em muitos casos, os ENUMs são utilizados para definir um conjunto fixo de valores possíveis, tornando o código menos propenso a erros e mais legível. Eles são amplamente usados em programação para representar estados, categorias ou qualquer conjunto de valores constantes.

Por exemplo, ao invés de utilizar valores literais para representar cargos de funcionários em um sistema, utilizamos um ENUM para definir estes valores de forma clara e centralizada. Isso facilita a manutenção e a compreensão do código, além de reduzir a probabilidade de erros como digitação incorreta de strings.

O dicionário de valores enumerados (ENUMs) irá representar essa estrutura das constantes, e além de conter, o nome da constante irá descrever de forma sucinta a descrição da constante.

3.2.5 ENUM TipoServico

A tabela 14 representa o dicionário dos atributos da ENUMTipoServico.

Tabela 14 - Dicionário de atributos ENUM Tipo Serviço.

ENUM: TipoServico	
ATRIBUTO	DESCRIÇÃO
banho	Descrição do serviço de banho
atendimento	Descrição do atendimento veterinário
tosa	Descrição do serviço tosa

Fonte: O Autor

3.2.6 ENUM TipoPessoa

A tabela 16 representa o dicionário de valores enumerados do ENUM TipoPessoa

Tabela 16 - Dicionário de valores enumerados (ENUMs) TipoPessoa

ENUM: TipoPessoa	
CONSTANTE	DESCRIÇÃO
empregado	Empregado
client	Cliente

Fonte: O Autor

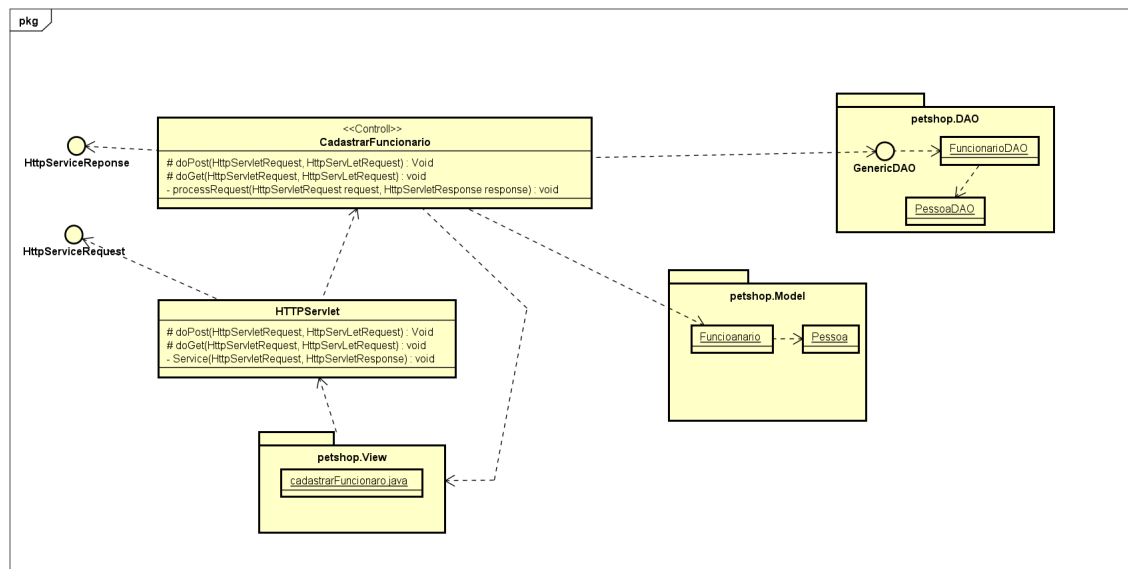
3.3. Diagramas de Classes Camada Controller - DAO

O Controlador (Controller) é responsável por receber as requisições do usuário montar os objetos correspondentes e passá-los ao Model para que faça o que tem que ser feito. Depois de receber o retorno da operação executada, o controlador apresenta o resultado para o usuário, geralmente o redirecionando para a View (CORDEIRO, 2012).

Os diagramas das figuras abaixo representam as classes na camada Controller.

Diagrama de Classe Controller Cadastrar.

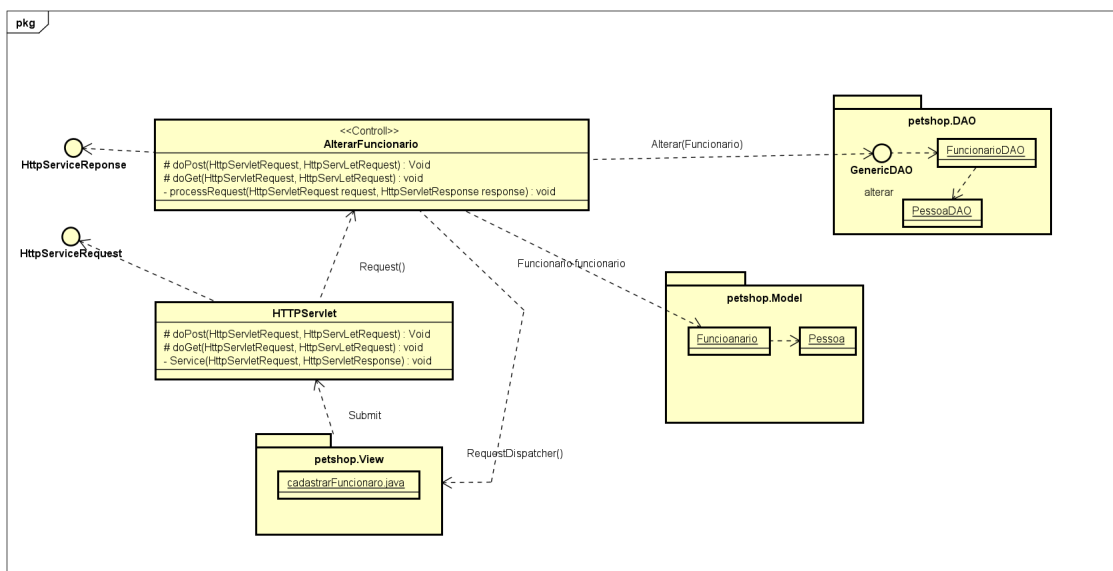
Figura 10 - Controller CadastrarFuncionario



Fonte: O Autor

Diagrama de Classe Controller Alterar.

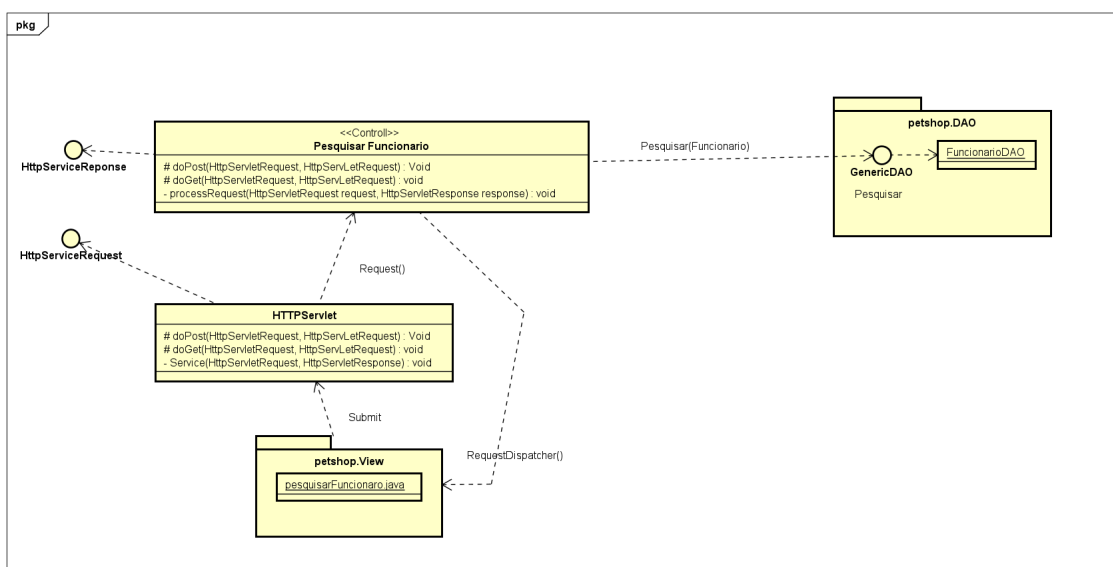
Figura 11 - Controller AlterarFuncionario



Fonte: Os Autores

Diagrama de Classe Controller Pesquisar.

Figura 12 - Controller PesquisarFuncionario



Fonte: O Autor

3.3.2. Diagramas de Classes Camada DAO

De acordo com Gonçalves (2007) o padrão DAO (Data Access Object) é o padrão mais utilizado para acesso a dados contidos em um banco de dados. Sempre que você precisa acessar um banco de dados que está mantendo seu modelo de objetos, é melhor empregar o padrão DAO.

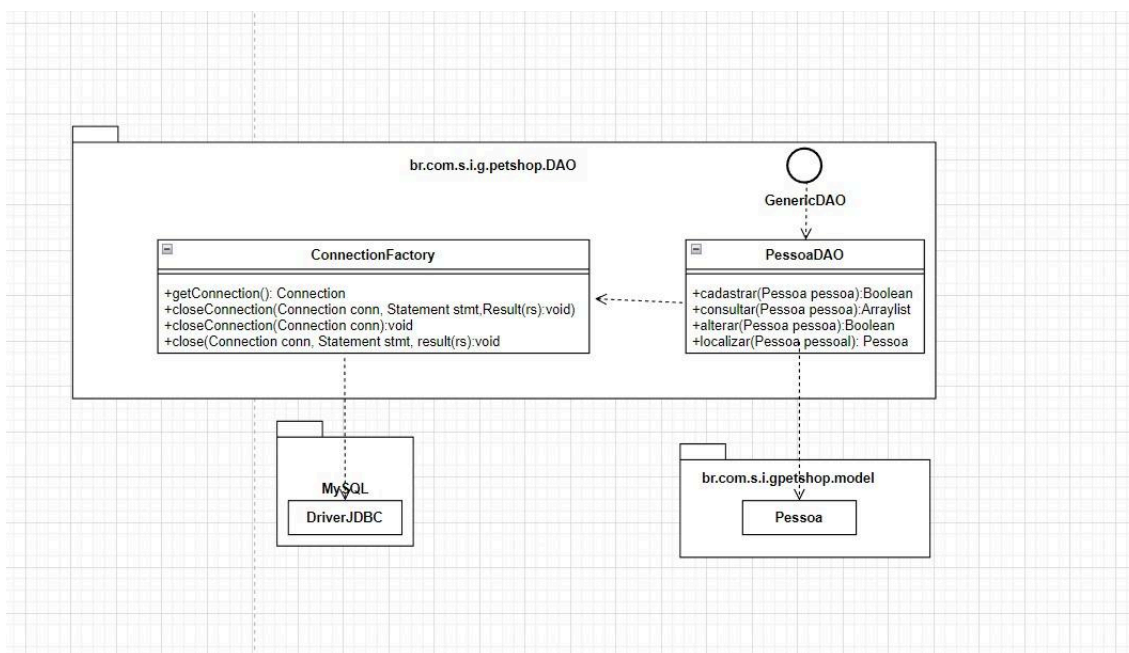
Tipicamente um DAO inclui métodos para inserir, selecionar, atualizar e excluir objetos de um banco de dados.

