Server Feeder - Sistema Web para Cadastro e Gerenciamentos de Alimentadores Automáticos para Pets

Joel Mendes de Oliveira, Rodolfo Francisco de Oliveira[1]

[1]Instituto Federal de Educação, Ciência e Tecnologia - Campus Hortolândia (IFSP)
CEP: 13183-250 - Hortolândia - SP - Brasil

joelmendes2006@hotmail.com, rodolfo.oliveira@ifsp.edu.br

Abstract. *The number of pets has been growing in Brazil and around the world in recent years. To assist pet owners in replenishing food and water for their animals, this work aims to present a web system called *Server Feeder*, which allows the registration and configuration of Automatic Feeders. A *Server Feeder* user can register on the platform and manage multiple automatic feeders. The system was developed using MySQL, HTML, CSS, Bootstrap, and PHP. The integration of *Server Feeder* with Automatic Feeders could be a topic for future research and development.*

Resumo. *O número de animais de estimação vem crescendo no Brasil e no mundo ao longo dos últimos anos. Afim de auxiliar os tutores à repor os alimentos e água para tais animais, o presente trabalho visa apresentar um sistema web denominada Server Feeder, a qual permite o cadastro e parametrização de Alimentadores Automáticos. Um usuário do Server Feeder pode se cadastrar na plataforma e cadastrar vários dos seus alimentadores automáticos. O sistema foi desenvolvido utilizando as tecnologias MySQL, HTML, CSS, Bootstrap e PHP. A integração do Server Feeder com os Alimentadores Automáticos poderá ser tema de discussão de trabalhos futuros.*

1. Introdução

O Brasil se encontra na terceira posição, do *ranking* mundial de maior população *pet*. O número de animais de estimação no país é de quase 160 milhões atualmente. Baseando se nos dados de 2022, os cães lideram o *ranking* com 60 milhões, seguido das aves que são 40 milhões, gatos são 30 milhões, peixes ornamentais são 20 milhões e répteis e pequenos mamíferos são 2,5 milhões. Em 2023, o mercado *pet* brasileiro faturou 68,7 bilhões, onde foi obtido um recorde do segmento, com 14% de aumento referente a 2022. Estatisticamente há 1,6 animais de estimação por residência brasileira. Sendo assim, eles realmente são integrantes dos lares e fazem parte das famílias, então, é necessário que tenham alimentos e cuidados de qualidade [ABINPET 2023].

Hoje, o cotidiano familiar é muito corriqueiro, entre idas e vindas de trabalhos e viagens, as pessoas reservam um tempo para os *pets*. Com todos esses contratempos, os *pets* ficam mais tempo sozinhos do que acompanhados por seus donos, onde muitos *pets*, não tem uma boa alimentação, seus alimentos ficam muito exposto. No mercado existe, recurso para que essa ausência de cuidado diminua, que conhecemos como os alimentadores automáticos para *pets*.

Esse trabalho inclui a criação de um servidor, onde serão cadastrados e armazenados dados de alimentadores automáticos para *pets*, o Server Feeder. O Server Feeder permite que o usuário de um alimentador automáticos, cadastre e gerencie seus alimentadores.

O presente trabalho é organizado da seguinte maneira: no Capítulo 2 é apresentado a Introdução do trabalho; no Capítulo 3 apresenta a Fundamentação Teórica do trabalho; o Capítulo 4 apresenta alguns trabalhos correlatos encontrados na literatura; O Capítulo 5 apresenta a proposta do trabalho; o Capítulo 6 apresenta a implementação do trabalho; por fim o Capítulo 7 apresenta as conclusões e perspectivas de trabalhos futuros.

2. Fundamentação Teórica

Este capítulo apresenta os principais conceitos e ferramentas utilizadas ao longo do trabalho, tais como HTML, Javascript, CSS, PHP, MySQL e Boostrap.

2.1. Scrum

Existente desde o início do 1990, porém, ganhou popularidade apenas na década seguinte, tornando- se o método mais bem- sucedido no desenvolvimento de *software*, aumentando em 3,5 vezes a chance de sucesso do produto, apesar de ser usada para o desenvolvimento de diversos produtos, e não apenas *software*, como foi no princípio. Esse método tem sido usado em empresas de pequenos, médios e grandes portes, em empresas nacionais e multinacionais [SABBAGH 2022].

O Scrum é simples e leve, e proporciona às pessoas, equipes e organizações, gerarem valores, por meio de soluções adaptadas para variados problemas, por mais complexo que seja. Estruturalmente, o Scrum não é completo, mas isso é de propósito, é definido apenas as partes essenciais para implementação da teoria do Scrum, que é complementado com a inteligência coletivas das pessoas que o usam [SUTHERLAND 2020].

2.2. HTML

HyperText Markup Language, é o significado da sigla HTML, que em português quer dizer linguagem para marcação de hipertexto. Os documentos para a *Web* possuem texto que são chamados hipertexto. Os documentos para *Web* se interligam por intermédio dos *links*, que costumamos ver nos sites que utilizamos diariamente. O primeiro requisito para criação de páginas *Web*, é o conhecimento em HTML. Como qualquer idioma, a HTML é composta por palavras ou termos, sintaxe e regras que devem ser seguidas, para que o navegador entenda sua escrita e apresente para o usuário da pagina [SILVA 2018].

2.3. Javascript

JavaScript é a linguagem essencial da *Web*. A maioria das aplicações *Web*, dispositivos e todos os navegadores possuem interpretadores JavaScript, o que a torna a linguagem de programação mais utilizada da história [FLANAGAN 2013].

2.4. PHP

PHP *Hypertext Preprocessor* (Pré-Processador de Hipertexto) é uma das linguagens de programação mais utilizadas para páginas *web*. A interação da linguagem PHP com o mundo *web* é muito maior, com referência às outras linguagens de programação, o que concede às páginas estáticas mais dinamismo. Uma outra grande vantagem do PHP, é que é gratuito e pode ser baixado no site https://www.php.net .[NIEDERAUER 2017].

2.5. MySQL Server

O MySQL Server é o Sistema de Gerenciamento de Banco de Dados (SGBD) *open source* mais utilizado no mundo. Ao todo, 17 dos 20 maiores provedores de *software* do mundo usam MySQL em suas aplicações. O MySQL Server é gratuito, possibilitando que os fornecedores dediquem mais recursos financeiros para o restante -+-+do produto. [MySQL 2012]

2.6. Bootstrap

O Bootstrap é um *framework* JavaScript, HTML e CSS para desenvolvimento com *design* moderno para interface *web* responsiva, alinhada com a filosofia *mobile first*, sendo um dos mais populares do mundo. Torna o desenvolvimento do *frontend* mais fácil e rápido, sendo possível de ser utilizado por desenvolvedores com nível mínimo de conhecimento, e adaptável para todos os dispositivos e projetos de qualquer complexidade. [MAUJOR 2018]

3. Trabalhos Correlatos

Nesta seção, é relatado alguns trabalhos onde foram desenvolvidos alguns alimentadores automatizados para *pets*. A pesquisa pela busca desses trabalhos fora realizadas através da plataforma do Google Academy. Os seguintes termos foram utilizados para pesquisa: 'alimentador automatizado' e 'alimentador para *pets* baseado em IoT'. Os trabalhos foram pesquisados no mês de dezembro de 2023. Nas pesquisas realizadas não foram encontrados trabalhos semelhantes ao Server Feeder, porém, foram encontrados 3 trabalhos correlatos, que podem se alinhar com o Server Feeder, os quais serão explicados nos subtópicos a seguir.

