SISTEMA FIEP UNIDADE CENTRO

TÉC. EM ANÁLISE E DESENVOLVIMENTO DE SISTEMAS

VICTOR LUIS GOEDICKE

**PROTÓTIPO LAR BASTET**

Ong de animais

CURITIBA-PR

2023

VICTOR LUIS GOEDICKE

**PROTÓTIPO LAR BASTET**

Ong de animais

Trabalho de conclusão apresentado ao curso de técnico de análise e desenvolvimento de sistemas do sistema Fiep unidade centro, como exigência parcial para obtenção de aprovação sob orientação do Prof. Gustavo Vinícius Linhares Caes.

CURITIBA-PR

2023

**LISTA DE SIGLAS**

PHP: Hypertext Preprocessor;

JS: JavaScript;

HTML: HyperText Markup Language;

CSS: Cascading Style Sheets;

CRUD: Create, Read, Update, Delete;

OOP: Object oriented programming;

SQL: Structured Query Language;

XAMPP: Pacote com os principais servidores de código aberto do mercado, incluindo FTP, banco de dados MySQL e Apache com suporte as linguagens PHP e Perl.

**Sumário**

[**1. INTRODUCAO** 8](#_Toc137851334)

[1.1 Motivação 8](#_Toc137851335)

[1.2 Objetivo e metas 8](#_Toc137851337)

[**2. ANALISE DE REQUISITOS** 9](#_Toc137851338)

[2.1 Requisitos funcionais nos métodos: 10](#_Toc137851339)

[Dos usuários: 10](#_Toc137851340)

[2.1.1 Tela Cadastro 10](#_Toc137851341)

[2.1.2. Tela de Login 18](#_Toc137851342)

[2.1.3. Feed de fotos 19](#_Toc137851343)

[2.1.4. Redefinir senha 20](#_Toc137851344)

[2.1.5 – Unicidade da empresa 21](#_Toc137851345)

[2.1.6 – Cadastrar parceiro 22](#_Toc137851346)

[2.1.7 – Cadastro de animal para adoção 23](#_Toc137851347)

[2.2 Requisitos não funcionais 35](#_Toc137851348)

[**3.** **MATERIAIS E MÉTODOS** 36](#_Toc137851349)

[3.1 Materiais 36](#_Toc137851350)

[3.2 Métodos 39](#_Toc137851351)

[3.2.1 Métodologia de desenvolvimento 39](#_Toc137851352)

[**4.** **MODELAGEM DO SISTEMA E CASOS DE USO** 39](#_Toc137851353)

[4.2 Casos de uso 40](#_Toc137851354)

[UC01 - Fazer Login 40](#_Toc137851355)

[UC02´- Cadastrar usuario 41](#_Toc137851356)

[UC03 – Adoção de animais 43](#_Toc137851357)

[4.2.1 – Observação 43](#_Toc137851358)

[**5.** **TELAS DO PROJETO** 44](#_Toc137851359)

[5.6 Tela do sistema 46](#_Toc137851360)

[5.7 Tela de parceria 46](#_Toc137851361)

[5.8 Tela colocar para adotar 47](#_Toc137851362)

[5.11 Listagem para o crud 48](#_Toc137851364)

[**6.** **CONCLUSÃO** 50](#_Toc137851365)

**AGRADECIMENTOS**

Primeiramente agradeço minha namorada que sempre me apoiou no meu crescimento escolar e pessoal.

Aos professores que com suas competências contribuíram e muito no decorrer do desenvolvimento deste trabalho.

Agradeço especialmente ao orientador Gustavo Vinícius Linhares Caes pelas ideias e tempo disposto para avaliação e evolução no decorrer da confecção deste trabalho

Resumo: O protótipo visa ajudar ongs de animais, tendo como objetivo facilitar o cadastro de informações tanto de adotantes quanto o de animais resgatados. Foi desenvolvido em OOP (Programação Orientada a Objetos) com tecnologias como PHP, JavaScript, HTML e CSS, além de CRUD administrativo. O nome Lar Bastet é oriundo da mitologia egípcia, também conhecida como Bast ou Ubasti, é uma deusa da mitologia egípcia. Ela era adorada como uma divindade felina associada à proteção, fertilidade, maternidade e alegria. Bastet era frequentemente representada como uma leoa ou como uma mulher com cabeça de leoa. Ela era considerada a deusa do lar, da família e protetora dos faraós. Nos dias atuais, Bastet continua sendo uma figura conhecida e é frequentemente retratada como uma deusa felina nas representações artísticas e culturais do Antigo Egito. Ela também é uma figura popular na cultura popular, muitas vezes sendo associada à imagem de gatos e à proteção de animais domésticos.

Abstract: The prototype aims to help animal NGOs, with the objective of facilitating the registration of information from both adopters and rescued animals. It was developed in OOP (Object Oriented Programming) with technologies such as PHP, JavaScript, HTML and CSS, as well as administrative CRUD. The name Lar Bastet comes from Egyptian mythology, also known as Bast or Ubasti, is a goddess of Egyptian mythology. She was worshiped as a feline deity associated with protection, fertility, motherhood and joy. Bastet was often represented as a lioness or as a woman with a lioness's head. She was considered the goddess of the home, family and protector of the pharaohs. In the present day, Bastet remains a well-known figure and is often portrayed as a feline goddess in ancient Egyptian artistic and cultural depictions. She is also a popular figure in popular culture, often associated with the image of cats and the protection of domestic animals.

# INTRODUçãO

## **1.1 Motivação**

## Segundo dados da Organização Mundial da Saúde, estima-se que, no Brasil, há 30 milhões de animais domésticos abandonados, entre eles 20 milhões de cães e 10 milhões de gatos. O número é superlativo porque a adoção de bichos não é aplicada em grande escala por brasileiros. Ainda prevalece a compra de animais.

## **1.2 Objetivo e metas**

Tem como objetivo principal, incentivar a comunidade em um geral, a prática mais o ato de adotar, assim como já citado, existem milhões de animais em situação de rua e incentivar a comunidade, daria aos animais a chance de possuir uma nova vida e uma família que os trate com o devido respeito e amor.

A meta foi desenvolver um software intuitivo com uma interface amigável onde torna o processo de adoção mais simples aos usuários , buscamos por parceiros, não apenas com empresas, mas com outras ongs e a comunidade em geral.

# ANÁLISE de requisitos

A análise de requisitos é uma etapa crucial no desenvolvimento de qualquer projeto, incluindo projetos na web. Essa fase envolve a identificação, compreensão e documentação das necessidades e expectativas dos usuários, bem como dos objetivos e restrições do sistema a ser desenvolvido.

A análise de requisitos visa estabelecer uma base sólida para a construção do sistema, garantindo que as funcionalidades, características e comportamentos esperados sejam claramente definidos e compreendidos por todos os envolvidos no projeto.

Os **requisitos funcionais** são declarações que descrevem as funcionalidades, comportamentos e ações específicas que um sistema ou software deve ser capaz de executar. Eles representam as principais características e capacidades do sistema, definindo o que ele deve fazer para atender às necessidades dos usuários.

Esses requisitos descrevem os diferentes recursos e funcionalidades que o sistema deve oferecer, fornecendo uma visão clara das ações que os usuários podem realizar e dos resultados esperados. Eles são geralmente expressos na forma de casos de uso, histórias de usuário ou especificações detalhadas.

Os **requisitos não funcionais** são critérios ou características que definem a qualidade, desempenho, segurança e usabilidade de um sistema ou software, além de especificar as restrições e limitações que devem ser atendidas. Eles não estão diretamente relacionados às funcionalidades específicas do sistema, mas sim às suas qualidades gerais.

Ao contrário dos requisitos funcionais, que descrevem o que o sistema deve fazer, os requisitos não funcionais se concentram em como o sistema deve ser, definindo os atributos que o sistema precisa ter para ser considerado adequado e eficiente. Eles podem abranger diversas áreas e aspectos do sistema, como desempenho, segurança, confiabilidade, usabilidade, escalabilidade, compatibilidade, entre outros.

## **Requisitos funcionais nos métodos:**

**Dos usuários:**

Serão apresentados os requisitos funcionais presentes nos métodos do arquivo “UsuarioDB.php” .