Dependendo de como você implementa o padrão DAO, você poderá ter um DAO para cada classe de objetos em sua aplicação ou poderá ter um único DAO que é responsável por todos os seus objetos (GONÇALVES, 2007).

Os Diagramas das figuras abaixo representam as classes na camada DAO.

Diagrama de Classe Camada DAO Classe Pessoa.

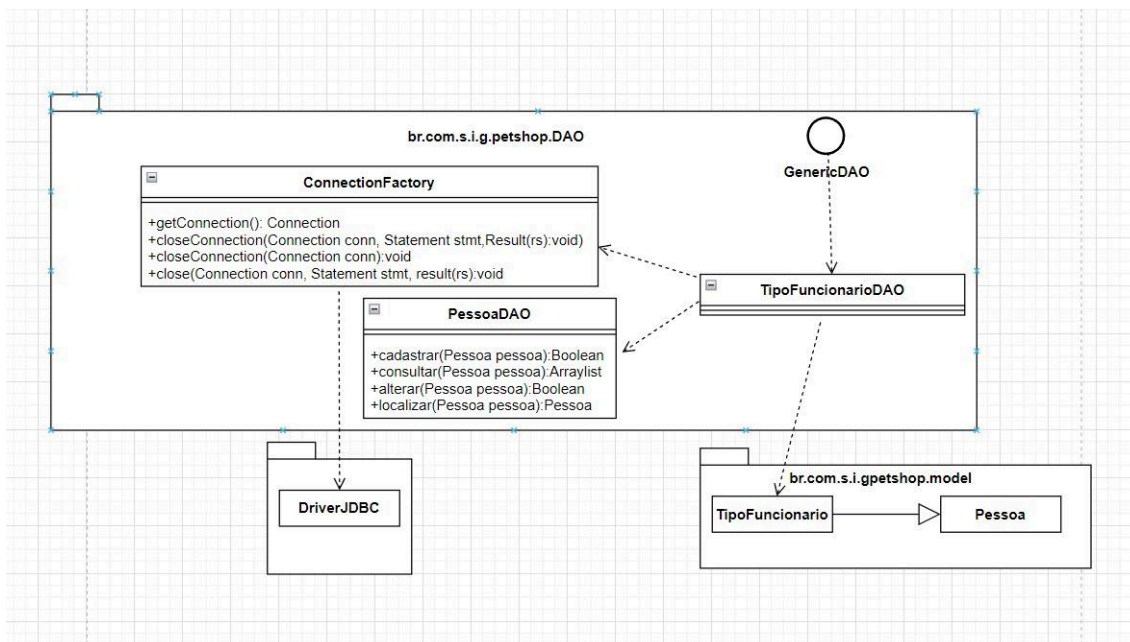
Figura 13 - DAO Classe Pessoa



Fonte: O Autor

Diagrama de Classe Camada DAO Classe Tipo Funcionario.

Figura 14 - DAO Classe TipoFuncionario



Fonte: O Autor

Diagrama de Classe Camada DAO Classe Tipo Cliente.

```
classDiagram
    class br.com.s.i.g.petshop.DAO {
        class ConnectionFactory {
            +getConnection(): Connection
            +closeConnection(Connection conn, Statement stmt, Result(rs): void
            +closeConnection(Connection conn): void
            +close(Connection conn, Statement stmt, result(rs): void
        }
        class PessoaDAO {
            +cadastrar(Pessoa pessoa): Boolean
            +consultar(Pessoa pessoa): Arraylist
            +alterar(Pessoa pessoa): Boolean
            +localizar(Pessoa pessoa): Pessoa
        }
    }
    class br.com.s.i.g.petshop.model {
        class TipoCliente
        class Pessoa
    }
    class br.com.s.i.g.petshop.DAO {
        class TipoClienteDAO {
        }
    }
    class GenericDAO
    ConnectionFactory ..> TipoClienteDAO
    PessoaDAO ..> TipoClienteDAO
    GenericDAO ..> TipoClienteDAO
    TipoClienteDAO ..> DriverJDBC
    TipoClienteDAO ..> TipoCliente
    TipoCliente ..|> Pessoa
```

The diagram illustrates the DAO layer structure. The **br.com.s.i.g.petshop.DAO** package contains **ConnectionFactory** and **PessoaDAO**. **ConnectionFactory** has methods: `+getConnection(): Connection`, `+closeConnection(Connection conn, Statement stmt, Result(rs): void`, `+closeConnection(Connection conn): void`, and `+close(Connection conn, Statement stmt, result(rs): void`. **PessoaDAO** has methods: `+cadastrar(Pessoa pessoa): Boolean`, `+consultar(Pessoa pessoa): Arraylist`, `+alterar(Pessoa pessoa): Boolean`, and `+localizar(Pessoa pessoa): Pessoa`. The **br.com.s.i.g.petshop.model** package contains **TipoCliente** and **Pessoa**, with **TipoCliente** inheriting from **Pessoa**. The **br.com.s.i.g.petshop.DAO** package also contains **TipoClienteDAO**, which inherits from **GenericDAO**. Dashed arrows indicate dependencies: **ConnectionFactory** depends on **TipoClienteDAO**, **PessoaDAO** depends on **TipoClienteDAO**, **TipoClienteDAO** depends on **ConnectionFactory**, **TipoClienteDAO** depends on **DriverJDBC**, and **TipoClienteDAO** depends on **TipoCliente**.

Diagrama de Classe Camada DAO Classe Animal.

The diagram shows a package named **br.com.s.i.g.petshop.DAO** containing two interfaces: **ConnectionFactory** and **AnimalDAO**. **ConnectionFactory** defines methods for managing database connections. **AnimalDAO** defines methods for animal management. A package **MySQL** contains the **DriverJDBC** class, which implements **ConnectionFactory**. Another package **br.com.s.i.g.petshop.model** contains the **Animal** class, which implements **AnimalDAO**. A dashed dependency arrow points from **AnimalDAO** to **ConnectionFactory**, indicating that **AnimalDAO** depends on **ConnectionFactory**.

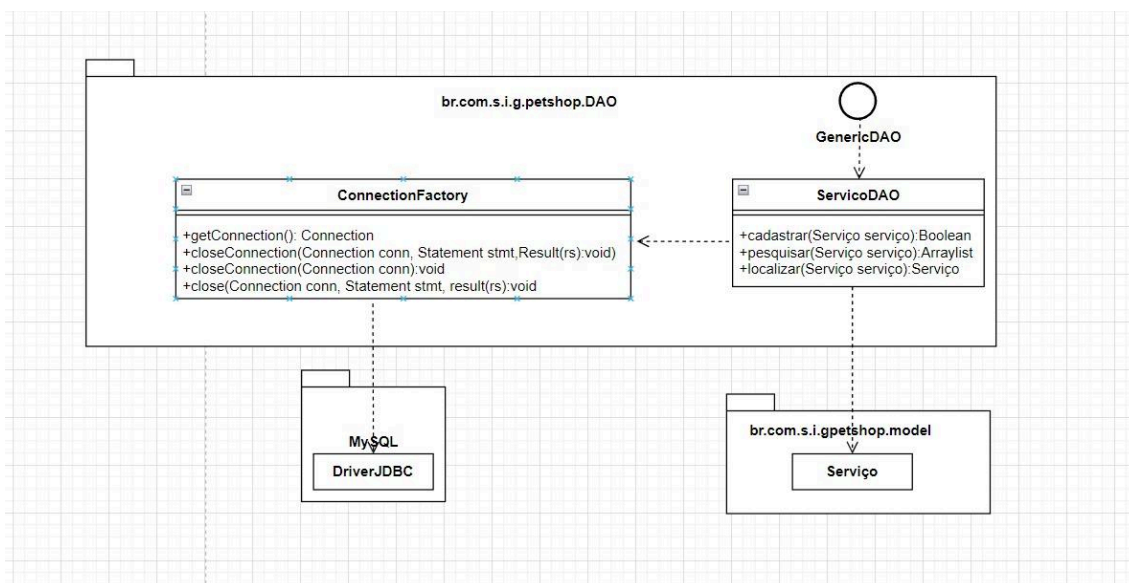
```

packageDiagram
    package br.com.s.i.g.petshop.DAO {
        interface ConnectionFactory {
            +getConnection(): Connection
            +closeConnection(Connection conn, Statement stmt, Result(rs):void)
            +closeConnection(Connection conn): void
            +close(Connection conn, Statement stmt, result(rs):void
        }
        interface AnimalDAO {
            +cadastrar(Animal animal): Boolean
            +exibirEspecie(Animal animal): Arraylist
            +alterar(Animal animal): Boolean
            +localizar(Animal, animal): Animal
        }
    }
    package MySQL {
        class DriverJDBC
    }
    package br.com.s.i.g.petshop.model {
        class Animal
    }
    br.com.s.i.g.petshop.DAO..ConnectionFactory
    br.com.s.i.g.petshop.DAO..AnimalDAO
    MySQL..DriverJDBC
    br.com.s.i.g.petshop.model..Animal
    br.com.s.i.g.petshop.DAO..AnimalDAO-->ConnectionFactory

```

Diagrama de Classe Camada DAO Classe Serviço.

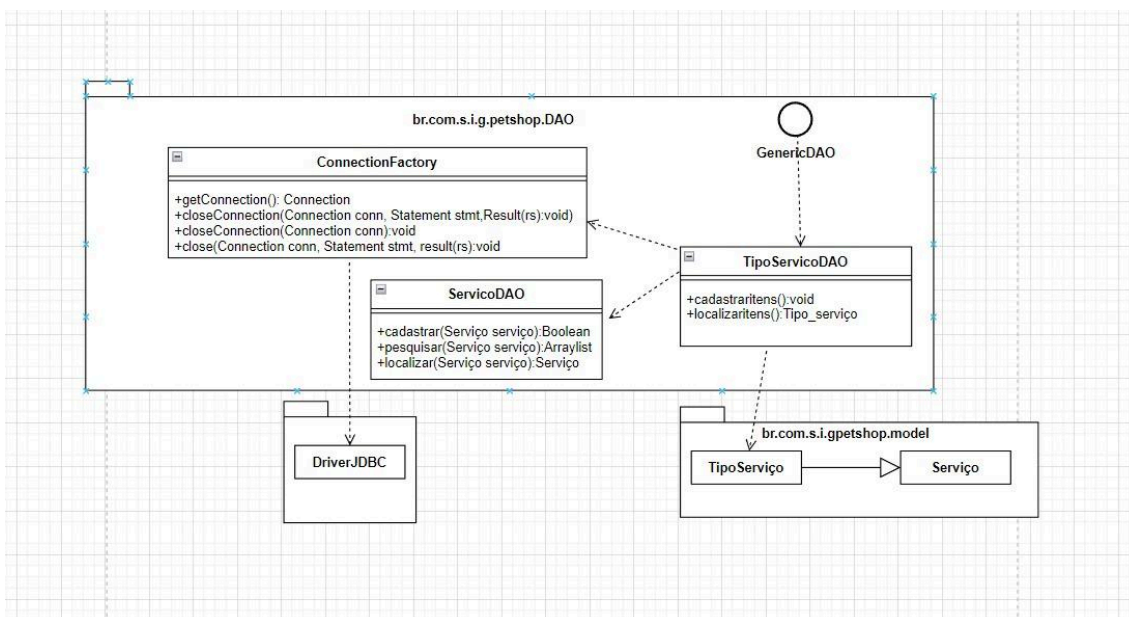
Figura 17 - DAO Serviço



Fonte: O Autor

Diagrama de Classe Camada DAO Classe TipoServiço.