3.1. Sistema supervisório parar alimentador automático para cachorros e gatos em situação de rua na UFERSA

A ideia desse sistema é facilitar o trabalho das pessoas que alimentam cachorros e gatos em situação de rua, que vivem próximos ao *campus* de uma universidade. Já existia um alimentador no local, mas foi realizada um *upgrade* do mesmo. Este consistiu em substituir um microcontrolador Arduíno Mega 2560, onde era realizadas a automações e controles das tarefas de abastecimento de água, iluminação e liberação da comida com a presença do animal e a verificação de disponibilidade de comida, para um módulo *WiFi* ESP8266 NodeMCU ESP-12. Com o *upgrade*, foi implementado no produto o protocolo MQTT (*Message Queuing Telemetry Transport*), que através da Internet, possibilitou a aquisição e o envio de informações do processo remotamente [DANTAS 2022].

3.2. Aplicação do conceito de Internet das coisas no desenvolvimento de um alimentador para animais domésticos

Este trabalho aborda o desenvolvimento de um protótipo de comedouro para auxiliar na alimentação de *pets*. Isso ajudaria a diminuir o descuido com a alimentação dos *pets*, e criar uma rotina de alimentação dos *pets*, evitando o surgimento de doenças patológicas, como a obesidade, por exemplo. Através de uma aplicação *mobile*, é possível realizar a dosagem de alimentos em um recipiente [RAMOS 2023].

3.3. Alimentador automático para cães

Este alimentador automático possui relógios e datas ajustáveis, programação de horas e dosagens para a rotina de alimentação do animal. Foi produzido com canos de PVC, um microcontrolador Arduíno e componentes eletrônicos [BATISTA et al. 2015].

3.4. Comparação das Propostas

A Tabela 1, apresenta uma comparação das características do Server Feeder com os outros trabalhos correlatos.

Na coluna 'Gerenciar vários alimentadores', informa se o projeto possibilita o gerenciamento de mais de um alimentador. Em questão, apenas o Server Feeder possui esse recurso.

'Gerência quantidade mínima' é um recurso que identifica se o alimentador está com alimentos no dispense. Haverá uma configuração em que o sistema alertará o usuário, que o alimento está chegando ao fim, de acordo com a quantidade mínima que for parametrizada para o alimentador. Apenas o Server Feeder possui essa função.

'Gerenciar quantidade máxima' é um recurso que identifica se já excedeu a quantidade máxima suportada pelo alimentador, que também precisa ser parametrizada no cadastro do mesmo. Esse também é um recurso que tem apenas no Server Feeder.

Por possuir uma aplicação *Web*, é necessário que a interface seja responsiva, ou seja, se adapta a qualquer dispositivos e navegadores. Server Feeder possui uma interface responsiva, assim como o Alimentador IOT.

Diferente dos projetos Alimentador IOT e Alimentador UFERSA, o Server Feeder, não possui o módulo Alimentador. O Server Feeder será apenas para gerenciamento dos alimentadores. O módulo alimentador poderá ser desenvolvido em trabalhos futuros.

Tabela 1. Tabela de comparações dos trabalhos correlatos

Projeto	Gerencia vários alimentadores	Gerencia quantidade miníma	Gerencia quantidade máxima	Interface responsiva	Interface Web	Módulo alimentador
Server Feeder	Sim	Sim	Sim	Sim	Sim	Não
Alimentador UFERSA	Não	Não	Não	Não	Não	Sim
Alimentador IOT	Não	Não	Não	Sim	Sim	Sim

4. Apresentação da Proposta

Este capítulo apresenta a proposta do sistema, sob um ponto de vista da análise do mesmo. É apresentado os requisitos funcionais, o Diagrama de Caso de Uso, o Modelo Entidade Relacionamento e o Diagrama de Entidade e Relacionamento.

4.1. Definições de Requisitos Funcionais

Em um documento de requisitos, é descrito as especificações de requisitos de usuário e de sistema. Os requisitos devem ser escritos de forma clara, simples, onde seja fácil de entender e sem equívoco, onde é especificado apenas o funcionamento externo do sistema e suas restrições operacionais. Os requisitos de usuários precisam ser escritos de uma maneira que seja de fácil compreensão para quem for usar o sistema, mesmo que o usuário não tenha conhecimento técnico. Os requisitos de usuários descritos nesta etapa são: Requisitos funcionais e Requisitos não funcionais.[lan SOMMERVILLE 2018] Requisitos Funcionais são declarações dos serviços que serão fornecidos pelo sistema, como será sua reação ao ser solicitado pelo usuário, e em algumas ocasiões, é declarado o que o sistema não deve fazer. [SOMMERVILLE 2011] Já os requisitos não funcionais são as restrições que os serviços do sistema possui. [SOMMERVILLE 2011].

A Tabela 2 apresenta os requisitos funcionais do presente trabalho. Para que o sistema possa ser utilizado, é necessário que o usuário se cadastre no sistema, como relatado na identificação RF01.

Na identificação RF02, a instrução é para que o usuário tenha acesso às funções do sistema, é necessário efetuar o login no sistema.

Na identificação RF03, o acesso do usuário é validado, verificando se o mesmo possui registro no banco de dados, e só é permitido o acesso, se o usuário realmente estiver cadastrado. Validação realizada no nome e senha do usuário, uma etapa obrigatória no sistema, tornando- o mais seguro.

Na identificação RF04, é permitido que o usuário possa realizar alterações em seu dados, por exemplo, senha, e-mail. Não é uma etapa obrigatória.

Na identificação RF05, permite que o usuário cadastre seus alimentadores, uma etapa importante, porque não faria sentido se cadastrar no sistema, sem ter um alimentador para cadastrar e gerenciar, já que esse é o intuito do Server Feeder.

Na identificação RF06, permite que o usuário altere as informações do alimentador, por exemplo, nome, quantidade mínima e máxima. Não é uma etapa obrigatória, mas apenas se usuário desejar.

Na identificação RF07, permite que o usuário exclua o alimentador de sua lista de gerenciamento, também não é uma etapa obrigatória.

Na Tabela 3 apresenta os requisitos não funcionais do presente trabalho. Para que o usuário use o sistema, é necessário que tenha acesso a internet, como relatado na identificação RNF01.

Na identificação RNF02, permite que o sistema se adapte em qualquer resolução e dispositivos.

Na identificação RNF03, permite que apenas usuário cadastrado, acesse os dados do usuário e alimentadores.