## **Tela Cadastro**

**RF01 - Cadastro de usuários**

|  |
| --- |
| 1. O sistema deve permitir o cadastro de novos usuários. |
| 2. O usuário deve fornecer as seguintes informações para realizar o cadastro:  a. Nome (obrigatório),b. Sobrenome (obrigatório),c. Senha (obrigatório),  d. Confirmação de senha (obrigatório),e. CPF (obrigatório),f. Email (obrigatório) |
| 3. O sistema deve filtrar e validar as informações fornecidas pelo usuário. |
| 4. Caso algum campo obrigatório esteja vazio, o sistema deve exibir uma mensagem de alerta informando que todas as informações devem ser fornecidas. |
| 5. O sistema deve verificar se o CPF possui exatamente 11 dígitos. Caso contrário, deve exibir uma mensagem de alerta informando que o CPF deve ter 11 dígitos. |
| 6. O sistema deve verificar se a senha possui pelo menos 6 dígitos. Caso contrário, deve exibir uma mensagem de alerta informando que a senha deve ter pelo menos 6 dígitos. |
| 7. Após a validação das informações, o sistema deve inserir os dados na tabela "cadastro" do banco de dados. |
| 8. O sistema deve retornar verdadeiro se o cadastro for realizado com sucesso e falso caso contrário. |

**RF02 - Verificação de campos não respondidos pelo usuário**

**RF02.1 - Validação de campos obrigatórios:**

|  |
| --- |
| 1. O sistema deve realizar a validação em tempo real dos campos obrigatórios durante o preenchimento do formulário de cadastro. |
| 2. O sistema deve verificar se todos os campos obrigatórios foram respondidos pelo usuário antes do envio do formulário. |
| 3. Caso algum campo obrigatório não tenha sido respondido, o sistema deve exibir uma mensagem de erro específica para cada campo em falta. |
| 4. As mensagens de erro devem ser claras e indicar qual campo precisa ser preenchido pelo usuário. |
| 5. A validação dos campos obrigatórios deve ocorrer à medida que o usuário preenche o formulário, proporcionando feedback imediato. |
| 6. O sistema deve permitir a personalização dos campos obrigatórios e suas respectivas validações de acordo com as regras de negócio do sistema e as necessidades do formulário de cadastro. |

**RF02.2 - Exibição de mensagens de erro:**

|  |
| --- |
| 1. O sistema deve exibir mensagens de erro adequadas para cada campo obrigatório não respondido pelo usuário. |
| 2. As mensagens de erro devem ser apresentadas de forma clara e legível ao usuário. |
| 3. Cada mensagem de erro deve indicar o campo específico que precisa ser preenchido. |
| 4. A exibição das mensagens de erro deve ser realizada em tempo real, à medida que o usuário preenche o formulário. |
| 5. O sistema deve garantir que as mensagens de erro sejam visíveis e não causem confusão ao usuário. |
| 6. As mensagens de erro devem ser exibidas em localizações próximas aos campos correspondentes para facilitar a identificação dos campos em falta. |

**RF02.3 - Impedimento de envio de formulário com campos obrigatórios não preenchidos:**

|  |
| --- |
| 1. O sistema deve impedir o envio do formulário de cadastro caso existam campos obrigatórios não respondidos pelo usuário. |
| 2. Ao tentar enviar o formulário com campos em falta, o sistema deve exibir uma mensagem de erro geral, indicando que todos os campos obrigatórios devem ser preenchidos. |
| 3. O sistema não deve permitir que o formulário seja submetido até que todos os campos obrigatórios sejam preenchidos corretamente. |
| 4. O usuário só deve ser capaz de enviar o formulário após a correção de todos os campos em falta. |

RF02.4 - Feedback de conclusão do preenchimento correto:

|  |
| --- |
| 1. Após o preenchimento correto de todos os campos obrigatórios, o sistema deve exibir uma confirmação ao usuário informando que o cadastro pode ser prosseguido. |
| 2. O feedback de conclusão do preenchimento correto deve ser claro e visível ao usuário. |
| 3. O sistema deve indicar visualmente que o cadastro pode ser finalizado sem problemas quando todos os campos obrigatórios estiverem preenchidos corretamente. |
| 4. O usuário deve receber uma mensagem de confirmação ao finalizar o preenchimento correto do formulário. |

**RF03 - Validação de senhas iguais**

RF03.1 - Validação de senhas iguais:

|  |
| --- |
| 1. O sistema deve realizar a validação em tempo real das senhas digitadas nos campos de "senha" e "confirmação de senha" durante o preenchimento do formulário de cadastro. |
| 2. O sistema deve verificar se as senhas digitadas nos campos correspondentes são iguais. |
| 3. Caso as senhas não sejam iguais, o sistema deve exibir uma mensagem de erro indicando a inconsistência das senhas. |
| 4. A mensagem de erro deve alertar o usuário sobre a divergência das senhas e orientá-lo a digitar novamente as senhas para garantir a correspondência. |
| 5. A validação de senhas iguais deve ocorrer imediatamente após o usuário digitar as senhas nos campos correspondentes. |
| 6. O sistema deve proporcionar feedback imediato ao usuário caso as senhas digitadas não sejam iguais. |
| 7. O sistema não deve permitir o envio do formulário de cadastro caso as senhas não sejam iguais, obrigando o usuário a corrigir a divergência antes de prosseguir. |

RF03.2 - Exibição de mensagem de erro:

|  |
| --- |
| 1. A mensagem de erro indicando que as senhas fornecidas não correspondem deve ser clara e legível ao usuário. |
| 2. A mensagem de erro deve ser exibida próximo aos campos de "senha" e "confirmação de senha" para facilitar a identificação da divergência. |
| 3. A exibição da mensagem de erro deve ser realizada em tempo real, assim que o usuário digitar as senhas. |
| 4. O sistema deve garantir que a mensagem de erro seja visível e não cause confusão ao usuário. |
| 5. A mensagem de erro deve orientar o usuário a corrigir as senhas digitadas para garantir a correspondência. |

RF03.3 - Impedimento de envio de formulário com senhas não correspondentes:

|  |
| --- |
| 1. O sistema não deve permitir o envio do formulário de cadastro caso as senhas fornecidas pelo usuário não correspondam. |
| 2. Ao tentar enviar o formulário com senhas não correspondentes, o sistema deve exibir uma mensagem de erro geral, indicando que as senhas devem ser iguais. |
| 3. O sistema não deve permitir que o formulário seja submetido até que as senhas correspondam corretamente. |
| 4. O usuário só deve ser capaz de enviar o formulário após corrigir a divergência nas senhas. |

RF03.4 - Feedback de conclusão do preenchimento correto das senhas:

|  |
| --- |
| 1. Após o preenchimento correto das senhas nos campos de "senha" e "confirmação de senha", o sistema deve exibir uma confirmação ao usuário informando que as senhas correspondem. |
| 2. O feedback de conclusão do preenchimento correto das senhas deve ser claro e visível ao usuário. |
| 3. O sistema deve indicar visualmente que as senhas correspondem quando ambas forem digitadas corretamente. |
| 4. O usuário deve receber uma mensagem de confirmação ao finalizar o preenchimento correto das senhas. |

**RF04 - Verificação de Usuário Existente**

RF04.1 - Verificação de CPF e e-mail duplicados:

|  |
| --- |
| 1. Durante o processo de cadastro, o sistema deve realizar consultas ao banco de dados para verificar se o CPF e o e-mail fornecidos pelo usuário já estão registrados na tabela "cadastro". |
| 2. O sistema deve comparar o CPF e o e-mail fornecidos pelo usuário com os registros existentes na tabela "cadastro" para identificar possíveis duplicidades. |
| 3. Caso seja identificada uma correspondência entre o CPF ou e-mail fornecidos e os registros existentes, o sistema deve exibir uma mensagem de erro informando que o usuário já está cadastrado com as informações fornecidas. |
| 4. A mensagem de erro deve ser clara e indicar especificamente se a duplicidade está relacionada ao CPF, ao e-mail ou a ambos. |
| 5. A verificação de CPF e e-mail duplicados deve ocorrer antes do registro do novo usuário no banco de dados. |
| 6. O sistema deve proporcionar feedback imediato ao usuário caso seja identificada uma duplicidade de CPF ou e-mail. |
| 7. A verificação de duplicidades deve abranger tanto novos cadastros quanto atualizações de informações de usuários existentes. |
| 8. O sistema deve garantir que a verificação de CPF e e-mail duplicados seja eficiente e não cause impactos significativos no desempenho do sistema. |

RF04.2 - Consideração da unicidade dos usuários:

|  |
| --- |
| 1. O sistema deve considerar a unicidade dos usuários com base nos campos de CPF e e-mail. |
| 2. O sistema não deve permitir o cadastro de um novo usuário caso o CPF ou o e-mail fornecidos já estejam registrados na tabela "cadastro". |
| 3. A verificação de duplicidade de CPF e e-mail deve ser realizada separadamente, permitindo que um usuário possa ter um CPF já registrado, mas um e-mail diferente, ou vice-versa. |
| 4. O sistema deve garantir que cada usuário seja identificado de forma única por seu CPF e/ou e-mail na tabela "cadastro". |
| 5. A unicidade dos usuários deve ser considerada em todas as operações relacionadas ao cadastro, incluindo a criação, atualização e exclusão de registros na tabela "cadastro". |
| 6. Caso a verificação de CPF e e-mail duplicados seja realizada durante uma atualização de informações de um usuário existente, o sistema deve permitir a atualização caso o CPF ou o e-mail fornecidos já estejam associados ao próprio usuário em edição. |
| 7. O sistema deve registrar no banco de dados apenas novos usuários que não apresentem duplicidade de CPF ou e-mail com registros existentes na tabela "cadastro". |

.