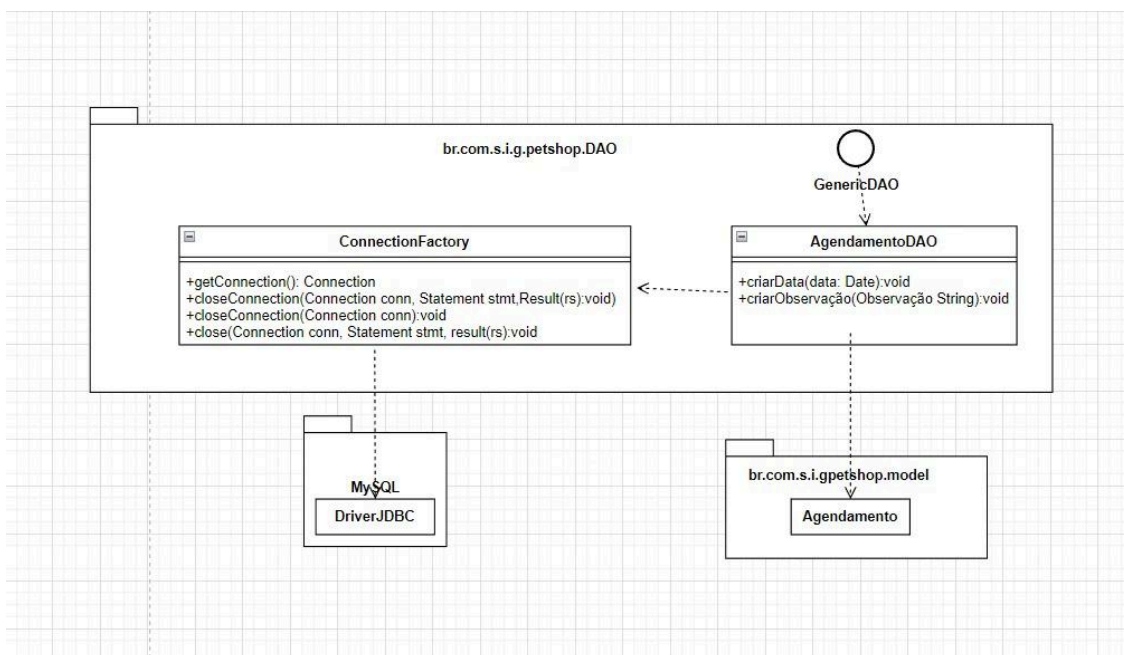
Figura 18 - DAO TipoServiço



Fonte: O Autor

Diagrama de Classe Camada DAO Classe Agendamento.

Figura 19 - DAO Agendamento



Fonte: O Autor

3.4 Diagrama de Mapeamento Relacional

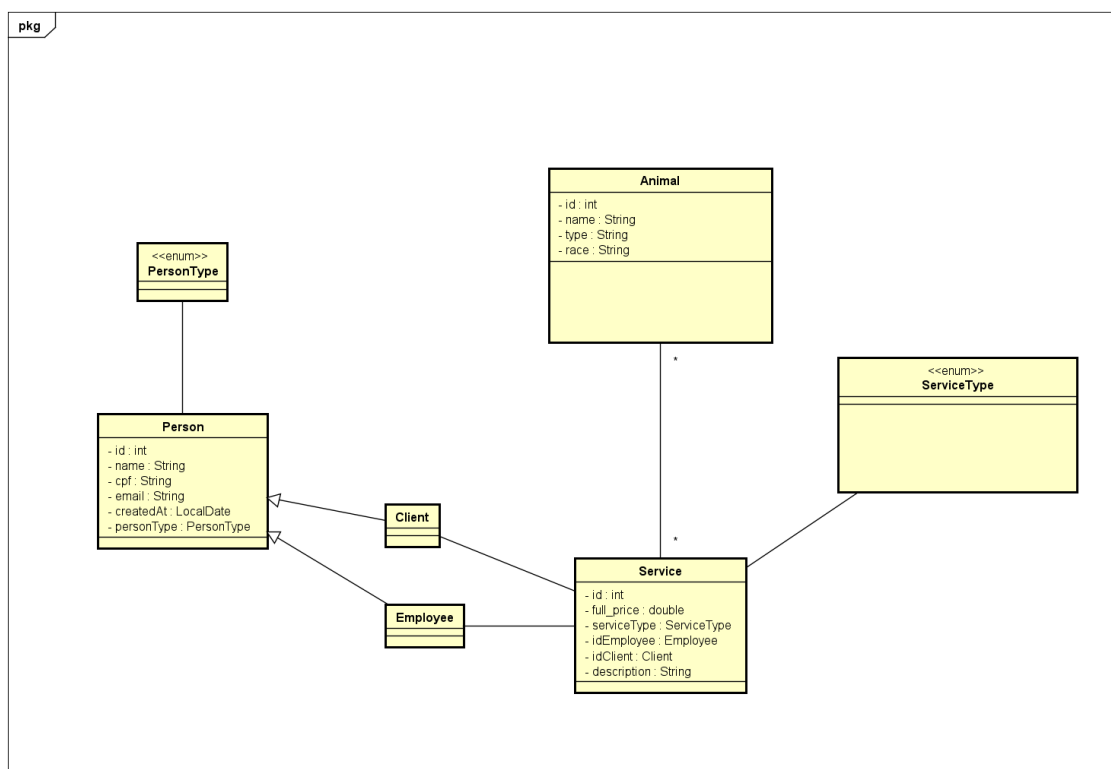
Um banco de dados é uma coleção de dados relacionados, onde dados são fatos conhecidos que podem ser registrados e possuem significado implícito.

Um banco de dados pode ou não ser relacional. Este termo indica que o banco tem relacionamentos entre seus dados (tabelas, entidades).

Os bancos de dados orientados a objetos são conhecidos como bancos de dados de objeto (BDO) e os sistemas de bancos de dados são conhecidos como sistemas de gerenciamento da dados de objeto.

Figura abaixo representa um diagrama de mapeamento objeto-relacional do sistema.

Figura 20 - DER



Fonte: O Autor

3.5 Projeto Físico do Banco de Dados

No projeto físico do banco de dados, estão incluídos os scripts para a criação das tabelas necessárias ao desenvolvimento do sistema. Utilizamos o PostgreSQL (Windows) para o desenvolvimento deste projeto. O PostgreSQL é um sistema gerenciador de banco de dados relacional (SGBD) de código aberto amplamente utilizado em diversas aplicações devido à sua versatilidade e compatibilidade com múltiplas plataformas e linguagens de programação.

3.5.1. Tabela Pessoa

O código abaixo representa a estrutura SQL para criação da tabela Pessoa.

```
CREATE TABLE Pessoa (
```

```
id INT AUTO_INCREMENT PRIMARY KEY,
```

```
nome VARCHAR(100),
```

```
cpf VARCHAR(11),  
email VARCHAR(250),  
createdAt localDate,  
tipoPessoa personType,  
);
```

3.5.2. Tabela Animal

O código abaixo representa a estrutura SQL para criação da tabela Animal.

```
CREATE TABLE Animal (  
id INT PRIMARY KEY,  
name VARCHAR(50),  
type VARCHAR(50),  
race VARCHAR(50)  
);
```

3.5.3. Tabela Service

O código abaixo representa a estrutura SQL para criação da tabela Service.

```
CREATE TABLE Service (  
id INT PRIMARY KEY,  
fullPrice double,  
serviceType ServiceType,  
idEmployee Employee,  
idClient Client,  
description VARCHAR(50)  
);
```

3.6 Especificação do Layout de Telas

Neste capítulo se encontra a representação das telas do sistema, bem como suas determinadas descrições.

Para o desenvolvimento da interface gráfica foi utilizado HTML básico para o desenvolvimento, apoiado em recursos de Javascript e CSS.

3.6.1 - Tela Principal

A figura 21 representa a tela principal, onde o usuário poderá efetuar o login no sistema e assim ter acesso às páginas específicas baseado em seu nível de acesso.

Figura 21-Tela Login

Login

Username:

tella.pereira@gmail.com

Password:

...

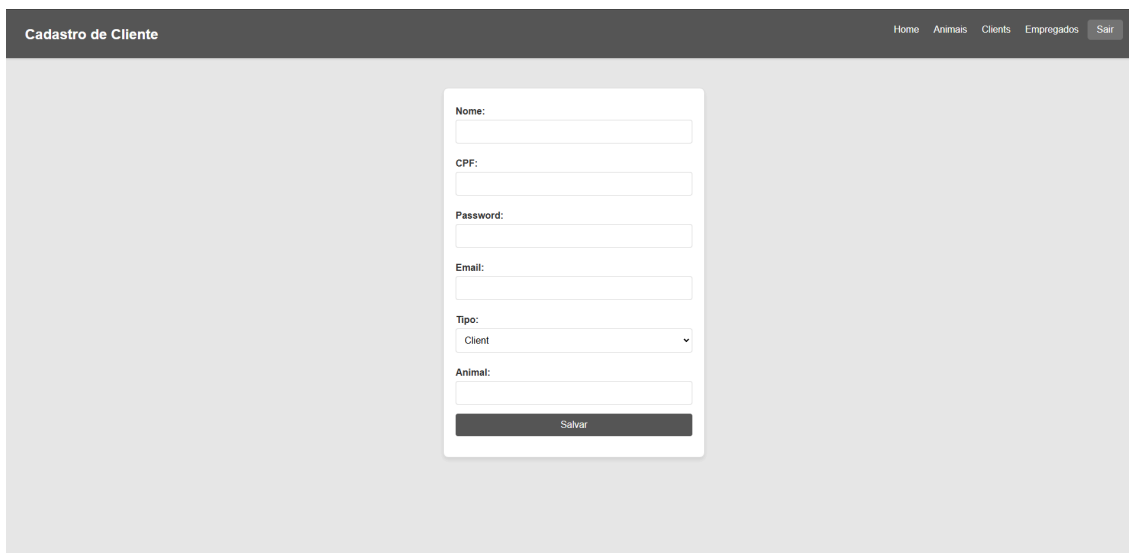
Login

Fonte:O Autor

3.6.2 - Tela de Cadastro de Cliente/Empregado

A figura 22 e a 23 representa a tela de cadastro de Cliente e empregados, pois os tipos de dados inseridos são os mesmos, já que suas classes herdam seus atributos da Classe Pessoa, com poucas diferenças.