4.2. Diagrama de Caso de Uso

O objetivo do diagrama de caso de uso é de apresentar uma visão geral externa das funcionalidade proposta no sistema, para o usuário, porém, não apresenta como tais funcionali-

Tabela 2. Tabela de Requisitos Funcionais

Identificação	Nome	Descrição	Classificação
RF01	Cadastrar Usuário	Cadastrar usuário para que possa usar o sistema	Obrigatório
RF02	Efetuar Login	Permitir acesso ao sistema	Obrigatório
RF03	Validar usuário	Validar acesso ao sistema	Obrigatório
RF04	Alterar usuário	Permitir alterar dados do usuário	Desejável
RF05	Cadastrar alimentador	Permitir inclusão dos alimentadores	Importante
RF06	Alterar alimentador	Permitir alterações dos dados do alimentador	Desejável
RF07	Excluir alimentador	Fazer a exclusão de alimentadores	Desejável

Tabela 3. Tabela de Requisitos Não Funcionais

Identificação	Nome	Descrição
RNF01	Acesso a internet	O usuário precisa de acesso a internet para utilizar o sistema
RNF02	Responsividade	Torna o sistema adaptável a qualquer resolução de tela e dispositivos
RFN03	Segurança	Permite que apenas usuários cadastrado acesse as informações dos alimentadores e usuários

dades serão implementadas. É utilizado na fase de levantamento de requisitos e elicitação do sistema, porém, é consultado constantemente no processo de modelagem, serve de base para outros diagramas. [GUEDES 2018]. A Figura 1, apresenta o Diagrama de Caso e de Uso do Server Feeder. O processo de utilização do sistema, pelo usuário, é seguinte:

- Realizar cadastro pessoal
- Efetuar Login
- Alterar dados do cadastro do pessoal
- Cadastrar alimentador
- Alterar dados do alimentador
- Excluir alimentador

O sistema em si, realiza os seguintes processos:

- Verificar a senha do usuário, quando este, acessa o sistema. Quando a senha está correta, o acesso é liberado, quando não, é informado ao usuário que a senha está errada.
- Nas alterações e exclusões dos dados do alimentador, o sistema realiza a verificação se o mesmo pertence ao usuário.

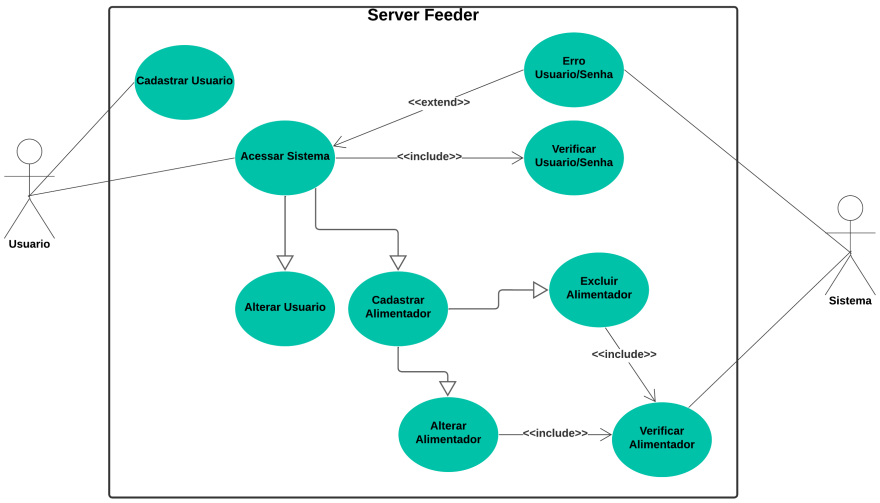


Figura 1. Diagrama de caso de uso - Server Feeder

4.3. Modelo de Entidade Relacionamento e Diagrama de Entidade e Relacionamento

A Figura 2, apresenta o Modelo de Entidade e Relacionamento (MER). O MER é constituído de diagramas, que são utilizados para projetar Bancos de Dados Relacionais, baseando- se em relações de objetos reais onde são representados por entidade e seus relacionamentos.

A entidade Cuidador é a entidade que irá conter as informações (atributos) do cuidador. O atributo ID_CUIDADOR, é o atributo que armazena a chave do cadastro do cuidador. Os atributos LOGRADOURO, NUMERO, COMPLEMENTO, CEP, CIDADE, BAIRRO, UF, se referem aos dados de endereço do usuário. Já o atributo EMAIL não se repete no banco de dados, podendo ser usada por apenas um único usuário. Os atributo USUÁRIO e SENHA são necessários para que o cuidador tenha acesso ao sistema, sendo o atributo SENHA criptografado no banco de dados, para que o sistema tenha uma segurança maior, com os dados do cliente.

A entidade Alimentador é a entidade que irá conter as informações (atributos) do alimentador dos *pets*. O atributo ID_ALIMENTADOR, é o atributo que armazena a chave do cadastro do alimentador, no banco de dados. Os atributos CAPACIDADE_MAXIMA e CAPACIDADE_MINIMA, armazena as quantidades mínima e máxima em kg, que o alimentador suporta. Tais dados serão utilizados para parametrizar o alimentador físico (etapa não contemplada neste trabalho). Já o atributo ID_CUIDADOR identifica a qual cuidador pertence o alimentador, se tratando de uma chave estrangeira.

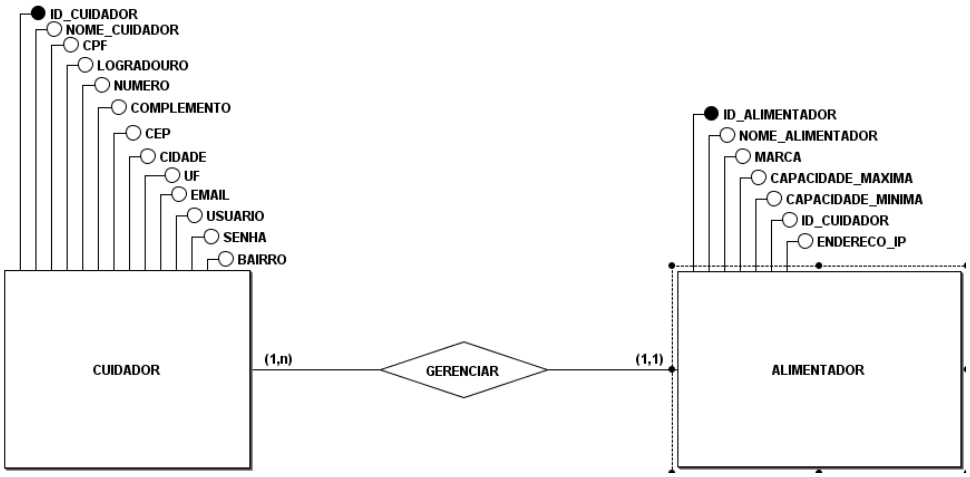


Figura 2. Modelo de Entidade Relacionamento - Server Feeder

A figura 3, apresenta o Diagrama de Entidade e Relacionamento (DER). O DER é uma representação do que foi escrito no MER, porém, de forma gráfica. O ID_CUIDADOR, é a chave primária (*Primary Key*), da entidade Cuidador, no banco de dados. Os atributos CPF, EMAIL e USUÁRIO, são informações únicas (*Unique*), no banco de dados.