## **2.1.2. Tela de Login**

**Requisito Funcional:**

RF05 - Verificação de Credenciais de Login

|  |
| --- |
| 1. Durante o processo de login, o sistema deve comparar o CPF e a senha fornecidos pelo usuário com os registros armazenados na tabela "cadastro". |
| 2. A comparação das credenciais deve ser realizada de forma precisa, considerando apenas os caracteres numéricos do CPF e sem considerar caracteres especiais, espaços em branco ou diferenças de maiúsculas e minúsculas na senha. |
| 3. O sistema deve realizar a busca na tabela "cadastro" com base no CPF fornecido e verificar se existe um registro correspondente. |
| 4. Caso um registro seja encontrado com o CPF fornecido, o sistema deve comparar a senha fornecida com a senha armazenada no registro correspondente. |
| 5. Se a senha fornecida pelo usuário coincidir exatamente com a senha armazenada, o sistema considerará as credenciais válidas e permitirá o acesso ao usuário. |
| 6. Caso as credenciais não coincidam com as informações armazenadas, o sistema deve considerá-las inválidas e negar o acesso ao usuário. |
| 7. O sistema deve garantir a segurança das informações ao realizar a verificação das credenciais de login. |
| 8. Em caso de credenciais inválidas, o sistema deve exibir uma mensagem de erro adequada para informar o usuário sobre a falha no login. |
| 9. O sistema deve registrar logs de tentativas de login, incluindo as informações de CPF, data e hora da tentativa de login, para fins de segurança e auditoria. |
| 10. O sistema deve garantir a eficiência na verificação das credenciais de login, evitando impactos significativos no desempenho do sistema. |

## **2.1.3. Feed de fotos**

RF01 - Formulário de Adoção:

|  |
| --- |
| 1. O sistema deve exibir uma galeria de fotos e descrições dos animais disponíveis para adoção. |
| 2. O usuário deve ter a opção de selecionar um animal específico de interesse. |
| 3. Ao selecionar um animal, o usuário deve clicar em um botão para ser redirecionado para o formulário de adoção. |
| 4. O formulário de adoção deve solicitar as seguintes informações obrigatórias do usuário: nome, sobrenome, email e nome do animal que deseja adotar. |
| 5. O sistema deve realizar a validação em tempo real durante o preenchimento do formulário para verificar se todos os campos obrigatórios foram respondidos. |
| 6. Se algum campo obrigatório não for respondido, o sistema deve exibir uma mensagem de erro específica para cada campo em falta, indicando qual campo precisa ser preenchido. |
| 7. A mensagem de erro deve ser clara e informar ao usuário sobre os campos que precisam ser preenchidos corretamente. |
| 8. A validação de campos não respondidos deve ocorrer em tempo real, à medida que o usuário preenche o formulário, para fornecer feedback imediato. |
| 9. O sistema deve garantir que as informações preenchidas no formulário de adoção sejam enviadas corretamente para o processamento posterior. |
| 10. Após o preenchimento correto de todos os campos obrigatórios, o usuário deve ter a opção de enviar o formulário de adoção para iniciar o processo de adoção. |
| 11. O sistema deve registrar os dados do formulário de adoção para futuras referências e processamento. |
| 12. O sistema deve garantir a segurança e proteção das informações pessoais fornecidas pelo usuário durante o preenchimento do formulário de adoção. |

2.1.4. Redefinir senha

RF01 - Redefinição de Senha do Usuário:

|  |
| --- |
| 1. O sistema deve permitir que o usuário redefina sua senha, fornecendo uma nova senha desejada e o CPF correspondente. |
| 2. Ao iniciar o processo de redefinição de senha, o sistema verifica se a nova senha e o CPF foram fornecidos. Se algum dos campos estiver vazio, uma mensagem de alerta deve ser exibida, informando ao usuário que todas as informações devem ser preenchidas. O usuário é redirecionado para a página de redefinição de senha para corrigir o preenchimento. |
| 3. O sistema verifica se a nova senha possui pelo menos 6 caracteres. Se a senha tiver menos de 6 caracteres, uma mensagem de alerta deve ser exibida, informando ao usuário que a senha deve ter no mínimo 6 caracteres. |
| 4. Caso todas as informações estejam corretas, o sistema realiza uma atualização na tabela "cadastro" do banco de dados, substituindo a senha atual pela nova senha fornecida, associada ao CPF correspondente. |
| 5. O sistema verifica se a atualização afetou alguma linha no banco de dados. Se a atualização for bem-sucedida, retornando um número maior que zero de linhas afetadas, o sistema considera a redefinição de senha como concluída com sucesso. |
| 6. Caso a atualização não afete nenhuma linha no banco de dados, a redefinição de senha é considerada não concluída. |
| 7. Após a redefinição de senha concluída com sucesso, o usuário deve receber uma confirmação de que a senha foi alterada com sucesso. |
| 8. O sistema deve garantir a segurança e proteção das informações pessoais fornecidas pelo usuário durante o processo de redefinição de senha. |

**Dos parceiros**

## 2.1.5 – Unicidade da empresa

|  |
| --- |
| 1. O sistema deve verificar a unicidade da empresa durante o processo de cadastro. |
| 2. Durante o cadastro, o sistema realiza uma consulta à tabela "empresarial" no banco de dados para verificar se já existe um registro com o mesmo nome da empresa fornecido. |
| 3. A verificação de unicidade deve ser realizada de forma precisa, sem considerar caracteres especiais, espaços em branco ou diferenças de maiúsculas e minúsculas. |
| 4. Se já existir um registro com o mesmo nome da empresa no banco de dados, o sistema considera a empresa como não única e impede o cadastro. |
| 5. Caso não exista nenhum registro com o mesmo nome da empresa, o sistema permite o cadastro da empresa. |
| 6. O sistema deve fornecer um feedback claro ao usuário, informando-o sobre o resultado da verificação de unicidade e se o cadastro foi permitido ou impedido. |
| 7. A verificação de unicidade deve ocorrer em tempo real, à medida que o usuário preenche os campos de cadastro da empresa. |
| 8. O sistema deve garantir a integridade dos dados e evitar duplicidades de empresas no banco de dados. |

## 2.1.6 – Cadastrar parceiro

|  |
| --- |
| 1. O sistema deve permitir o cadastro de uma nova empresa no sistema. |
| 2. Durante o cadastro, o usuário deve fornecer as seguintes informações obrigatórias: nome da empresa, mensagem, assunto e endereço de e-mail. |
| 3. O sistema deve validar se todas as informações necessárias foram fornecidas pelo usuário. Caso alguma informação esteja faltando, o sistema deve exibir uma mensagem de alerta solicitando ao usuário que preencha todos os campos obrigatórios. |
| 4. Os valores fornecidos nos campos (nome da empresa, mensagem, assunto e endereço de e-mail) devem ser filtrados e sanitizados para evitar problemas de segurança, removendo qualquer conteúdo indesejado ou potencialmente perigoso. |
| 5. O sistema deve construir e executar uma consulta SQL para inserir os dados fornecidos na tabela "empresarial" do banco de dados. |
| 6. Após a inserção dos dados, o sistema deve verificar se o cadastro foi realizado com sucesso. Isso pode ser feito verificando se algum registro foi afetado no banco de dados após a execução da consulta. |
| 7. Se o cadastro for bem-sucedido, o sistema deve retornar uma confirmação ao usuário, indicando que o cadastro foi concluído com sucesso. |
| 8. Em caso de erro no cadastro, o sistema deve informar ao usuário que ocorreu um problema e que o cadastro não foi concluído. |
| 9. É importante ressaltar que o sistema deve garantir a segurança e integridade dos dados inseridos no banco de dados, utilizando técnicas adequadas de validação e armazenamento seguro. |
| 10. O processo de cadastro deve ocorrer em tempo real, à medida que o usuário preenche os campos necessários. |