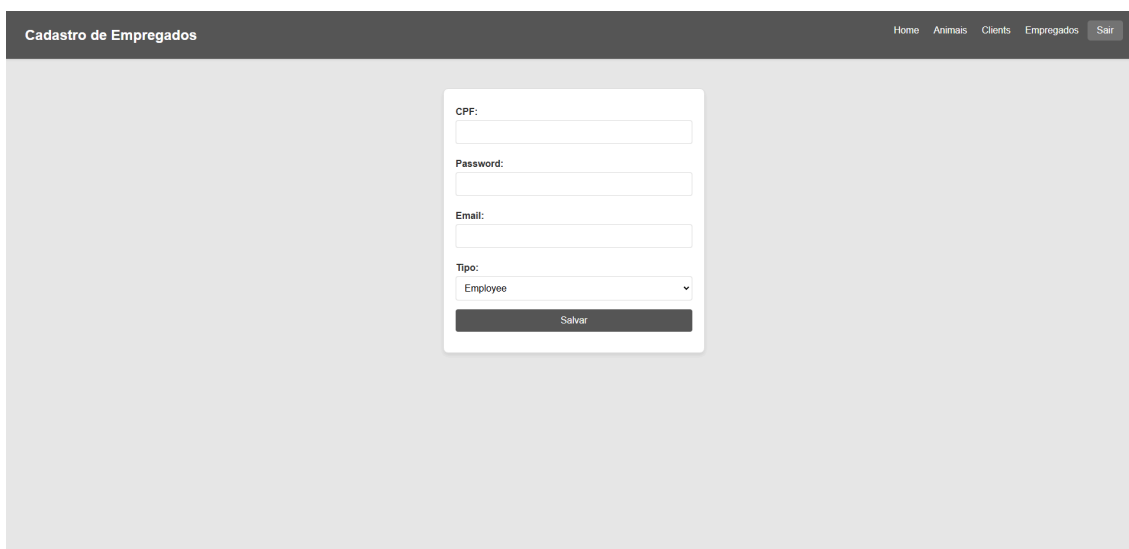
Figura 22 - Tela Cadastro Cliente



The screenshot shows a web application interface for 'Cadastro de Cliente'. At the top, there is a dark header bar with the title 'Cadastro de Cliente' on the left and navigation links 'Home', 'Animais', 'Clientes', 'Empregados', and a 'Sair' button on the right. The main content area is light gray and contains a white form box. The form has the following fields: 'Nome:' (text input), 'CPF:' (text input), 'Password:' (text input), 'Email:' (text input), 'Tipo:' (dropdown menu with 'Client' selected), and 'Animal:' (text input). At the bottom of the form is a dark 'Salvar' button.

Fonte:O Autor

Figura 23 - Tela Cadastro Empregado



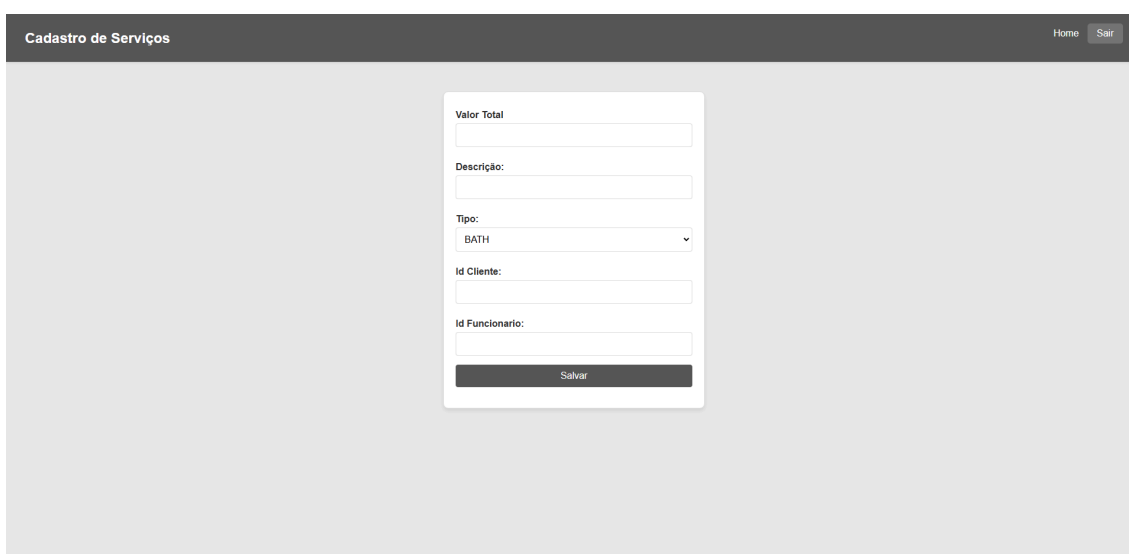
The screenshot shows a web application interface for 'Cadastro de Empregados'. At the top, there is a dark header bar with the title 'Cadastro de Empregados' on the left and navigation links 'Home', 'Animais', 'Clientes', 'Empregados', and a 'Sair' button on the right. The main content area is light gray and contains a white form box. The form has the following fields: 'CPF:' (text input), 'Password:' (text input), 'Email:' (text input), 'Tipo:' (dropdown menu with 'Employee' selected), and an empty text input field at the bottom. At the bottom of the form is a dark 'Salvar' button.

Fonte:O Autor

3.6.3 Tela De Cadastro de Serviços

A figura 24 representa o cadastro da ordem de serviço, a classe conta com os atributos que buscam descrever melhor o serviço.

Figura 24 - Tela Cadastro Serviço



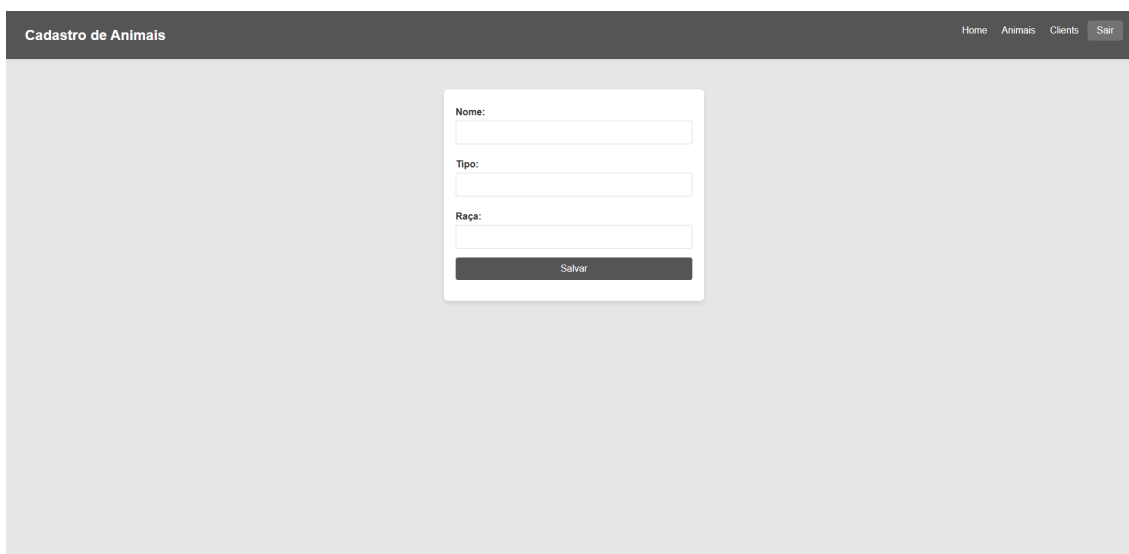
The screenshot shows a web application interface for 'Cadastro de Serviços'. At the top, there is a dark header bar with the title 'Cadastro de Serviços' on the left and 'Home' and 'Sair' links on the right. The main content area is light gray and contains a white form box. The form has the following fields: 'Valor Total' (text input), 'Descrição:' (text input), 'Tipo:' (dropdown menu with 'BATH' selected), 'Id Cliente:' (text input), and 'Id Funcionario:' (text input). At the bottom of the form is a dark 'Salvar' button.

Fonte:O Autor

3.6.4 Tela de Cadastro de Animais

A Figura 25 representa o que é necessário para o cadastro de um animal no sistema, com atributos simples é possível cadastrar diversos animais por ela.

Figura 25 - Tela de Cadastro de Animais

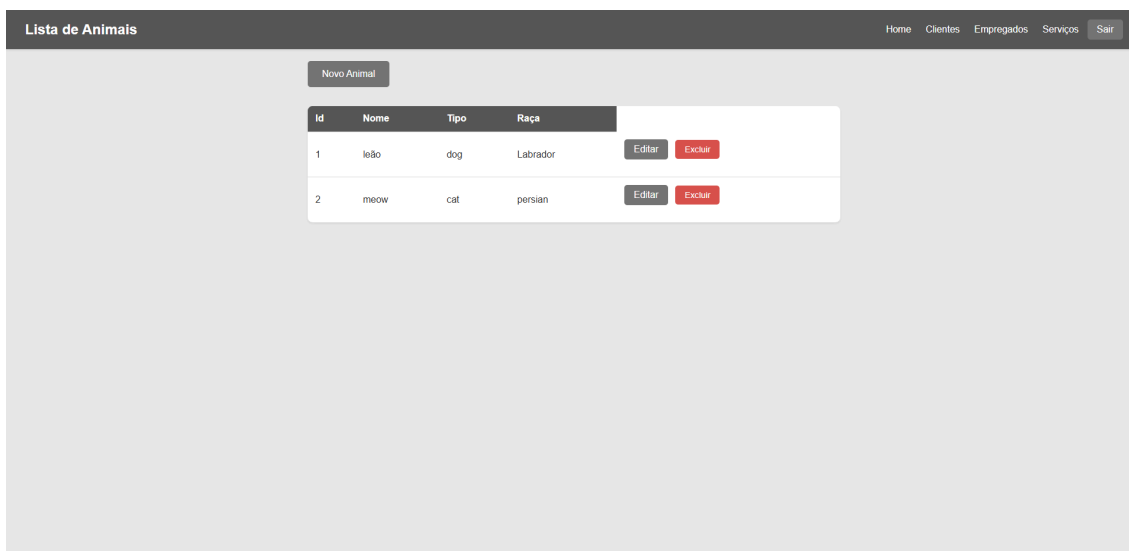


Fonte:O Autor

3.6.5 Tela de Listagem de Animais

A tela de listagem mostra todos os animais registrados no sistema, dando ao usuário o controle necessário sobre esse registro.