Já na entidade Alimentador, o atributo ID_ALIMENTADOR, é a chave primária (*Primary Key*), da entidade no banco de dados. O atributo ID_CUIDADOR, é a chave estrangeira (*Foreign Key*) no banco de dados, sendo que essa chave estrangeira, é a chave primária da entidade Cuidador.

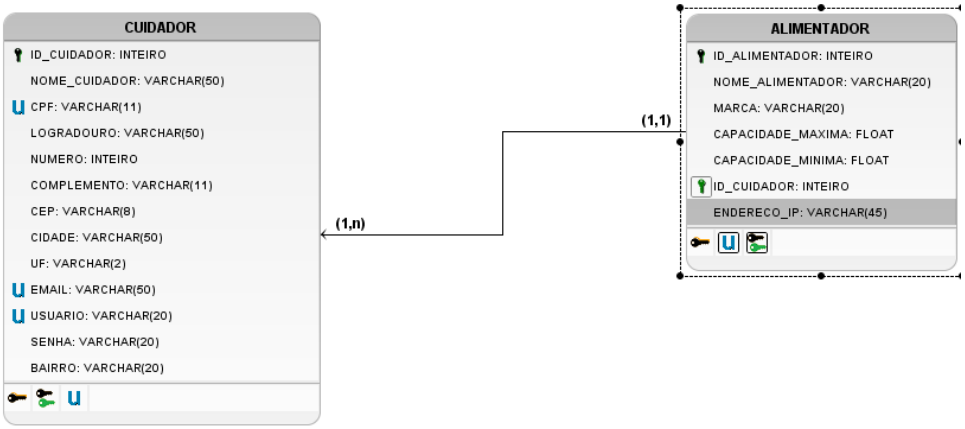


Figura 3. Diagrama de Entidade e Relacionamento - Server Feeder

5. Implementação da Proposta

Neste capítulo será apresentado algumas explicações do código fonte e telas do sistema.

Nas Figuras 4, 5, 6, é apresentado o código fonte da tela de cadastro de usuário. Especificamente foi aplicado a API do ViaCep, que realiza consulta na base de dados dos Correios. O Server Feeder, faz o consumo da API ViaCep, para que no momento em que o usuário esteja realizando o cadastro, ao inserir o CEP, automaticamente é preenchido os campos rua, bairro, cidade, uf, exigindo do usuário preencher apenas os outros campos.

```
<!-- Inicio do formulario -->
<!-- Formulário com classes do Bootstrap -->
<div class="container mt-5">
    <h1 class="mb-4">Cadastrar Usuário</h1>

    <script>
        function limpa_formulário_cep() {
            //Limpa valores do formulário de cep.
            document.getElementById('rua').value = ("");
            document.getElementById('bairro').value = ("");
            document.getElementById('cidade').value = ("");
            document.getElementById('uf').value = ("");
        }

        function meu_callback(conteudo) {
            if (!("erro" in conteudo)) {
                //Atualiza os campos com os valores.
                document.getElementById('rua').value = (conteudo.logradouro);
                document.getElementById('bairro').value = (conteudo.bairro);
                document.getElementById('cidade').value = (conteudo.localidade);
                document.getElementById('uf').value = (conteudo.uf);
            } //end if.
            else {
                //CEP não Encontrado.
                limpa_formulário_cep();
                alert("CEP não encontrado.");
            }
        }

        function pesquisacep(valor) {

            //Nova variável "cep" somente com dígitos.
            var cep = valor.replace(/\D/g, '');

            //Verifica se campo cep possui valor informado.
            if (cep != "") {
```

Figura 4. Código da tela de cadastro de usuário - Server Feeder

```
}

function pesquisacep(valor) {

    //Nova variável "cep" somente com dígitos.
    var cep = valor.replace(/\D/g, '');

    //Verifica se campo cep possui valor informado.
    if (cep != "") {

        //Expressão regular para validar o CEP.
        var validacep = /^[0-9]{8}$/;

        //Valida o formato do CEP.
        if (validacep.test(cep)) {

            //Preenche os campos com "..." enquanto consulta webservice.
            document.getElementById('rua').value = "...";
            document.getElementById('bairro').value = "...";
            document.getElementById('cidade').value = "...";
            document.getElementById('uf').value = "...";

            //Cria um elemento javascript.
            var script = document.createElement('script');

            //Sincroniza com o callback.
            script.src = 'https://viacep.com.br/ws/' + cep + '/json/?callback=meu_callback';

            //Insere script no documento e carrega o conteúdo.
            document.body.appendChild(script);

        } //end if.
        else {
            //cep é inválido.
```

Figura 5. Código da tela de cadastro de usuário - Server Feeder

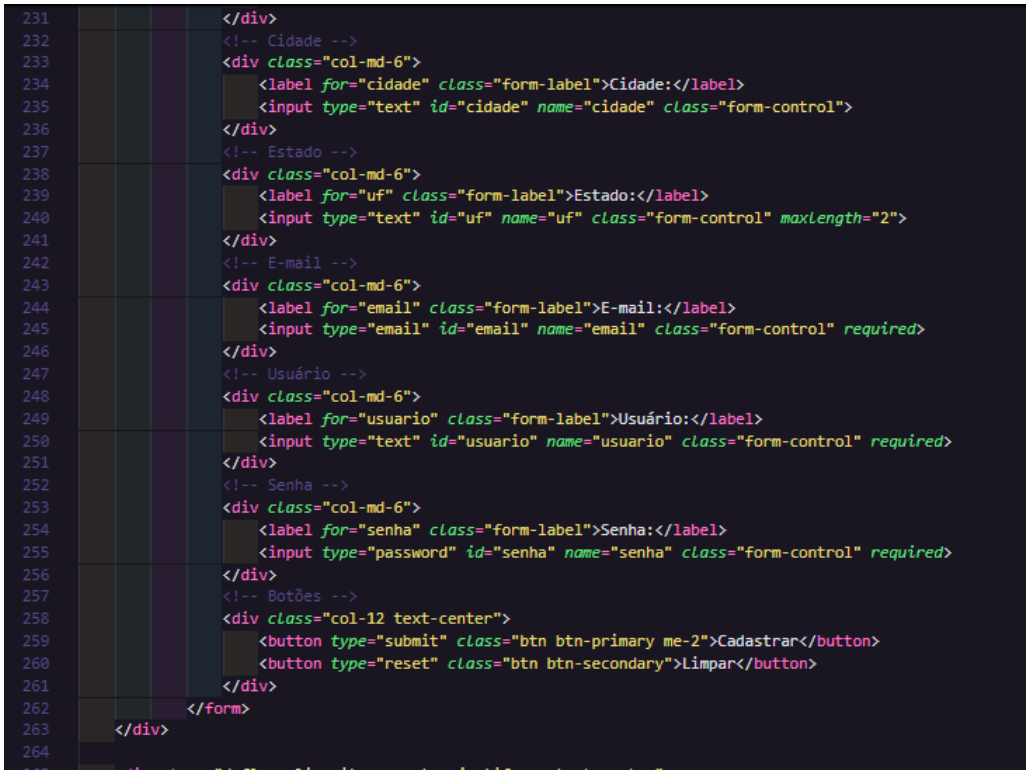
```
            //Cria um elemento javascript.
            var script = document.createElement('script');

            //Sincroniza com o callback.
            script.src = 'https://viacep.com.br/ws/' + cep + '/json/?callback=meu_callback';

            //Insere script no documento e carrega o conteúdo.
            document.body.appendChild(script);

        } //end if.
        else {
            //cep é inválido.
            limpa_formulário_cep();
            alert("Formato de CEP inválido.");
        }
    } //end if.
    else {
        //cep sem valor, limpa formulário.
        limpa_formulário_cep();
    }
};
</script>
```