Da adoção

## 2.1.7 – Cadastro de animal para adoção

|  |
| --- |
| 1. O sistema deve permitir o cadastro de um novo animal para adoção. |
| 2. Durante o cadastro, o usuário deve fornecer as seguintes informações obrigatórias: nome, sobrenome, Email e nome do animal a ser adotado. |
| 3. O sistema deve filtrar e sanitizar os valores fornecidos nos campos (nome, sobrenome, Email e nome do animal) para evitar problemas de segurança, removendo qualquer conteúdo indesejado ou potencialmente perigoso. |
| 4. O sistema deve validar se todas as informações necessárias foram fornecidas pelo usuário. Caso alguma informação esteja faltando, o sistema deve exibir uma mensagem de alerta solicitando ao usuário que preencha todos os campos obrigatórios. |
| 5. Se todas as informações forem fornecidas corretamente, o sistema deve inserir os dados na tabela "adoção" do banco de dados |
| 6. O sistema deve verificar se o cadastro foi realizado com sucesso verificando se algum registro foi afetado no banco de dados após a inserção dos dados. |
| 7. Em caso de cadastro bem-sucedido, o sistema deve retornar uma confirmação ao usuário. |
| 8. Caso ocorra algum erro durante o cadastro, o sistema deve informar ao usuário que houve um problema e o cadastro não foi concluído. |
| 9. O sistema deve permitir a listagem de todos os candidatos à adoção, recuperando os registros da tabela "adoção" do banco de dados. |
| 10. O sistema deve permitir a edição dos candidatos à adoção, atualizando os dados na tabela "adoção" do banco de dados. |
| 11. O sistema deve permitir a exclusão de um candidato à adoção, removendo o registro correspondente da tabela "adoção" do banco de dados. |

RF02 - Conexão com o Banco de Dados:

|  |
| --- |
| 1. O sistema deve estabelecer uma conexão com o banco de dados. |
| 2. A conexão deve ser realizada utilizando os parâmetros de configuração definidos no código (host, usuário, senha e nome do banco de dados). |
| 3. A classe "conexão" deve possuir um construtor que realiza a conexão com o banco de dados ao ser instanciada. |
| 4. A conexão com o banco de dados deve ser estabelecida utilizando a função "mysqli\_connect". |
| 5. Após a conexão ser estabelecida, o sistema deve selecionar o banco de dados correto utilizando a função "mysqli\_select\_db". |
| 6. Se a conexão com o banco de dados for estabelecida com sucesso e o banco de dados for selecionado corretamente, a classe "conexao" deve retornar o objeto de conexão "conn1". |
| 7. O sistema deve utilizar o objeto de conexão retornado para realizar operações de consulta, inserção, atualização ou exclusão no banco de dados. |

Do crud:

- Usuários:

RF03 - Listar Usuários:

Requisitos Funcionais:

|  |
| --- |
| 1. O sistema deve executar uma consulta SQL para selecionar todos os registros da tabela "cadastro". |
| 2. A consulta SQL deve ser construída utilizando o comando SELECT \* FROM cadastro. |
| 3. O sistema deve utilizar o objeto de conexão retornado pela classe "conexao" para executar a consulta no banco de dados. |
| 4. Os resultados da consulta devem ser armazenados em uma variável. |
| 5. Para cada registro retornado pela consulta, o sistema deve criar um objeto "UsuarioDB" e preencher suas propriedades com os valores recuperados do banco de dados. |
| 6. Os objetos "UsuarioDB" criados devem ser adicionados a um array de usuários. |
| 7. Ao final da iteração, o sistema deve retornar o array contendo todos os usuários recuperados do banco de dados. |

Resultado Esperado:

O sistema deve ser capaz de listar todos os usuários cadastrados no sistema. Os usuários devem ser apresentados de forma estruturada, permitindo a visualização de seus dados, como ID, nome, sobrenome, senha, CPF e e-mail.

Observação:

Essa é uma descrição simplificada do requisito funcional para listar usuários. Em uma implementação real, é importante considerar aspectos adicionais, como paginação dos resultados, ordenação, filtros de pesquisa e tratamento de erros.

RF04 - Editar Usuário:

|  |
| --- |
| 1. O sistema deve receber como entrada o ID do usuário a ser editado, bem como os novos valores para nome, sobrenome, senha e e-mail. |
| 2. O sistema deve construir uma consulta SQL do tipo UPDATE para atualizar os dados do usuário na tabela "cadastro". |
| 3. A consulta SQL deve ser construída utilizando o comando UPDATE cadastro SET nome = ?, sobrenome = ?, senha = ?, email = ? WHERE id = ?. |
| 4. O sistema deve utilizar o objeto de conexão retornado pela classe "conexao" para executar a consulta no banco de dados. |
| 5. Os parâmetros da consulta devem ser definidos corretamente para evitar a injeção de SQL. Os valores do nome, sobrenome, senha e e-mail devem ser vinculados à declaração preparada (prepared statement). |
| 6. O sistema deve executar a declaração preparada para realizar a atualização dos dados do usuário. |
| 7. Após a execução da declaração, o sistema deve verificar se algum registro foi afetado no banco de dados. |
| 8. Se algum registro tiver sido afetado, significa que a edição foi bem-sucedida, e o sistema deve retornar um valor verdadeiro (true). Caso contrário, significa que ocorreu um erro durante a edição, e o sistema deve retornar um valor falso (false). |

Resultado Esperado:

O sistema deve permitir a edição dos dados de um usuário cadastrado. Ao realizar a edição e salvar as modificações, o sistema deve retornar um valor indicando se a operação foi bem-sucedida ou não.

Observação:

Essa é uma descrição simplificada do requisito funcional para editar usuário. Em uma implementação real, é importante considerar aspectos adicionais, como validação dos dados, restrições de tamanho e formato, e tratamento de erros.

RF05 - Deletar Usuário:

|  |
| --- |
| 1. O sistema deve receber como entrada o ID do usuário a ser deletado. |
| 2. O sistema deve construir uma consulta SQL do tipo DELETE para remover o registro do usuário da tabela "cadastro". |
| 3. A consulta SQL deve ser construída utilizando o comando DELETE FROM cadastro WHERE id = ?. |
| 4. O sistema deve utilizar o objeto de conexão retornado pela classe "conexao" para executar a consulta no banco de dados. |
| 5. Os parâmetros da consulta devem ser definidos corretamente para evitar a injeção de SQL. O valor do ID do usuário deve ser vinculado à declaração preparada (prepared statement). |
| 6. O sistema deve executar a declaração preparada para realizar a exclusão do usuário. |
| 7. Após a execução da declaração, o sistema deve verificar se o registro foi removido do banco de dados. |
| 8. Se o registro tiver sido removido com sucesso, o sistema deve exibir uma mensagem informando que o usuário foi deletado. |
| 9. Caso contrário, o sistema deve exibir uma mensagem informando que ocorreu um erro durante a exclusão do usuário |

.

Resultado Esperado:

O sistema deve permitir a exclusão de um usuário cadastrado. Ao realizar a exclusão, o sistema deve exibir uma mensagem indicando se a operação foi realizada com sucesso ou se ocorreu algum erro.

Observação:

Essa é uma descrição simplificada do requisito funcional para deletar usuário. Em uma implementação real, é importante considerar aspectos adicionais, como confirmação do usuário antes da exclusão, restrições de exclusão em cascata, entre outros.

Dos Parceiros

RF05 - Listar Empresas:

|  |
| --- |
| 1. O sistema deve criar um array vazio chamado "empresas" para armazenar as informações das empresas. |
| 2. O sistema deve construir uma consulta SQL do tipo SELECT para buscar todos os registros da tabela "empresarial". |
| 3. O sistema deve utilizar o objeto de conexão retornado pela classe "conexao" para executar a consulta no banco de dados. |
| 4. O sistema deve percorrer os resultados obtidos da consulta utilizando um loop, recuperando as informações de cada empresa. |
| 5. Para cada registro retornado, o sistema deve criar um objeto da classe "empresa" e preencher seus atributos com os valores correspondentes do banco de dados. |
| 6. O objeto da empresa preenchido deve ser adicionado ao array de empresas. |
| 7. Após percorrer todos os registros, o sistema deve retornar o array de empresas contendo todas as informações das empresas cadastradas. |

Resultado Esperado:

O sistema deve ser capaz de listar todas as empresas cadastradas e retornar um array contendo as informações das empresas. Cada elemento do array deve corresponder a um objeto da classe "empresa" preenchido com os dados do respectivo registro do banco de dados.