Figura 26 - Tela de Listagem Animais



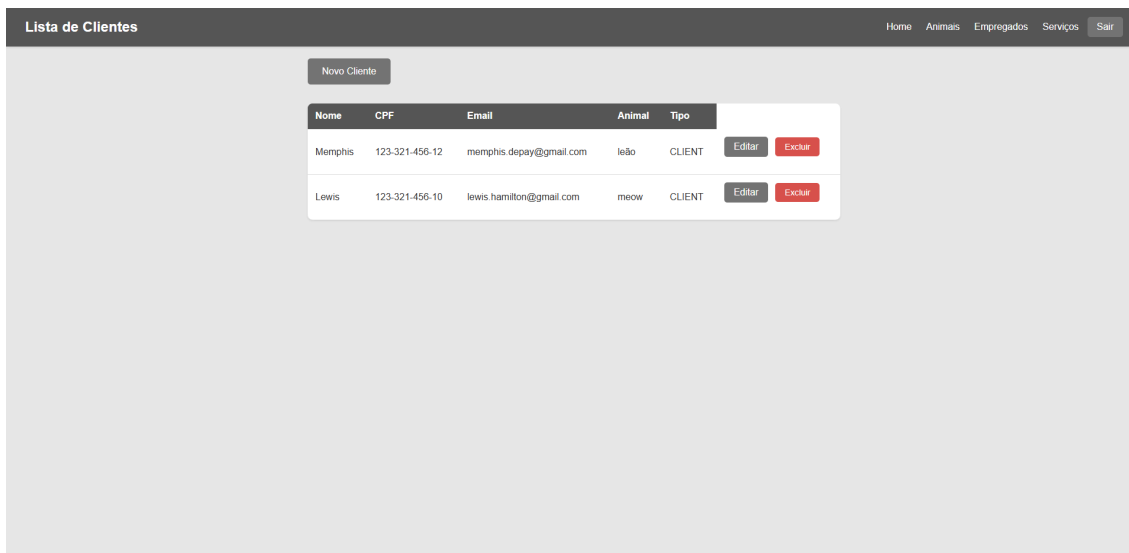
Id	Nome	Tipo	Raça	
1	leão	dog	Labrador	Editar Excluir
2	meow	cat	persian	Editar Excluir

Fonte:O Autor

3.6.6 Tela De Listagem de Clientes

A Tela de Clientes lista todos os clientes no sistema, seus dados, qual animal pertence a ele e a partir do cliente se pode puxar um para a ordem de serviço.

Figura 27 - Tela de Listagem De Clientes



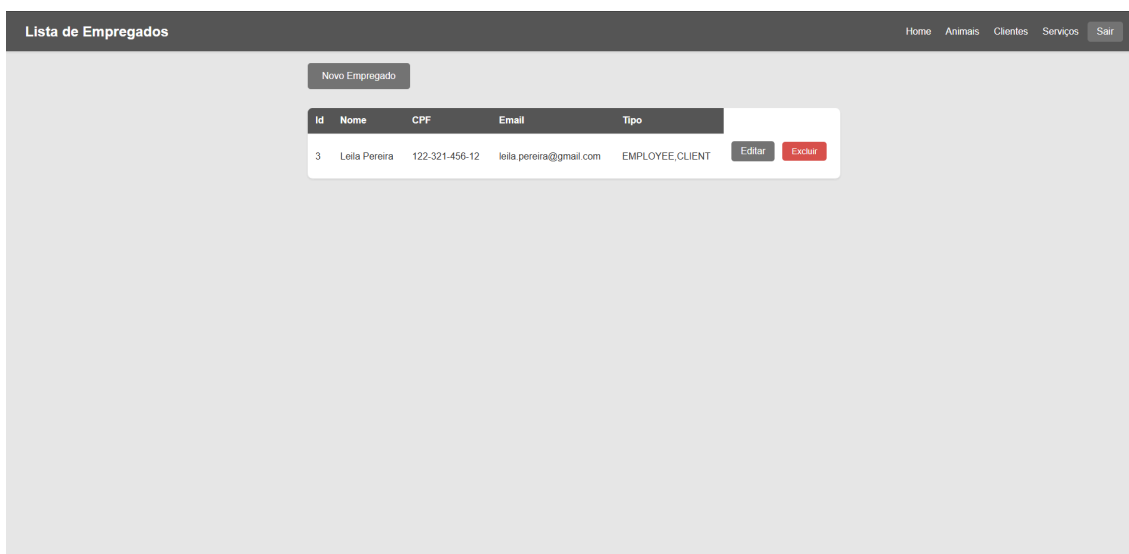
Nome	CPF	Email	Animal	Tipo		
Memphis	123-321-456-12	memphis.depai@gmail.com	leão	CLIENT	Editar	Excluir
Lewis	123-321-456-10	lewis.hamilton@gmail.com	meow	CLIENT	Editar	Excluir

Fonte:O Autor

3.6.7 Tela de Listagem de Empregados

A tela de Empregados, assim como clientes, representa todos que estão cadastrados no sistema, com a única diferença em seu nível de acesso.

Figura 28 - Listagem de Empregados



Id	Nome	CPF	Email	Tipo		
3	Leila Pereira	122-321-456-12	leila.pereira@gmail.com	EMPLOYEE.CLIENT	Editar	Excluir

Fonte:O Autor

3.6.8 Tela de Listagem de Serviços

A tela de serviços mostra qual ordem ainda está aberta, qual o funcionário e cliente registrados na ordem, os valores e o tipo de serviço.

Figura 29 - Tela de Listagem de Serviços

Lista de Serviços						Home Animais Clientes Empregados Sair	
Novo Serviço							
Id	Valor Total	Descrição	Tipo de Serviço	Id do Funcionario	Id do Cliente		
1	40	Banho	BATH	3	1	Editar	Excluir

Fonte: O Autor

3.7 Diagrama de Classes - Interface Homem-Máquina

De acordo com Góes (2014), as interfaces permitem que objetos externos ao sistema possam colaborar com uma ou mais classes do sistema.

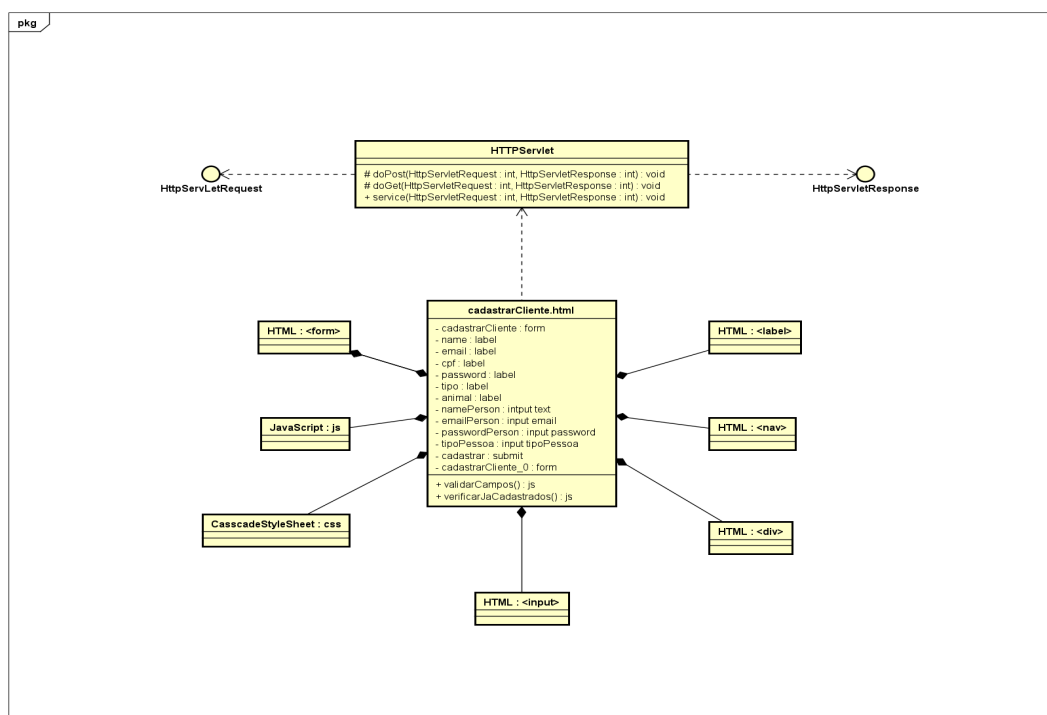
Também são exemplos de interface, nesse caso denominadas interfaces gráficas, as interações que os usuários fazem com a tela do sistema, por exemplo, selecionar uma opção.

Interfaces são coleções de operações que especificam serviços de um componente. É por meio delas que os componentes se comunicam com o mundo externo, seja para oferecer ou receber serviços. As interfaces fazem as associações entre os componentes do software (GÓES, 2014).

Os diagramas que virão a seguir representam a conexão das classes na camada VIEW.

A figura 30 representa o cadastro de Cliente.