Figura 6. Código da tela de cadastro de usuário - Server Feeder

Nas Figuras 7 e 8, apresenta o formulário de cadastro do usuário. Na 7, no início do formulário é usado o método POST, onde é acessada as funções proposta na página cadastrocuidador.php.

```
<form method="POST" action="cadastrocuidador_action.php" class="row g-3">
    <!-- Nome -->
    <div class="col-md-6">
        <label for="name" class="form-label">Nome:</label>
        <input type="text" id="name" name="nome_cuidador" class="form-control" required>
    </div>
    <!-- CPF -->
    <div class="col-md-6">
        <label for="cpf" class="form-label">CPF:</label>
        <input type="text" id="cpf" name="cpf" class="form-control" required>
    </div>
    <!-- CEP -->
    <div class="col-md-6">
        <label for="cep" class="form-label">CEP:</label>
        <input type="text" id="cep" name="cep" class="form-control" maxlength="9" onblur="pesquisacep(this.value);" required>
    </div>
    <!-- Rua -->
    <div class="col-md-6">
        <label for="rua" class="form-label">Rua:</label>
        <input type="text" id="rua" name="logradouro" class="form-control">
    </div>
    <!-- Número -->
    <div class="col-md-6">
        <label for="numero" class="form-label">Número:</label>
        <input type="text" id="numero" name="numero" class="form-control" required>
    </div>
    <!-- Complemento -->
    <div class="col-md-6">
        <label for="complemento" class="form-label">Complemento:</label>
        <input type="text" id="complemento" name="complemento" class="form-control">
    </div>
    <!-- Bairro -->
    <div class="col-md-6">
        <label for="bairro" class="form-label">Bairro:</label>
        <input type="text" id="bairro" name="bairro" class="form-control">
```

Figura 7. Código da tela de cadastro de usuário - Server Feeder



```
        </div>
        <!-- Cidade -->
        <div class="col-md-6">
            <label for="cidade" class="form-label">Cidade:</label>
            <input type="text" id="cidade" name="cidade" class="form-control">
        </div>
        <!-- Estado -->
        <div class="col-md-6">
            <label for="uf" class="form-label">Estado:</label>
            <input type="text" id="uf" name="uf" class="form-control" maxlength="2">
        </div>
        <!-- E-mail -->
        <div class="col-md-6">
            <label for="email" class="form-label">E-mail:</label>
            <input type="email" id="email" name="email" class="form-control" required>
        </div>
        <!-- Usuário -->
        <div class="col-md-6">
            <label for="usuario" class="form-label">Usuário:</label>
            <input type="text" id="usuario" name="usuario" class="form-control" required>
        </div>
        <!-- Senha -->
        <div class="col-md-6">
            <label for="senha" class="form-label">Senha:</label>
            <input type="password" id="senha" name="senha" class="form-control" required>
        </div>
        <!-- Botões -->
        <div class="col-12 text-center">
            <button type="submit" class="btn btn-primary me-2">Cadastrar</button>
            <button type="reset" class="btn btn-secondary">Limpar</button>
        </div>
    </form>
</div>
```

Figura 8. Código da tela de cadastro de usuário - Server Feeder

Na Figura 9, é apresentado o código fonte de validação do CPF do usuário, quando inserido no momento do cadastro. No comando realiza os seguintes processos:

- Usa a função validarCPF, para validar o número do CPF digitado no campo CPF do formulário de cadastro de usuário.

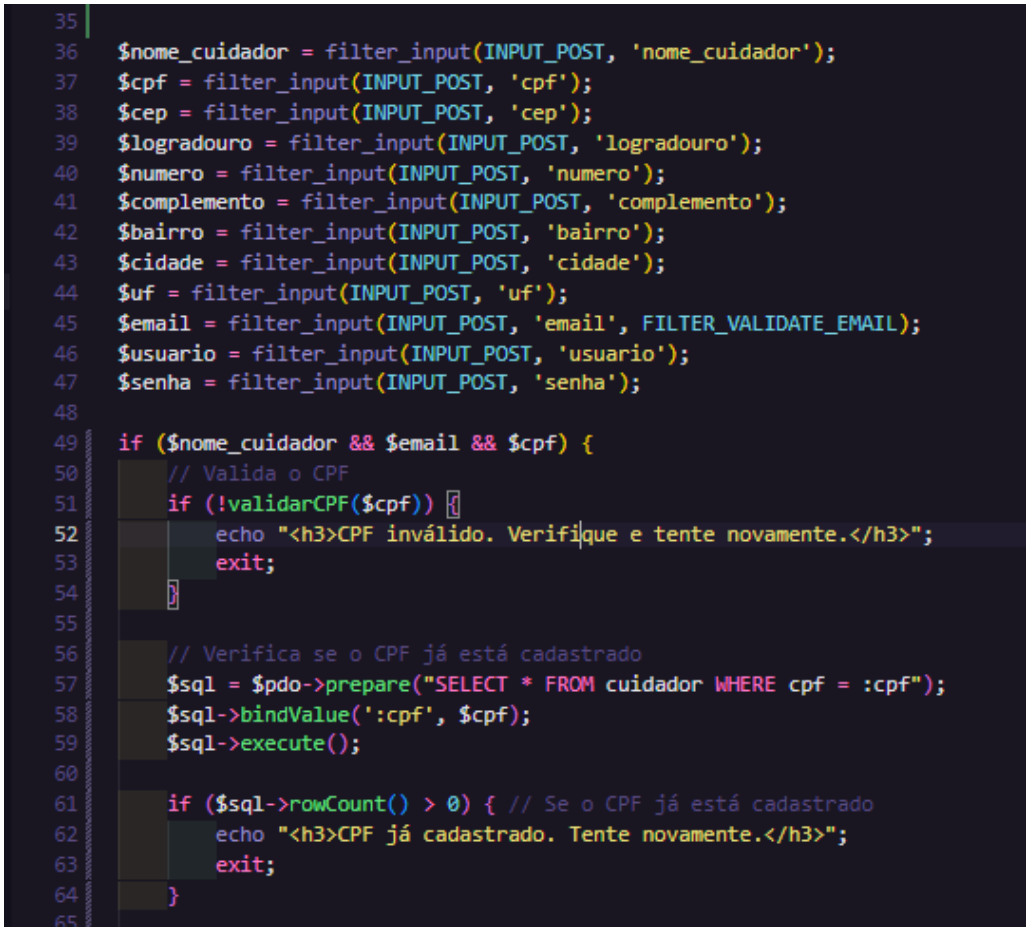
- Verifica se os caracteres são numéricos, caso seja não , são removidos.
- Faz a verificação e cálculos dos números do CPF, se possui 11 dígitos e se é válido.