RF04 - Editar Empresa:

|  |
| --- |
| 1. O sistema deve receber como entrada o ID da empresa a ser editada, juntamente com os novos valores para nome da empresa, assunto, email e mensagem. |
| 2. O sistema deve construir uma consulta SQL do tipo UPDATE para atualizar os dados da empresa na tabela "empresarial". |
| 3. A consulta SQL deve ser construída utilizando o comando UPDATE empresarial SET empresa = ?, assunto = ?, email = ?, mensagem = ? WHERE id = ?. |
| 4. O sistema deve utilizar o objeto de conexão retornado pela classe "conexao" para executar a consulta no banco de dados. |
| 5. Os parâmetros da consulta devem ser definidos corretamente para evitar a injeção de SQL. Os novos valores para nome da empresa, assunto, email e mensagem devem ser vinculados à declaração preparada (prepared statement), juntamente com o valor do ID da empresa. |
| 6. O sistema deve executar a declaração preparada para realizar a atualização dos dados da empresa. |
| 7. Após a execução da declaração, o sistema deve verificar se houve alguma alteração nos registros do banco de dados. |
| 8. Se alguma alteração tiver sido realizada com sucesso, o sistema deve retornar true. |
| 9. Caso contrário, o sistema deve retornar false, indicando que ocorreu algum erro durante a edição dos dados da empresa. |

Resultado Esperado:

O sistema deve permitir a edição dos dados de uma empresa cadastrada. Ao realizar a edição, o sistema deve retornar true se a operação for bem-sucedida ou false em caso de erro.

Dos animais:

RF08 - Listar Animais:

|  |
| --- |
| 1. O sistema deve criar um array vazio chamado "animais" para armazenar as informações dos animais. |
| 2. O sistema deve construir uma consulta SQL do tipo SELECT para buscar todos os registros da tabela "animais". |
| 3. O sistema deve utilizar o objeto de conexão retornado pela classe "conexao" para executar a consulta no banco de dados. |
| 4. O sistema deve percorrer os resultados obtidos da consulta usando um loop, recuperando as informações de cada animal. |
| 5. Para cada registro retornado, o sistema deve criar um objeto da classe "AnimalDB" e preencher seus atributos com os valores correspondentes do banco de dados. |
| 6. O objeto do animal preenchido deve ser adicionado ao array de animais. |
| 7. Após percorrer todos os registros, o sistema deve retornar o array de animais contendo todas as informações dos animais cadastrados. |

Resultado Esperado:

O sistema deve ser capaz de listar todos os animais cadastrados e retornar um array contendo as informações dos animais. Cada elemento do array deve corresponder a um objeto da classe "AnimalDB" preenchido com os dados do respectivo registro do banco de dados.

RF09 - Deletar Animal:

|  |
| --- |
| 1. O sistema deve receber como entrada o ID do animal a ser deletado. |
| 2. O sistema deve construir uma consulta SQL do tipo DELETE para remover o registro do animal da tabela "animais". |
| 3. O sistema deve utilizar o objeto de conexão retornado pela classe "conexao" para executar a consulta no banco de dados. |
| 4. O sistema deve executar a consulta de exclusão. |
| 5. O sistema deve verificar se a consulta de exclusão foi executada com sucesso verificando se o resultado da execução é verdadeiro (true). |
| 6. Se a exclusão for bem-sucedida, o sistema deve exibir uma mensagem de sucesso. |
| 7. Em caso de falha na exclusão, o sistema deve exibir uma mensagem de erro. |

Resultado Esperado:

O sistema deve permitir que o usuário delete um animal cadastrado. Ao realizar a exclusão, o sistema deve exibir uma mensagem de sucesso se a operação for bem-sucedida ou uma mensagem de erro em caso de falha.

RF01 - Listar Candidatos à Adoção:

|  |
| --- |
| - O sistema deve permitir a listagem de todos os candidatos à adoção. |
| - O sistema deve recuperar os registros da tabela "adocao" do banco de dados. |
| - Os candidatos à adoção devem ser armazenados em um array. |
| - Para cada registro recuperado, deve ser criada uma instância da classe "adocao" e preenchida com os dados correspondentes. |
| - As informações do candidato à adoção, como ID, nome, sobrenome, nome do bichinho e email, devem ser atribuídas às propriedades da instância da classe "adocao". |
| - A instância da classe "adocao" preenchida deve ser adicionada ao array de candidatos. |
| - Ao final, o array de candidatos deve ser retornado. |

RF02 - Editar Candidato à Adoção:

|  |
| --- |
| - O sistema deve permitir a edição dos dados de um candidato à adoção existente. |
| - O sistema deve atualizar os dados na tabela "adocao" do banco de dados. |
| - Os parâmetros a serem editados são: ID, nome, sobrenome, email e nome do bichinho. |
| - O sistema deve preparar e executar uma instrução SQL do tipo UPDATE para atualizar os dados do candidato à adoção correspondente. |
| - Os valores dos parâmetros devem ser vinculados à instrução SQL utilizando o método mysqli\_stmt\_bind\_param(). |
| - Após a execução da instrução SQL, o sistema deve verificar se algum registro foi afetado no banco de dados. |
| - Se algum registro foi afetado, significa que a edição foi bem-sucedida e o sistema deve retornar true. |
| - Caso contrário, o sistema deve retornar false. |

RF03 - Deletar Candidato à Adoção:

|  |
| --- |
| - O sistema deve permitir a exclusão de um candidato à adoção. |
| - O sistema deve remover o registro correspondente da tabela "adocao" do banco de dados. |
| - O parâmetro a ser utilizado é o ID do candidato à adoção. |
| - O sistema deve preparar e executar uma instrução SQL do tipo DELETE para remover o registro do candidato à adoção correspondente. |
| - Após a execução da instrução SQL, o sistema deve verificar se a remoção foi bem-sucedida. |
| - Se a remoção foi bem-sucedida, o sistema deve exibir uma mensagem de alerta informando que a adoção foi deletada com sucesso e redirecionar o usuário para a página de listagem de animais. |
| - Caso contrário, o sistema deve exibir uma mensagem de alerta informando que ocorreu uma falha ao deletar o candidato à adoção. |

## **Requisitos não funcionais**

Explicação

## 2.2.1 - Tela de cadastro

RN01-

|  |
| --- |
| 1. Usabilidade: O sistema deve ter uma interface intuitiva e de fácil utilização, facilitando o cadastro de novos usuários. |
| 2. Segurança: As informações fornecidas pelos usuários, como senha e CPF, devem ser armazenadas de forma segura, utilizando técnicas de criptografia e proteção contra acesso não autorizado. |
| 3. Desempenho: O sistema deve ser capaz de realizar o processamento e validação das informações de forma eficiente, garantindo tempos de resposta rápidos mesmo em momentos de alto tráfego. |
| 4. Disponibilidade: O sistema deve estar disponível e acessível para os usuários durante o período de cadastro, evitando indisponibilidades que possam impedir a realização do cadastro. |
| 5. Confiabilidade: O sistema deve ser capaz de lidar com situações inesperadas, como falhas de conexão ou erros no processamento das informações, de forma a não comprometer a integridade dos dados e garantir que o cadastro seja realizado corretamente. |
| 6. Escalabilidade: O sistema deve ser capaz de lidar com um número crescente de usuários e cadastros, sem comprometer o desempenho ou a disponibilidade. |
| 7. Manutenibilidade: O sistema deve ser projetado e implementado de forma modular, facilitando a manutenção e a realização de futuras melhorias. |
| 8. Internacionalização: O sistema deve ser capaz de lidar com diferentes idiomas e formatos de dados, permitindo o cadastro de usuários de diversas nacionalidades |
| 9. Rastreabilidade: O sistema deve ser capaz de registrar logs de atividades relacionadas ao cadastro de usuários, permitindo rastrear possíveis problemas ou eventos relevantes. |
| 10. Compliance: O sistema deve estar em conformidade com as leis e regulamentações aplicáveis em relação à proteção de dados pessoais, garantindo a privacidade dos usuários e o tratamento adequado das informações fornecidas. |

.