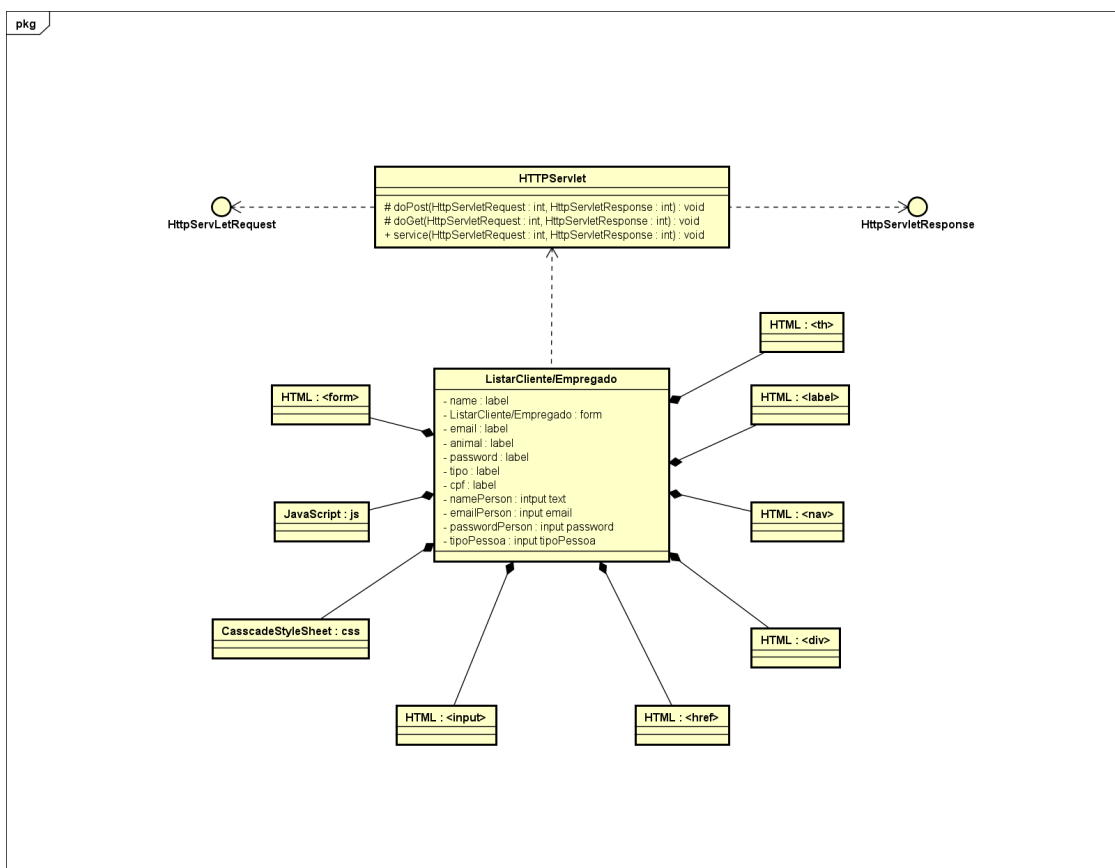
Figura 30 - DCIHM - Cadastrar Cliente



Fonte: O Autor

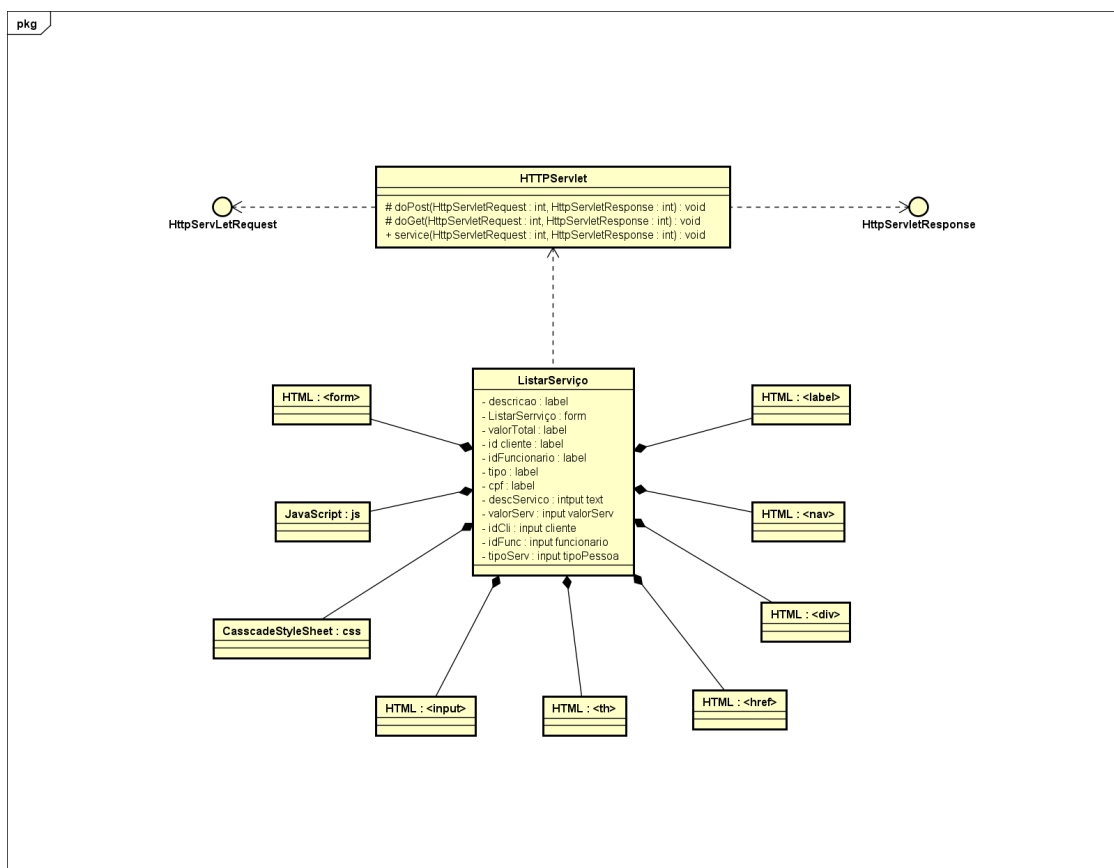
A figura 31 demonstra a listagem exemplo de cliente e suas conexões.

Figura 31 - DCIHM - Listar Cliente



Fonte: O Autor

Figura 32 - DCIHM - Listar Serviço



Fonte:O Autor

3.8 Diagrama de Sequencia - Modelo MVC

Este diagrama procura determinar a sequência de eventos que ocorrem em um determinado processo, identificando quais métodos devem ser disparados entre os atores e objetos envolvidos e em que ordem (GUEDES, 2014).

De acordo com Góes (2014), de preferência, devem ser construídos após os diagramas de caso de uso e classes, pois seus principais objetivos são: documentar casos de uso e demonstrar como os objetos do sistema se comunicam por meio de mensagens em ordem temporal.

Fonte: O Autor

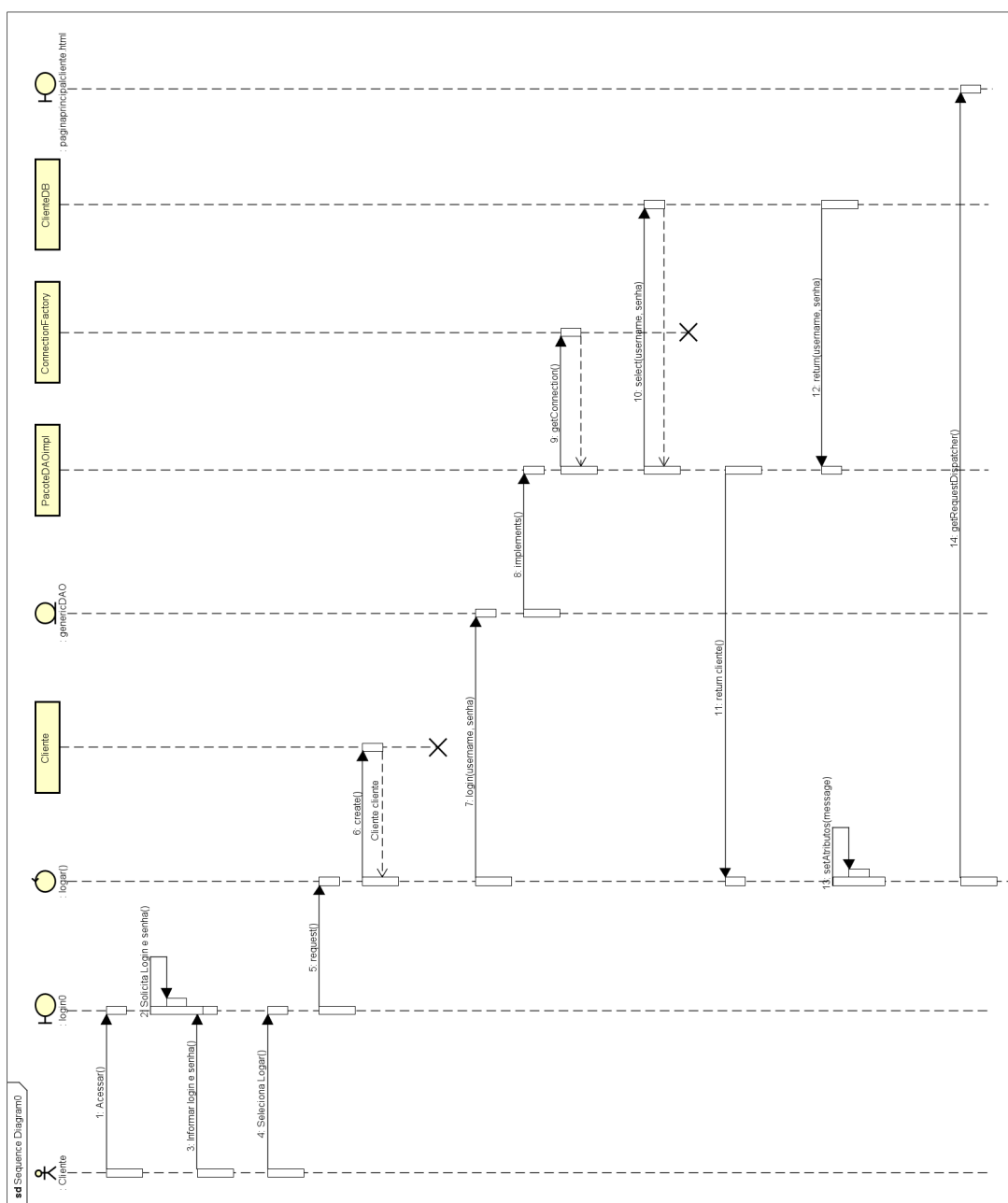
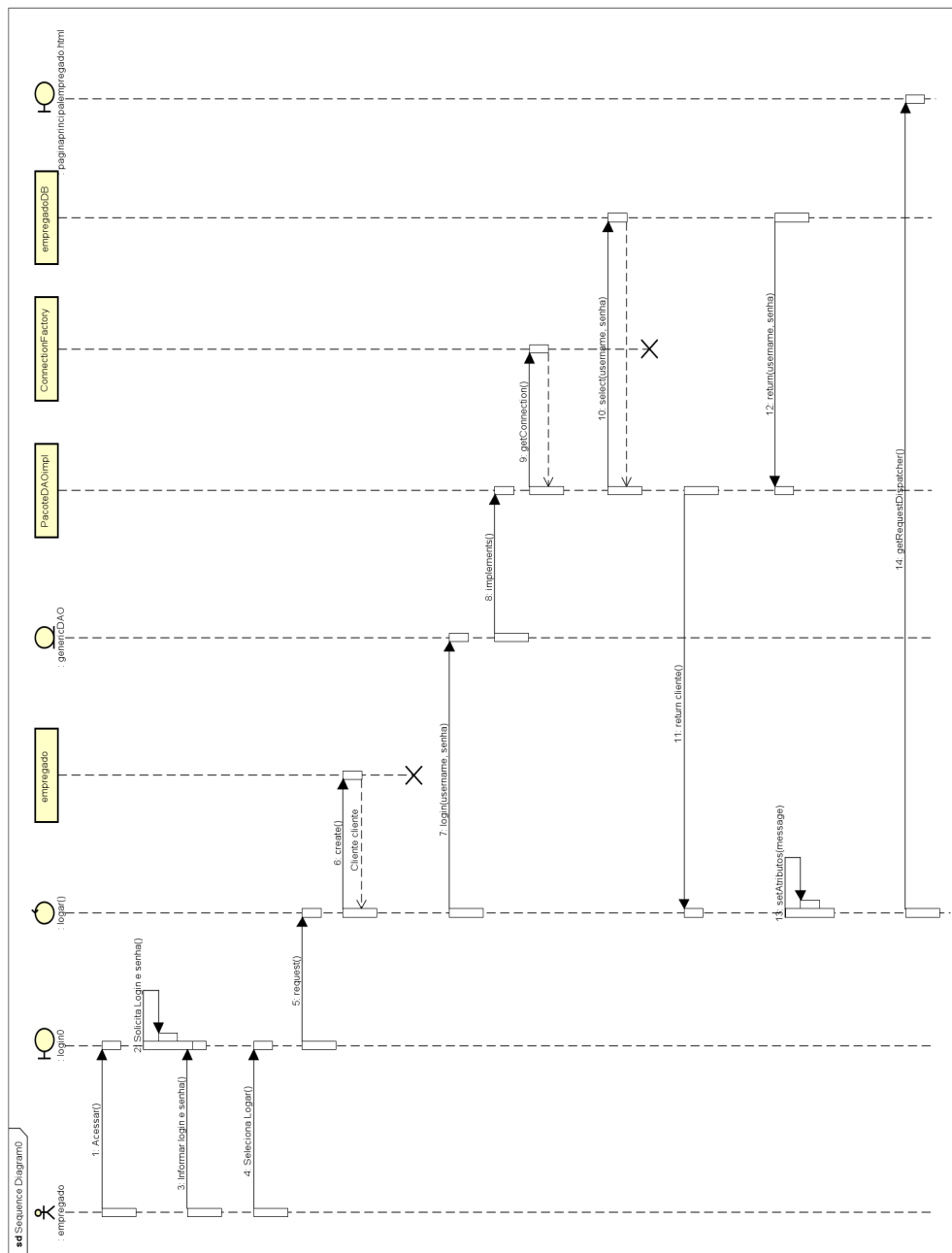
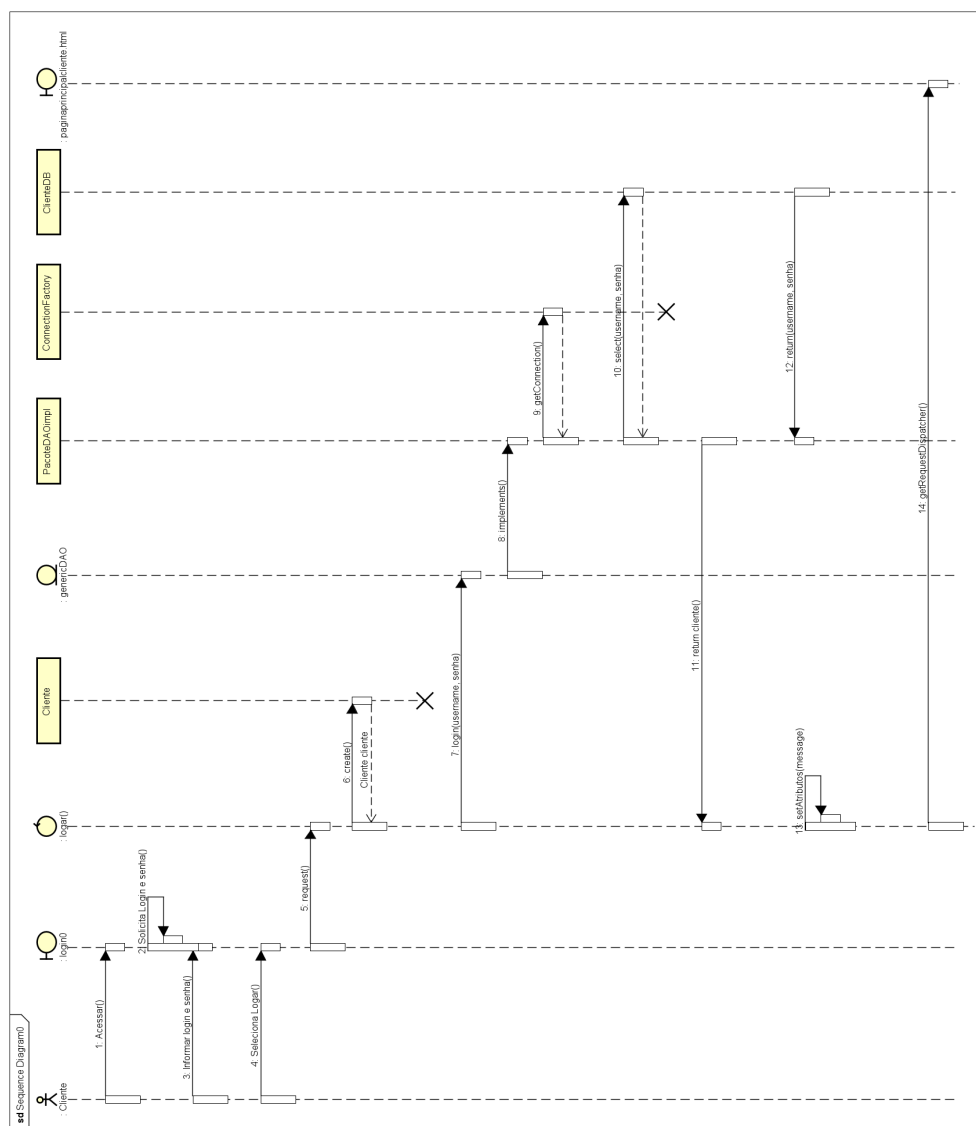