cadastrocuidador_action.php > ...

```php
<?php

require 'config.php';

// Função para validar CPF
function validarCPF($cpf)
{
    // Remove caracteres não numéricos
    $cpf = preg_replace('/[^0-9]/', '', $cpf);

    // Verifica se o CPF tem 11 dígitos
    if (strlen($cpf) != 11) {
        return false;
    }

    // Elimina CPFs inválidos conhecidos
    if (preg_match('/(\d)\1{10}/', $cpf)) {
        return false;
    }

    // Calcula os dígitos verificadores para validar o CPF
    for ($t = 9; $t < 11; $t++) {
        $d = 0;
        for ($c = 0; $c < $t; $c++) {
            $d += $cpf[$c] * (($t + 1) - $c);
        }
        $d = ((10 * $d) % 11) % 10;
        if ($cpf[$c] != $d) {
            return false;
        }
    }

    return true;
}
```

Figura 9. Código da validação do CPF - Server Feeder

Na Figura 10, apresentam as variáveis e parâmetros que recebem e gravam as informações digitadas no formulário de cadastro do usuário. Também é realizada uma consulta no banco de dados para verificar se o CPF já foi cadastrado por um outro usuário, caso já esteja em uso por um outro usuário, não é permitido a utilização do mesmo, caso contrário o cadastro é realizado com sucesso, como apresentado na Figura 11.



```php

$nome_cuidador = filter_input(INPUT_POST, 'nome_cuidador');
$cpf = filter_input(INPUT_POST, 'cpf');
$cep = filter_input(INPUT_POST, 'cep');
$logradouro = filter_input(INPUT_POST, 'logradouro');
$numero = filter_input(INPUT_POST, 'numero');
$complemento = filter_input(INPUT_POST, 'complemento');
$bairro = filter_input(INPUT_POST, 'bairro');
$cidade = filter_input(INPUT_POST, 'cidade');
$uf = filter_input(INPUT_POST, 'uf');
$email = filter_input(INPUT_POST, 'email', FILTER_VALIDATE_EMAIL);
$usuario = filter_input(INPUT_POST, 'usuario');
$senha = filter_input(INPUT_POST, 'senha');

if ($nome_cuidador && $email && $cpf) {
    // Valida o CPF
    if (!validarCPF($cpf)) {
        echo "<h3>CPF inválido. Verifique e tente novamente.</h3>";
        exit;
    }

    // Verifica se o CPF já está cadastrado
    $sql = $pdo->prepare("SELECT * FROM cuidador WHERE cpf = :cpf");
    $sql->bindValue(':cpf', $cpf);
    $sql->execute();

    if ($sql->rowCount() > 0) { // Se o CPF já está cadastrado
        echo "<h3>CPF já cadastrado. Tente novamente.</h3>";
        exit;
    }
```

Figura 10. Código da validação do CPF - Server Feeder

Na Figura 11, vemos como a senha é armazenada em *hashing*, o que proporciona mais segurança para a senha do usuário.

```php
    // Verifica se o e-mail já está cadastrado
    $sql = $pdo->prepare("SELECT * FROM cuidador WHERE email = :email");
    $sql->bindValue(':email', $email);
    $sql->execute();

    if ($sql->rowCount() > 0) { // Se o e-mail já está cadastrado
        echo "<h3>E-mail já cadastrado</h3>";
        exit;
    }

    // Caso contrário, faz o cadastro
    // Criptografa a senha antes de salvar no banco de dados
    $senhaHash = password_hash($senha, PASSWORD_DEFAULT);

    $sql = $pdo->prepare("INSERT INTO cuidador (nome_cuidador, cpf, cep, logradouro, numero, complemento, bairro, cidade, uf, email, usuario, senha)
    VALUES(:nome_cuidador, :cpf, :cep, :logradouro, :numero, :complemento, :bairro, :cidade, :uf, :email, :usuario, :senha)");
    $sql->bindValue(':nome_cuidador', $nome_cuidador);
    $sql->bindValue(':cpf', $cpf);
    $sql->bindValue(':cep', $cep);
    $sql->bindValue(':logradouro', $logradouro);
    $sql->bindValue(':numero', $numero);
    $sql->bindValue(':complemento', $complemento);
    $sql->bindValue(':bairro', $bairro);
    $sql->bindValue(':cidade', $cidade);
    $sql->bindValue(':uf', $uf);
    $sql->bindValue(':email', $email);
    $sql->bindValue(':usuario', $usuario);
    $sql->bindValue(':senha', $senhaHash); // Senha armazenada em hashing

    $sql->execute();

?>
```

Figura 11. Armazenamento da senha em *hashing* - Server Feeder

Na Figura 12, apresenta a tela de cadastro de usuários, onde possui o formulário para inserção dos dados do usuário.

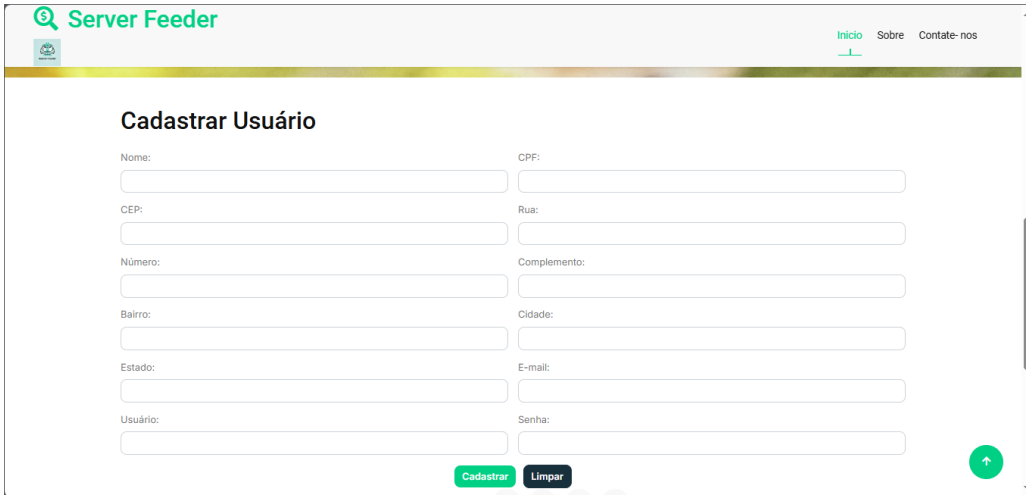


Figura 12. Tela de cadastro do usuário - Server Feeder

Na Figura 13, apresenta a tela de *login* do usuário. Após o usuário acessar o sistema, é direcionado para a lista de alimentadores cadastrados, e se caso ainda não tenha, na mesma tela há um link que o direciona para a tela de cadastro dos alimentadores, como mostrado nas Figuras 14 e 15.