## 2.2.2- Tela de login

|  |
| --- |
| 1. Segurança: A verificação de credenciais de login deve ser realizada de forma segura, protegendo as informações sensíveis, como senhas, contra acesso não autorizado. |
| 2. Desempenho: O sistema deve ser capaz de realizar a verificação das credenciais de forma eficiente, garantindo tempos de resposta rápidos e evitando impactos significativos no desempenho do sistema. |
| 3. Disponibilidade: O sistema deve estar disponível e acessível para os usuários durante o processo de verificação de credenciais, evitando indisponibilidades que possam impedir o acesso ao sistema. |
| 4. Confiabilidade: O sistema deve ser capaz de lidar com situações inesperadas, como falhas de conexão ou erros no processamento das informações de credenciais, de forma a não comprometer a integridade dos dados e garantir a correta verificação das credenciais de login. |
| 5. Rastreabilidade: O sistema deve registrar logs de tentativas de login, permitindo rastrear atividades suspeitas, identificar possíveis ataques ou violações de segurança e fornecer informações para fins de auditoria. |
| 6. Usabilidade: O sistema deve fornecer mensagens de erro claras e adequadas, facilitando a compreensão do usuário em caso de falha na verificação das credenciais de login. |
| 7. Manutenibilidade: O sistema deve ser projetado e implementado de forma modular, facilitando a manutenção e a realização de futuras melhorias na verificação de credenciais de login. |
| 8. Escalabilidade: O sistema deve ser capaz de lidar com um número crescente de usuários e verificações de credenciais de login, sem comprometer o desempenho ou a disponibilidade. |
| 9. Compliance: O sistema deve estar em conformidade com as leis e regulamentações aplicáveis em relação à proteção de dados pessoais, garantindo a privacidade dos usuários e o tratamento adequado das informações de credenciais de login. |
| 10. Monitoramento: O sistema deve ser capaz de monitorar a atividade de login, detectar e responder a possíveis tentativas de login maliciosas ou suspeitas, visando proteger a segurança das credenciais dos usuários. |

2.2.3 - Feed de fotos

|  |
| --- |
| 1. Usabilidade: O sistema deve apresentar uma galeria de fotos e descrições dos animais de forma visualmente atraente e intuitiva, facilitando a escolha do animal desejado pelos usuários. |
| 2. Confiabilidade: O sistema deve ser capaz de lidar com o fluxo de preenchimento do formulário de adoção de forma confiável, garantindo a precisão e a integridade dos dados inseridos. |
| 3. Segurança: O sistema deve proteger as informações pessoais fornecidas pelos usuários durante o preenchimento do formulário de adoção, utilizando técnicas de criptografia e proteção contra acesso não autorizado. |
| 4. Desempenho: O sistema deve ser capaz de exibir a galeria de fotos e as descrições dos animais de forma rápida e responsiva, evitando atrasos significativos no carregamento das imagens. |
| 5. Disponibilidade: O sistema deve estar disponível e acessível para os usuários durante o acesso à galeria de fotos e durante o preenchimento do formulário de adoção, evitando indisponibilidades que possam impedir o processo de adoção. |
| 6. Manutenibilidade: O sistema deve ser projetado e implementado de forma modular, facilitando a manutenção e a realização de futuras melhorias no feed de fotos e no formulário de adoção. |
| 7. Internacionalização: O sistema deve ser capaz de lidar com diferentes idiomas e formatos de dados, permitindo a adoção de animais por usuários de diferentes nacionalidades. |
| 8. Rastreabilidade: O sistema deve registrar os dados do formulário de adoção, armazenando informações como a data e hora de envio, para fins de rastreabilidade e referência futura. |
| 9. Compliance: O sistema deve estar em conformidade com as leis e regulamentações aplicáveis em relação à proteção de dados pessoais, garantindo a privacidade dos usuários e o tratamento adequado das informações fornecidas no formulário de adoção. |
| 10. Feedback em tempo real: O sistema deve realizar a validação dos campos obrigatórios em tempo real, fornecendo feedback imediato ao usuário sobre campos que não foram preenchidos corretamente ou estão em falta. |
| 11. Integração: O sistema deve ser capaz de enviar corretamente as informações preenchidas no formulário de adoção para o processamento posterior, integrando-se com outros sistemas ou serviços necessários para dar continuidade ao processo de adoção. |
| 12. Armazenamento seguro: O sistema deve garantir a segurança e proteção das informações pessoais fornecidas pelo usuário durante o preenchimento do formulário de adoção, armazenando-as de forma adequada e evitando acessos não autorizados. |

# Materiais e métodos

Este capítulo apresenta as ferramentas utilizadas para a elaboração do aplicativo móvel e o sistema web. Também serão apresentadas as metodologias utilizadas para atingir o resultado final.

## Materiais

Os materiais utilizados no desenvolvimento do sistema proposto foram:]

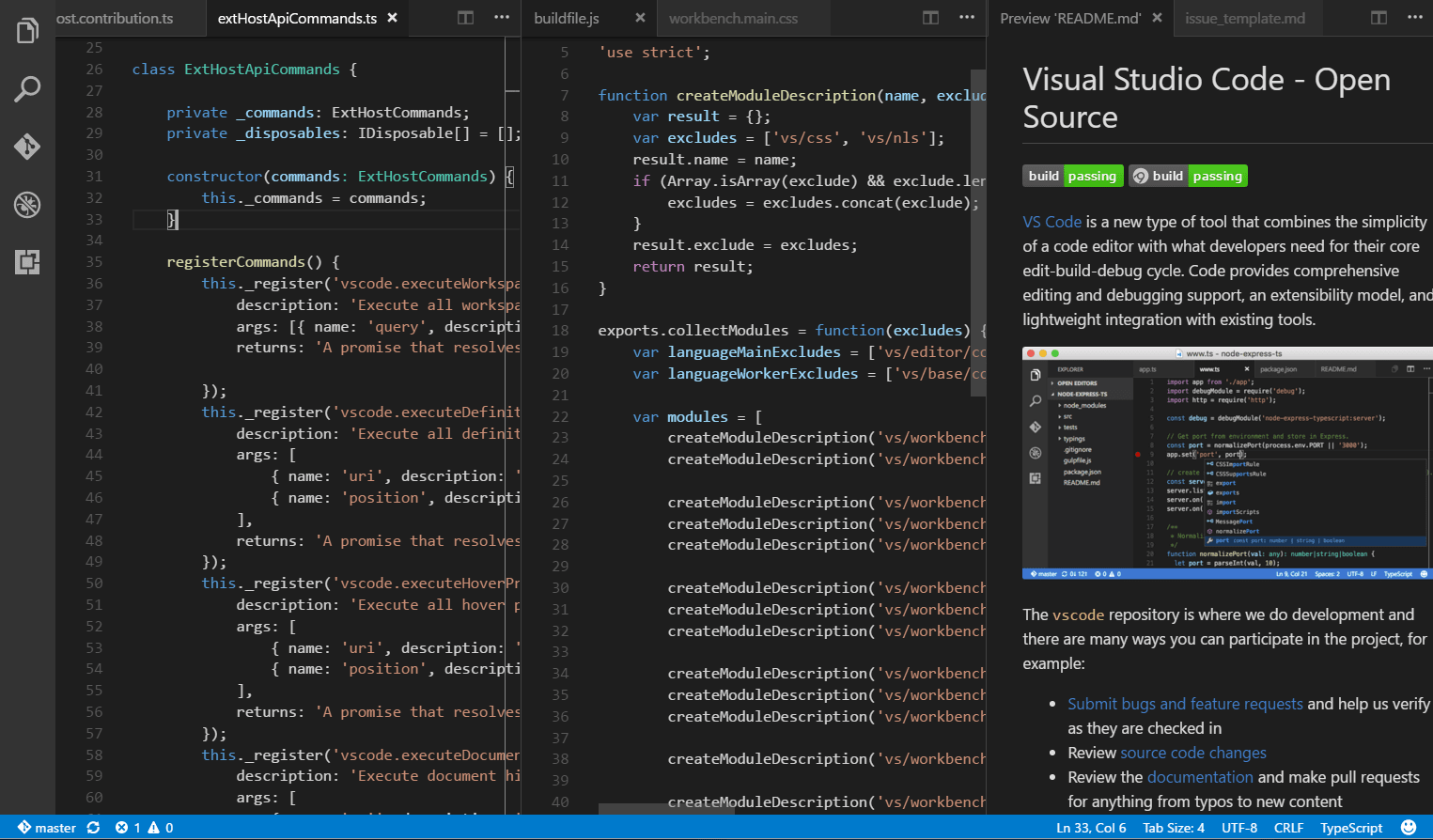
• Aplicativo Web

1. **Visual studio code**

O Visual Studio Code (VS Code) é um editor de código de código aberto desenvolvido pela Microsoft. A saber, ele está disponível para Windows, Mac e Linux. É criado com [Electron](https://www.treinaweb.com.br/blog/o-que-e-electron/), ferramenta criada pelo [GitHub](https://www.treinaweb.com.br/blog/git-e-github-quais-as-diferencas) que permite a criação de softwares Desktop com [HTML, CSS](https://www.treinaweb.com.br/blog/o-que-e-e-como-comecar-com-html-e-css/) e [JavaScript](https://www.treinaweb.com.br/blog/o-que-e-e-como-comecar-com-javascript).