Figura 34 - Login Empregado



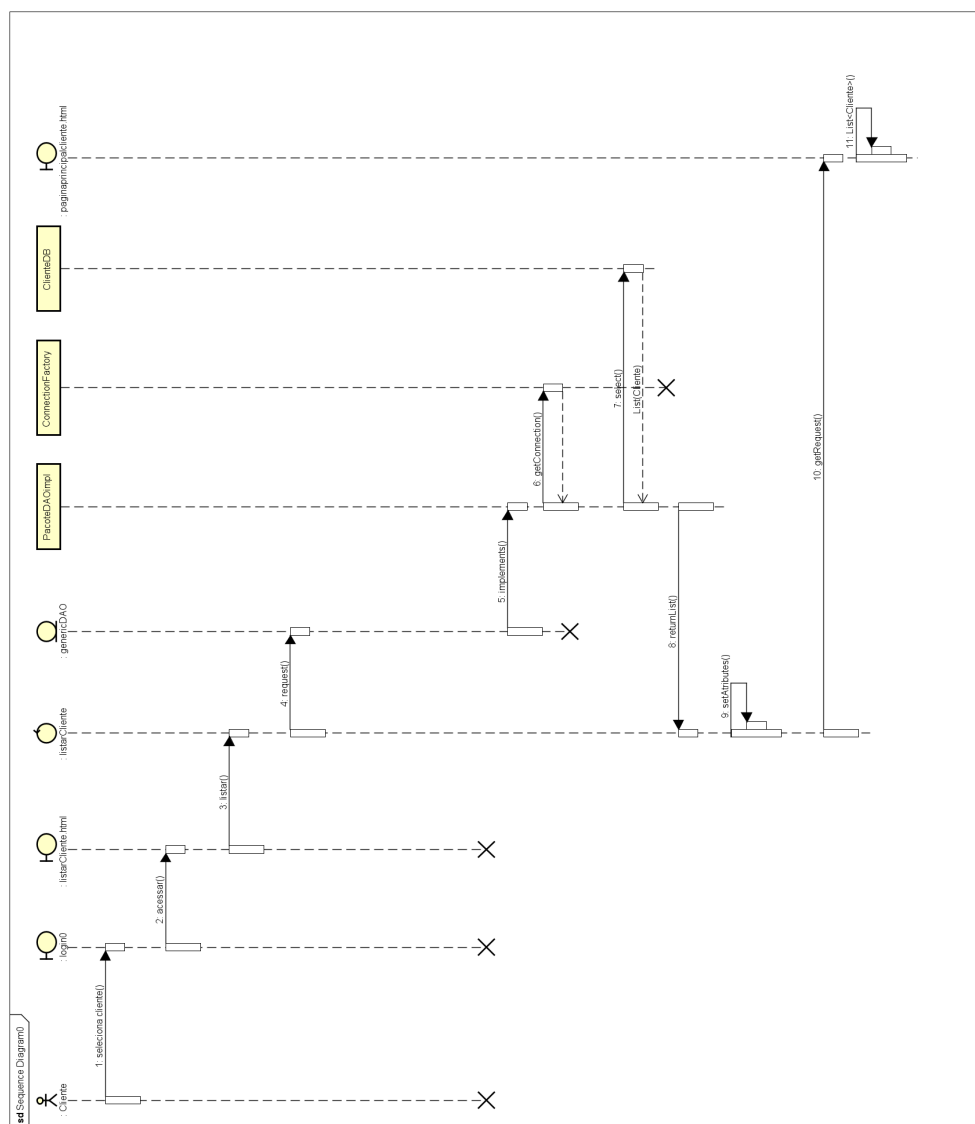
Fonte: O Autor

Figura 35 - Cadastrar Pessoa(Cliente/Funcionário)



Fonte:O Autor

Figura 36 - Sequência Listar Cliente/Funcionário



Fonte:O Autor

A mesma lógica é utilizada pelos outros listar, as requisições e afins.

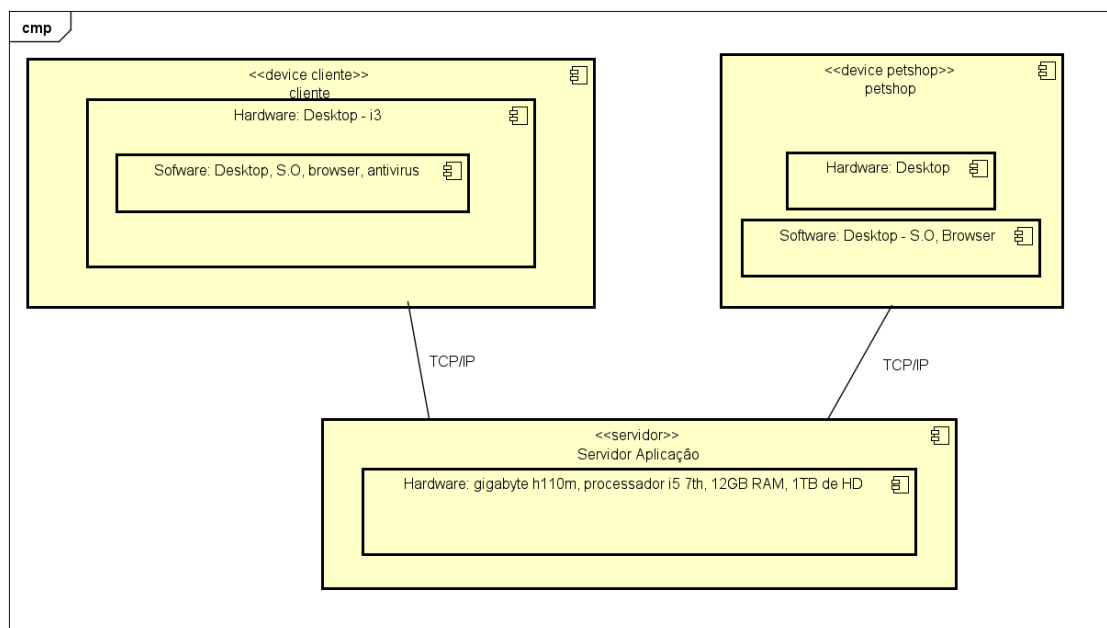
3.9 Diagrama de Implantação de Hardware e Software

O diagrama de implantação especifica um conjunto de construções que podem ser utilizadas para definir a arquitetura de execução de sistemas de informação.

Ele foca a organização da arquitetura sobre o qual o software será implantado e executado em termos de hardware, software e redes de comunicação (GÓES, 2014).

De acordo com Guedes (2014), o diagrama de implantação é o diagrama com a visão mais física da UML.

Figura 37 - Diagrama de implantação de Hardware



Fonte:O Autor

3.10 Controle de Acesso e Segurança

A segurança da informação é a proteção da informação quanto a vários tipos de ameaças, de modo a garantir a continuidade do negócio, minimizar o risco para o negócio, maximizar o retorno sobre o investimento e as oportunidades de negócio (NBR ISO/IEC 27002:2005).

O fornecimento de senhas pelos administradores de sistemas e utilização de senhas pelos usuários é uma parte específica da política de segurança, de grande importância para as organizações (GEUS E NAKAMURA, 2007).

A fim de adequar o sistema as normas da NRB ISO/IEC 27002:2005, o sistema possui controle de sessão e o controle de acesso é feito através de uma tela de acesso onde o usuário informa o seu username e sua senha, o sistema valida os dados, verifica o tipo de usuário e redireciona para página home.

4. METODOLOGIAS E TECNOLOGIAS USADAS

A elaboração deste projeto tem sua fundamentação teórica, em pesquisas na literatura pertinente, videoaulas e sites na internet.

A metodologia de programação é Orientada a Objetos, e a análise e desenvolvimento seguiu os preceitos da UML.