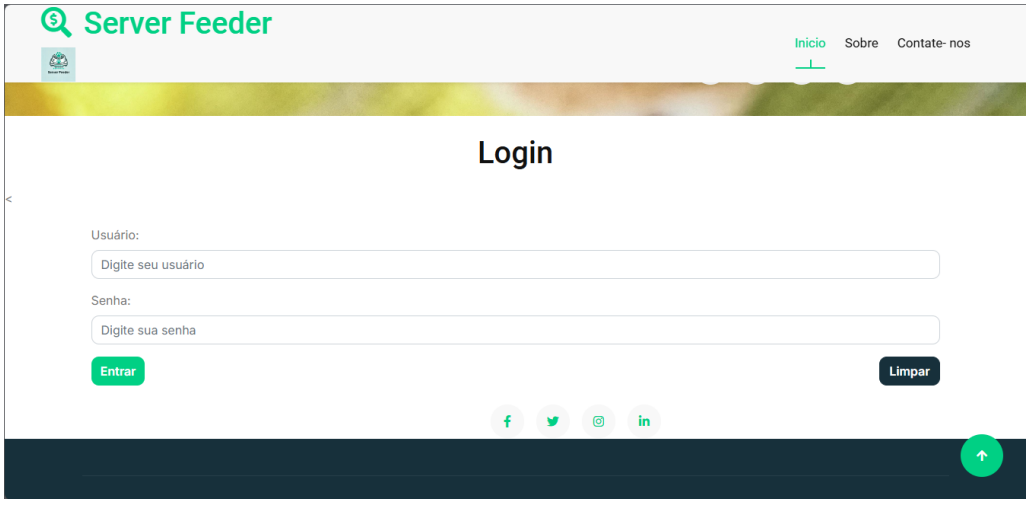


Figura 13. Tela *login* do usuário - Server Feeder

Figura 14. Lista de alimentadores cadastrados - Server Feeder

Figura 15. Tela alimentadores cadastrados - Server Feeder

Na Figura 16, apresenta o código fonte de validação de *login* (usuário e senha), que ocorre o seguinte processo:

- Captura os dados do usuário.
- Faz a limpeza dos dados que recebeu.
- Valida as informações, consultando o banco de dados.
- Verifica se a senha está correta, usando a função *password_verfy*, pois a senha é armazenada em *hashing* no banco de dados, como apresentado na Figura 13.
- Se os dados do usuário estiverem correto, é iniciada a sessão e armazenado o ID do usuário (chave primária no banco de dados).
- Redireciona o usuário para tela das listas de alimentadores cadastrados.
- Casos os dados do usuário estejam incorretos, ou o usuário não tenha preenchido todos os campos, ele será alertado pelo sistema.

Só é concedido acesso ao sistema para o usuário, após a conclusão do processo descrito acima.

validar_login.php > ...

```php
<?php

require 'config.php';

// Captura os dados do formulário
$usuario = filter_input(INPUT_POST, 'usuario', FILTER_DEFAULT);
$senha = filter_input(INPUT_POST, 'senha', FILTER_DEFAULT);

// Sanitização adicional para strings usando htmlspecialchars
$usuario = htmlspecialchars($usuario, ENT_QUOTES, 'UTF-8');
$senha = htmlspecialchars($senha, ENT_QUOTES, 'UTF-8');
if ($usuario && $senha) {
    // Busca o usuário e sua chave única (id_cuidador) no banco de dados
    $sql = $pdo->prepare("SELECT id_cuidador, usuario, senha FROM cuidador WHERE usuario = :usuario");
    $sql->bindValue(':usuario', $usuario);
    $sql->execute();

    if ($sql->rowCount() > 0) {
        $cuidador = $sql->fetch(PDO::FETCH_ASSOC);

        // Verifica a senha usando password_verify
        if (password_verify($senha, $cuidador['senha'])) {
            // Inicia a sessão e armazena o id_cuidador
            session_start();
            $_SESSION['id_cuidador'] = $cuidador['id_cuidador']; // Use a chave única

            // Redireciona o usuário após login bem-sucedido
            header("Location: lista_alimentador.php");
            exit;
        } else {
            echo "Senha incorreta. Tente novamente.";
        }
    } else {
        echo "Usuário não encontrado.";
    }
} else {
    echo "Por favor, preencha todos os campos.";
}

```

Figura 16. Código de validação de usuário - Server Feeder

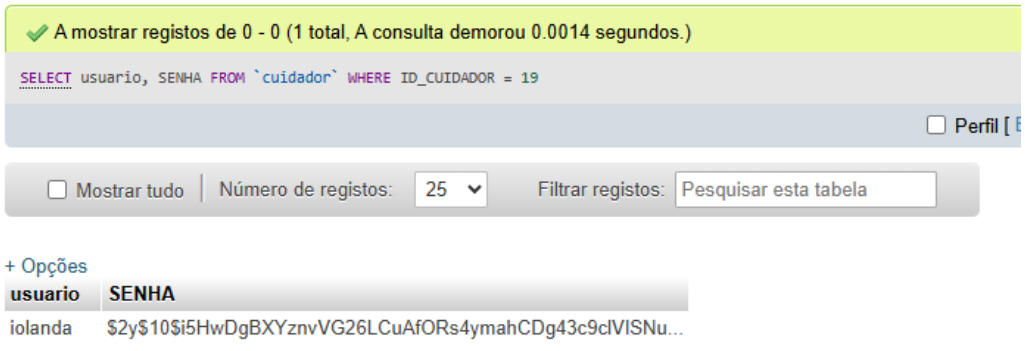


Figura 17. Senha do usuário armazenada em *hashing* - Server Feeder

Na Figura 17, é apresentado o código fonte da pagina logout.php, que permite ao usuário encerre seu acesso ao sistema.

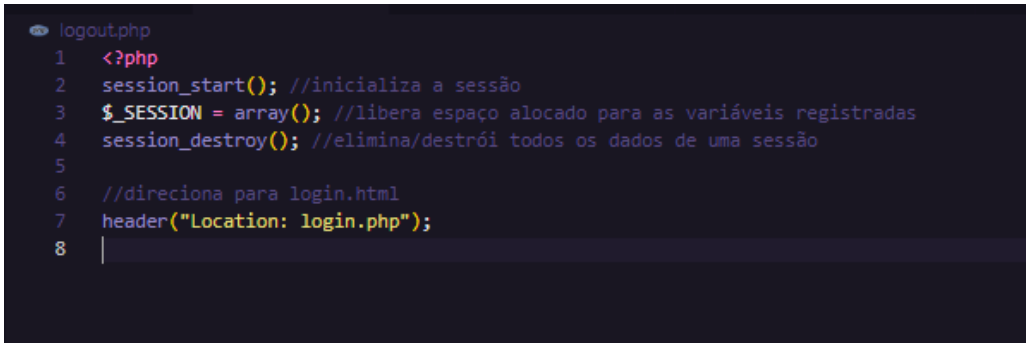


Figura 18. Código fonte da página de *logout* - Server Feeder

6. Conclusão

O desenvolvimento do projeto que visava a criação de um sistema Web, onde foi inclusivo o *front- end* e *Back-end*, para cadastro de gerenciamento de alimentadores automatizados para *pets* foi implementado conforme os requisitos inciais levantados. Os usuários precisam ser cadastrados no sistema, possibilitando- os cadastrar e gerenciar seus alimentadores para *pets*, sem o cadastro, as opções estarão inacessíveis.