Visual studio code é gratuito e está disponível no próprio site do Visual Studio Code

A seguir uma visão geral do sistema:



Visão geral do sistema Visual Studio Code

Fonte: <https://code.visualstudio.com/docs/getstarted/userinterface>

1. **Google Chrome**

Google Chrome é um navegador para internet, criado pela Google, combina tecnologias sofisticadas com um design simples para tornar a Web mais rápida, mais segura e mais prática. Um dos recursos que essa ferramenta possui e foi utilizado no decorrer do desenvolvimento do sistema proposto, é o recurso de inspeção de código. Com ele foi possível escolher características visuais para o sistema, depurar códigos e acompanhar algumas funcionalidades. Este recurso pode ser acessado pela tecla de atalho “F12” ou botão direito do mouse sobre o conteúdo web, em menu suspenso, opção “Inspecionar Elemento”.

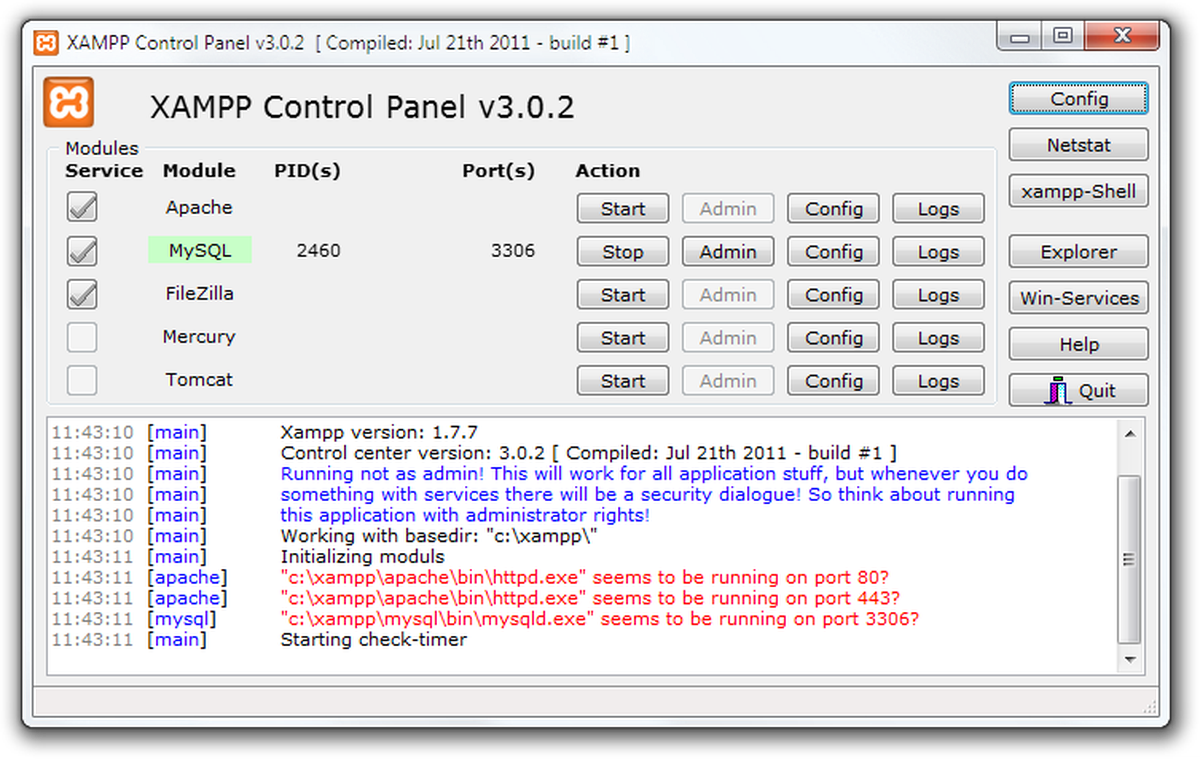
A seguir visão geral do sistema:

Uma imagem com texto, software, Ícone de computador, Página web

Descrição gerada automaticamente

1. **Xampp**

O XAMPP é uma solução de código aberto que é utilizada como um servidor local, simulando um servidor web real em uma máquina doméstica. É uma solução muito utilizada por estudantes que precisam testar os seus códigos e por profissionais de desenvolvimento que ainda não têm um orçamento para alugar um ambiente de testes remoto.



Sistema geral do xampp

Fonte: imagem do google

1. **HTML**

HTML (Linguagem de Marcação de HiperTexto) é o bloco de construção mais básico da web. Define o significado e a estrutura do conteúdo da web. Outras tecnologias além do HTML geralmente são usadas para descrever a aparência/apresentação (CSS) ou a funcionalidade/comportamento (JavaScript) de uma página da web.

1. **PHP**

PHP (Hypertext Preprocessor) é uma linguagem de programação de código aberto amplamente utilizada para desenvolvimento web. Ela é especialmente adequada para criar aplicativos e sites dinâmicos, nos quais o conteúdo é gerado de forma personalizada para cada usuário.

O PHP é uma linguagem interpretada que é executada no lado do servidor, ou seja, o código é processado pelo servidor web antes de ser enviado para o navegador do usuário. Isso permite que o PHP se comunique com bancos de dados, gerencie sessões, processe formulários e realize muitas outras tarefas necessárias para a criação de aplicativos web interativos.

## **Métodos**

## 3.2.1 Metodologia de desenvolvimento

1. **Projeto**

Para a atividade de projeto, foi elaborada uma arquitetura, através da confecção de alguns diagramas dispostos no decorrer deste trabalho. Para este foi utilizado à ferramenta Astah Community, está utiliza a linguagem UML para modelagem e auxilia na visualização dos produtos dos trabalhos e a comunicação entre os objetos em diagramas padronizados.

1. **Codificação**

Para esta atividade foram utilizadas ferramentas Visual Studio code, ferramenta de edição de texto livre especialmente configurada para as linguagens PHP, HTML, JS e SQL, utilizada para a confecção e manutenção dos arquivos PHP, HTML e JS do sistema web, e Xampp.

## **3.2.3 INSTALAÇÃO**

Para realizar a instalação do projeto em sua maquina é necessário o uso do aplicativo XAMPP(com a versão mais recente deve funcionar normalmente).

Clone o repositório para sua maquina local.

Abra o Xampp.

Ative as opções Apache e MySQL.

Logo após vá em Explorer, que aparece no lado direito do programa.

Busque pela pasta HTDOCS.

Na pasta HTDOCS, crie uma pasta chamada POO.

Coloque os arquivos e pastas nessa pasta.

## **3.2.4 USO**

Após o passo anterior (Instalação) é possivel continuar configurando sua máquina

Com seu Xampp aberto e seu VSCODE também, inicie seu navegador padrão do windows e digite <http://localhost/>.

Após isso entre na pasta POO que estará agora aparecendo em sua tela.

Clique e pronto você estará visualizando o projeto.

Aproveite as funcionalidades do projeto tais como [feed], [se cadastrar] , [adotar],[colocar para adoção],[fechar parcerias] etc.

MYSQL

Para configurar seu banco de dados são necessarias as seguintes ações:

No seu projeto clonado existe um arquivo chamado dbcadastro.php

Há um codigo SQL inserido no arquivo, copie o codigo completo e em seguida abra seu XAMPP.

Com o painel Xampp Control Painel aberto clique na opção admin do modulo MYSQL .