O sistema foi desenvolvido para a web com a ferramenta IntelliJ IDEA Community 2024.2.4, plataforma Spring Boot, na versão Java 17 e para o front-end foi utilizado o Visual Studio Code com html, css e Js.

A parte de diagramas da UML foi criada com a ferramenta Astah Community 10.0.

4.1 TECNOLOGIAS UTILIZADAS

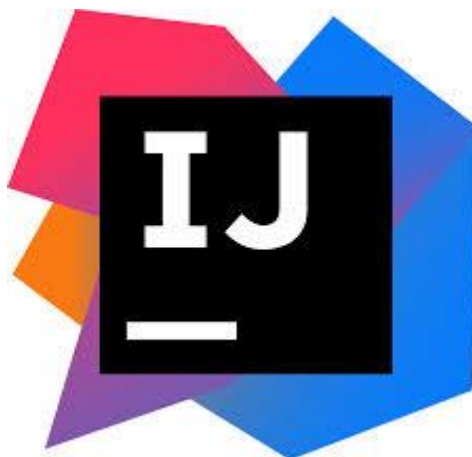
IntelliJ 2024.2.4 - O IntelliJ IDEA é amplamente conhecido pelo suporte avançado a Java, mas a IDE também abrange diversas outras linguagens de programação, tornando-se uma ferramenta versátil para desenvolvedores.

Ele suporta Kotlin, uma linguagem integrada de forma nativa, além de Groovy e Scala, frequentemente usadas em ambientes de desenvolvimento corporativo.

Também oferece suporte para Python (via plugin), JavaScript e TypeScript, atendendo à crescente demanda por desenvolvimento web, assim como linguagens de marcação e estilos como HTML, CSS e Sass.

Para desenvolvedores que trabalham com dados, há suporte para SQL, XML e JSON, com ferramentas de análise e visualização incluídas.

Figura 38 - Logo IntelliJ



Fonte:Google Imagens

Java JDK 17 - O Java Development Kit (JDK) 17 é uma versão de suporte a longo prazo (LTS) lançada pela Oracle em 14 de setembro de 2021.

Ele introduz novas funcionalidades e melhorias significativas na linguagem Java, na plataforma e nas bibliotecas, além de trazer maior estabilidade e segurança.

Figura 39 - Logo Java



Fonte:Google Imagens

Spring Boot - O **Spring Framework** é um dos frameworks mais populares e amplamente utilizados na construção de aplicações Java.

Ele fornece um ecossistema robusto para desenvolver aplicações empresariais, web, microservices e muito mais.

Com foco em simplicidade, flexibilidade e modularidade, o Spring facilita o desenvolvimento de aplicações complexas, oferecendo suporte a diversos padrões e práticas de mercado.

Figura 40 - Spring



Fonte: Google Imagens

HTML5 - HTML é uma abreviação de Hypertext Markup Language - Linguagem de Marcação de Hipertexto, é uma linguagem para publicação de conteúdo (texto, imagem, vídeo, áudio e etc) na Web, estando atualmente na sua versão 5. O HTML5 fornece ferramentas para a CSS e o Javascript fazerem seu trabalho da melhor maneira possível.

Figura 41 - HTML5



Fonte: Google Imagens

CSS - O CSS formata a informação entregue pelo HTML. Essa informação pode ser qualquer coisa: imagem, texto, vídeo, áudio ou qualquer outro elemento criado. Essa formatação na maioria das vezes é visual.

Figura 42 - CSS



Fonte: Google Imagens

Visual Studio Code - O **Visual Studio Code (VS Code)** é um editor de código leve, gratuito e de código aberto, desenvolvido pela **Microsoft**. Ele é amplamente utilizado por desenvolvedores para escrever, depurar e executar código em uma ampla variedade de linguagens de programação e frameworks. Sua popularidade é impulsionada pela flexibilidade, desempenho e uma vasta coleção de extensões disponíveis.

Figura 43 - VS CODE



Fonte: Google Imagens

Astah UML - O Astah é uma ferramenta de modelagem amplamente utilizada para criar diagramas e modelos visuais no desenvolvimento de software. É especialmente popular para diagramas baseados na UML (Unified Modeling Language), mas também suporta outros tipos de diagramas, como ER (Entidade-Relacionamento) e diagramas de sequência.

Figura 44 - Astah UML



Fonte: Google Imagens

5. CONSIDERAÇÕES FINAIS

A implementação do sistema fará com que o atendimento melhore, a organização dos clientes e pets ficará melhor.

A utilização de ferramentas modernas para desenvolvimento faz com que os usuários possam acessar o sistema de lugares distantes desde que tenham acesso à internet.

6. REFERÊNCIAS BIBLIOGRÁFICAS

APACHE FRIENDS, (2015) XAMPP. Disponível em:

<www.apachefriends.org/pt_br/index.html>. Acesso em 07 set. 2015.

Associação Brasileira de Normas Técnicas (ABNT). NBR ISO/IEC 27002:2005 –

Tecnologia da informação – Código de prática para a gestão da segurança da informação. Rio de Janeiro: ABNT, 2005.

BALDUÍNO, Plínio, Dominando Javascript com jQuery. 1.ed. São Paulo: Casa do

Código, 2012

BENTO, Evaldo Junior Desenvolvimento Web com PHP e MySQL. 1. ed. São

Paulo: Casa do Código, 2013.

BEZERRA, Eduardo Princípios de análise e projeto de sistemas com UML: 3. ed.

Rio de Janeiro: Elsevier Editora Ltda., 2015.

BONOME, Karoline S. et al. Disseminação do uso de aplicativos móveis na atenção à saúde. XIII Congresso Brasileiro em informática em Saúde – CBIS, 2012.

BRUMITT, Jason; JOBST, Erin E. Casos Clínicos em Fisioterapia Ortopédica. 1. ed. Porto Alegre: Mcgraw Hill, 2010.

CARBONIERI, Fernando Segundo o CFM, como deve ser feita uma anamnese

adequada?. Disponível em:

<<http://academiamedica.com.br/segundo-o-cfm-como->

[deve-ser-feita-uma-anamnese-adequada/](http://academiamedica.com.br/segundo-o-cfm-como-)>. Acesso em: 10 set. 2015.

Conselho Federal de Fisioterapia e Terapia Ocupacional – COFFITO, Resolução no

386, de 08 de junho de 2011, Dispõe sobre a utilização do método Pilates pelo fisioterapeuta e dá outras providências. Brasília: DOU no. 113, em 14 jun. 2011, Seção

1, p. 182.

CORDEIRO, Gilliard Aplicações Java para web com JSF e JPA 1.ed. São Paulo:

Casa do Código, 2012.

DEITEL, Abbey, DEITEL Harvey e DEITEL Paul Android para Programadores:

Uma abordagem baseada em aplicativos. 2. ed. Porto Alegre: Bookman Editora Ltda, 2015.

EIS, Diego e FERREIRA, Elcio HTML5 - Curso W3C Escritório Brasil, Disponível em:

< <http://www.w3c.br/pub/Cursos/CursoHTML5/html5-web.pdf>>. Acesso em 07. Set.

2015.

ELSMARI, Ramez e NAVATHE, Shamkant B. Sistemas de Banco de Dados 6.ed.

113

São Paulo: Pearson Education do Brasil, 2011.

FOWLER, Martin UML Essencial: um breve guia para a linguagem-padrão de modelagem de objetos. 3. ed. Porto Alegre: Bookman, 2005.

GEUS, Paulo Lício e NAKAMURA, Emilio Tissato Segurança de Redes em Ambientes Cooperativos 1.ed. São Paulo: Novatec Editora Ltda, 2007.

GÓES, Wilson Moraes Aprenda UML por meio de estudos de caso. 1. ed. São Paulo: Novatec Editora Ltda, 2014.

GONÇALVES, Edson. Desenvolvendo aplicações web com JSP, Servlet, Java Server Faces, Hibernate, EJB3 persistence, Ajax. 1, ed. Rio de Janeiro: Ciência Moderna, 2007.

GUEDES, Gilleanes Thorwald Araujo UML 2: Uma abordagem prática. 2. ed. São

Paulo: Novatec Editora Ltda., 2011.

GUEDES, Gilleanes Thorwald Araujo UML 2: Guia prático. 2. Ed. São Paulo: Novatec Editora Ltda, 2014.

iMASTERS (2015), Utilizando HttpClient e Gson no Android Studio. Disponível

em: <<http://imasters.com.br/linguagens/java/usando-httpclient-e-gson-android-studio/?trace=1519021197&source=single>>. Acesso em: 10 set. 2015.

JASPERSOFT (2015). Jasper Reports Library. Disponível em:

<<http://community.jaspersoft.com/project/jasperreports-library>>. Acesso em: 10 set.

2015.

LARMAN, Craig Utilizando UML e Padrões: Uma introdução à análise e ao projeto

orientados a objetos e ao desenvolvimento iterativo. 3. ed. Porto Alegre: Bookman,

2005.

MICROSOFT (2015), Disponível em: <<https://azure.microsoft.com/pt-br/overview/what-is-azure/>> Acesso em: 07 set. 2015.

MILANI, André PostgreSQL: Guia do Programador 1.ed. São Paulo: Novatec Editora Ltda., 2008.

MINISTÉRIO DA SAÚDE (2014). Saiba o que é mito e o que é verdade sobre dor

de coluna. Disponível em:
<<http://www.brasil.gov.br/saude/2014/09/saiba-o-que-e-mito-e-o-que-e-verdade-sobre-dor-de-coluna>>. Acesso em: 23 de mar. 2015.

MONTEIRO, João Bosco Google Andoid: Crie aplicações para celulares e tablets.

1. ed. São Paulo: Casa do Código, 2012.

NEUMANN, Donald A. Cinesiologia do Aparelho Musculoesquelético: Fundamentos para Reabilitação. 2. Ed. Rio de Janeiro: Elsevier Editora Ltda., 2011.

ORACLE (1) (2015), NetBeans IDE Features, Disponível em:

114

< https://netbeans.org/features/index_pt_BR.html>. Acesso em: 07 set. 2015.

ORACLE (2) (2015), Tecnologias Java, Disponível em:

< <http://www.oracle.com/br/java/technologies/all/index.html> >. Acesso em: 07 set.

2015.

ORACLE (3) (2015) MySql WorkBench. Disponível em: <

<https://www.mysql.com/products/workbench/>>. Acesso em: 07 set. 2015.

ORACLE (4) (2015) MySQL. Disponível em:

<<http://www.oracle.com/br/products/mysql/overview/index.html>>. Acesso em 20 set.

2015.

SILVA, Maurício Sammy, Java Script: Guia do programador. 1. ed. São Paulo: Editora Novatec Ltda, 2010.

SBROCCO, José Henrique Teixeira de Carvalho UML 2.5 com Enterprise Architect 10 Modelagem visual de projetos orientada a objetos. 1. Ed. São Paulo:

Editora Érica Ltda, 2014.

SCHIMTZ, Daniel Boostat 3: Framework front-end para desenvolvimento web e

mobile. 1.ed. British Columbia: LeanPub, 2014.

W3C, CSS3. Disponível em
<<https://www.w3.org/standards/webdesign/htmlcss>>

Acesso em: 07 set. 2015.