O projeto foi testado em nível de desenvolvimento. Durante os testes foi analisada a performance do sistema, que inicialmente estava muito lento ao carregar as páginas. Foi realizada a junção de algumas páginas, para que o sistema se tornasse mais ágil. Após os ajustes, percebemos uma grande melhora na performance.

Durante o processo de desenvolvimento do Sistema *Web*, houve desafios na inserção de informações no banco de dados. O sistema não estava efetuando as conversões das informações inseridas nos campos, por exemplo, *string* para *interger*.

Para o desenvolvimento do projeto foram aplicados os conhecimentos adquiridos nas disciplinas de Lógica de Programação, Banco de Dados, Desenvolvimento *Web*, Engenharia de *Software* e Interação Humano Computador.

Como sugestão de trabalhos futuros, é necessário implementar um exemplo de Módulo Alimentador, bem como o protocolo que conecta o sistema web ao mesmo, e testar ativamente a conexão entre os dois. Novos testes no sistema web também precisam ser realizados.

Os protocolos que serão implementados no sistema, são o TCP/IP, sendo TCP *(Transmission Control Protocol)* e IP *(Internet Protocol)*, em português Protocolo de Controle de Transmissão e Protocolo de Internet, respectivamente.

A escolha desses protocolos, como mostrado na 14, foram feitas devido o TCP/IP serem um conjuntos de protocolos de comunicação com recursos apropriado para serem utilizados nesse tipo de serviço como o do Server Feeder, pois permite a interconexão de servidores e computadores em uma determinada rede.

Referências

ABINPET (2023). Informações gerais do setor. *https://abinpet.org.br/informacoes-gerais-do-setor/*.

BATISTA, D. L. M., FRANÇA, G. S., PESSOA, G. A. P. C. A., BEZERRA, J. P. S. M., JUNIOR, J. R. G. S., and TAVARES, S. A. C. (2015). Alimentador automático para cães. *http://sistemaolimpo.org/midias/uploads/c2c424637ae10c01d4eb153d020a4e70.pdf, Acesso em: Dezembro 2023*.

DANTAS, L. G. (2022). Sistema supervisório parar alimentador automático para cachorros e gatos em situação de rua na ufersa. *https://repositorio.ufersa.edu.br/server/api/core/bitstreams/862731d1-a70c-4e71-9db0-696525ea7b9b/content, Acesso em: Dezembro 2023*.

FLANAGAN, D. (2013). Javascript: O guia definitivo. *https://www.google.com.br/books/edition/JavaScript/zWNyDgAAQBAJ?hl=pt-BR'I&'gbpv=1'I&'dq=javascript+portugu*

GUEDES, G. T. A. (2018). Uml 2 - uma abordagem prática. https://www.google.com.br/books/edition/UML_2_Uma_Abordagem_Prática/mJxMDwAAQBAJ?hl=pt-BR&gbpv=1&dq=diagrama+de+caso+de+uso&printsec=frontcover, Acesso em Setembro 2024.

lan SOMMERVILLE (2018). Engenharia de software 10ª edição. https://plataforma.bvirtual.com.br/Leitor/Publicacao/168127/pdf/0?code=3BmgOhAgI4lQjSZWzSFKgu Acesso em : Setembro 2024.

MAUJOR, M. S. S. (2018). Bootstrap: Um guia completo para construir aplicativos responsivos, modernos e eficientes. https://www.google.com.br/books/edition/Bootstrap/W-rIEAAAQBAJ?hl=pt-BR&gbpv=1&dq=bootstrap&printsec=frontcover, Acesso em Outbro de 2024.

MySQL (2012). 10 principais motivos para usar o mysql como um banco de dados incorporado. https://www.mysql.com/why-mysql/white-papers/10-principais-motivos-para-usar-o-mysql-como-um-banco-de-dados-incorporado/, Acesso em Outubro de 2024.

NIEDERAUER, J. (2017). Desenvolvendo websites com php: Aprenda a criar websites dinâmicos e interativos com php e banco de dados. https://www.google.com.br/books/edition/Desenvolvendo_Websites_com_PHP/ODM5DwAAQBAJ?hl=pBR'I&'gbpv=1'I&'dq=php+2023+portugu

RAMOS, T. P. (2023). Aplicação do conceito de internet das coisas no desenvolvimento de um alimentador para animais domésticos. *http://repositorio.unesc.net/bitstream/1/10356/1/Thiago*

SABBAGH, R. (2022). Scrum: Gestão ágil para produtos de suscesso. https://www.google.com.br/books/edition/Scrum/VVhcEAAAQBAJ?hl=pt-BR'I&'gbpv=1'I&'dq=scrum+2023'I&'printsec=frontcover, Acesso em: Dezembro 2023.

SILVA, M. S. (2018). Fundamentos de html5 e css3. https://www.google.com.br/books/edition/Fundamentos_de_HTML5_e_CSS3/ZyJyDwAAQBAJ?hl=pt-BR'I&'gbpv=1'I&'dq=HTML+2023+português'I&'printsec=frontcover.

SOMMERVILLE, I. (2011). Engenharia de software 10ª edição. https://plataforma.bvirtual.com.br/Leitor/Publicacao/2613/epub/0?code=BfNn+n5OaaxOD0pZ1ZHGxg Acesso em: Setembro 2024.

SUTHERLAND, K. S. J. (2020). O guia do scrum: O guia definitivo para o scrum: As regras do jogo. https://scrumguides.org/docs/scrumguide/v2020/2020-Scrum-Guide-Portuguese-European.pdf, Acesso em: Dezembro 2023.

Documento Digitalizado Restrito

Artigo Versão Final

Assunto: Artigo Versão Final
Assinado por: Rodolfo Oliveira
Tipo do Documento: Anexo
Situação: Finalizado
Nível de Acesso: Restrito
Hipótese Legal: Direito Autoral - conservar a obra inédita (Art. 24, III, da Lei nº 9.610/1998)
Tipo do Conferência: Documento Digital

Documento assinado eletronicamente por:

- **Rodolfo Francisco de Oliveira**, **PROFESSOR ENS BASICO TECN TECNOLOGICO**, em 17/03/2025 15:13:22.

Este documento foi armazenado no SUAP em 17/03/2025. Para comprovar sua integridade, faça a leitura do QRCode ao lado ou acesse https://suap.ifsp.edu.br/verificar-documento-externo/ e forneça os dados abaixo:

Código Verificador: 1969040
Código de Autenticação: f68d935a23