Após isso será aberto uma guia no seu navegador padrão com o seguinte caminho: <http://localhost/phpmyadmin/>

Vá em novo e digite o nome do Database: dbcadastro, após isso selecione utf8mb4\_unicode\_ci.

Clique em Criar após isso será aberta outra guia, clique em SQL e cole o código que está em sua Área de Transfêrencia.

# Modelagem do sistema E CASOS DE USO

A modelagem de sistemas é uma atividade fundamental no desenvolvimento de software, que visa criar representações estruturadas e abstratas de um sistema em questão. Essas representações, chamadas de modelos, capturam os principais aspectos do sistema, incluindo seus requisitos, funcionalidades, processos, dados e interações com outros elementos.

Neste capítulo, exploraremos a importância da modelagem de sistemas e seu papel no ciclo de vida no desenvolvimento desse software.

4.1 diagramação geral do site

Uma imagem com texto, file, diagrama, captura de ecrã

Descrição gerada automaticamente

Visão geral do sistema

## **4.2 Casos de uso**

### UC01 - Fazer Login

**1 Nome do Caso de Uso**

Fazer Login

**2 Breve descrição**

Autêntica o usuário no aplicativo dando a ele acesso às informações de sua conta, bem como às funcionalidades de gerar fatura, alterar limite, falar com um atendente, etc.

3 Atores

Usuário

**4 Fluxo de eventos**

* 1. Fluxo básico

1. O caso de uso se inicia quando o ator clica em “Login”.
2. O sistema exibe a tela inicial da web aplicação sem estar logado
3. O sistema exibe campos de entrada para CPF e senha.
4. O ator digita o seu CPF e senha [FA01]
5. O ator clica em “entrar”
6. O sistema autêntica o autor. [FE01]
   1. Fluxos alternativos

FA01- O usuário esqueceu a senha

1. Logo após digitar o CPF, o autor seleciona a opção para alterar senha.
2. O sistema exibe uma tela para o usuário trocar de senha.
3. o sistema exibe campos para a entrada do CPF e sua nova senha.
4. O autor clica em “Confirmar”.
5. O sistema redireciona o autor para a tela de login novamente.
   1. Fluxos de exceção

FE01 – Preenchimento incorreto da senha

1. O sistema informa que a senha inserida está incorreta.
2. A aplicação retorna a tela de inserção de dados

5- Pré condições

1. Ser cadastrado no sistema.
2. Estar conectado a internet.

6- Pós condições

O usuário é logado no sistema e tem acesso a sua conta.

Regra de negócio

RN01

1. Descrição: senha de no mínimo 6 caracteres, associada ao cpf digitado.
2. Obrigatoriedade: Sim.

## UC02- Cadastrar usuário

1 Nome do caso de uso

Cadastrar usuário

2 Breve descrição

Cadastrar visitante

3 Atores

Usuário

4 Fluxo de eventos

4.1 Fluxo básico - Preenchimento de informações

1. O caso de uso começa quando o usuário clica em “Cadastrar”.

2. o sistema apresenta na página inicial campos como: Nome, Sobrenome, CPF, Email, senha, confirmação de senha

3. O usuário preenche as informações [FE01]

4. O sistema exibe que o usuário foi cadastrado [FE02][FE03][FE04]

4.2 Fluxos de exceção

FE01 – Usuário não preencheu as informações

1. O sistema exibe uma mensagem de erro dizendo ao usuário que falta informações nos campos
2. O sistema retorna para o cadastro

FE02 – Usuário Já existente

1. O sistema exibe uma mensagem de erro dizendo ao usuário que o usuário já está sendo utilizado
2. O sistema retorna para a tela de cadastro

FE03 – Senhas incorretas

1. O sistema exibe uma mensagem de erro dizendo ao usuário que as senhas não coincidem
2. O sistema retorna para a tela de cadastro.

FE04 – Senha menor que 6 dígitos

1. O sistema exibe uma mensagem de erro dizendo que a senha que foi inserida pelo usuário possui menos que 6 dígitos
2. O sistema retorna para a tela de cadastro

5 Pré condições

1. Usuário novo
2. Conexão a internet

6 Pós condições

O usuário é cadastrado ao sistema e pode realizar a UC01.

## UC03 – Adoção de animais

1 Nome do caso de uso

Adoção de animais

2 Breve descrição

Feed de fotos com botão para adotar

3 Atores

Usuário

4 Fluxo de eventos

4.1 Fluxo básico

1. O caso de uso começa com um feed de fotos com alguns dos animais disponíveis para a adoção

2. o usuário ao clicar na foto dos animais é exibido mais informações sobre os mesmos

3. Caso o usuário se interesse ele clicará no botão “Me adote”

4. O sistema exibirá um formulário

5. O sistema exibirá campos: Nome, Sobrenome, Email, Nome do bichinho[FE01]

6. Após o usuário finalizar o preenchimento ele clicará em “concluir”.

4.2 Fluxos de exceção

FE01 – Usuário não preencheu as informações

1. O sistema exibe uma mensagem de erro dizendo ao usuário que falta informações nos campos
2. O sistema retorna para o cadastro

## 4.2.1 – Observação

A mesma lógica se repete em todas as pagina presentes dentro do sistema, tais como, Parceiros, Voluntários e colocar para adotar.

# Telas do projeto

* 1. Tela inicial

Uma imagem com texto, captura de ecrã

Descrição gerada automaticamente

* 1. Tela sobre nós

Uma imagem com texto, captura de ecrã, Tipo de letra, documento

Descrição gerada automaticamente

Uma imagem com captura de ecrã, texto, cor-de-rosa, Lilás

Descrição gerada automaticamente

* 1. Feed de fotos

Uma imagem com texto, captura de ecrã, Página web, Website

Descrição gerada automaticamente

* 1. Tela de cadastro

Uma imagem com texto, captura de ecrã, círculo, design

Descrição gerada automaticamente

* 1. Tela de login

Uma imagem com texto, captura de ecrã, Sistema operativo, Tipo de letra

Descrição gerada automaticamente

## Tela do sistema

Uma imagem com texto, captura de ecrã, carta

Descrição gerada automaticamente

## Tela de parceria

Uma imagem com texto, captura de ecrã, Sistema operativo, software

Descrição gerada automaticamente

## Tela colocar para adotar

Uma imagem com texto, captura de ecrã, Sistema operativo, software

Descrição gerada automaticamente

## **5.9 Tela de voluntariado**

Uma imagem com texto, captura de ecrã, Sistema operativo, design

Descrição gerada automaticamente

## **5.10 Tela para administradores**

Uma imagem com texto, captura de ecrã, Sistema operativo

Descrição gerada automaticamente

## **5.11 Listagem para o crud**

Uma imagem com texto, captura de ecrã, Sistema operativo, software

Descrição gerada automaticamente

# Conclusão

Ao longo deste trabalho final de desenvolvimento web, exploramos a criação de um sistema de adoção de animais, que visa conectar pessoas interessadas em adotar animais de estimação a animais disponíveis para adoção. Durante o processo de desenvolvimento, foram abordadas diversas etapas, desde a modelagem do sistema até a implementação de funcionalidades essenciais.

Iniciamos nosso trabalho com a modelagem do sistema, utilizando diagramas de classes, atividades e casos de uso para representar as entidades, os fluxos de atividades e as interações entre os atores e o sistema. Esses diagramas nos forneceram uma visão clara das funcionalidades e dos requisitos do sistema, permitindo uma melhor compreensão de seu funcionamento.

Em seguida, avançamos para a implementação do sistema, utilizando tecnologias web modernas. Utilizamos PHP como linguagem de programação do lado do servidor para processar as requisições e manipular os dados. Também utilizamos HTML, CSS e JavaScript para desenvolver a interface do usuário, tornando-a atraente e responsiva em diferentes dispositivos.

Durante o desenvolvimento, implementamos funcionalidades importantes, como o feed de fotos dos animais disponíveis para adoção, o formulário de adoção, o cadastro de usuários, parceiros e voluntários, bem como o sistema administrativo com operações CRUD. Essas funcionalidades proporcionam uma experiência abrangente aos usuários, facilitando o processo de adoção e fornecendo recursos adicionais para a gestão do sistema.

Além disso, destacamos a importância de práticas de segurança no desenvolvimento web, como a proteção contra vulnerabilidades de segurança, o armazenamento seguro de senhas e a validação adequada dos dados de entrada. Essas medidas são essenciais para garantir a integridade e a confidencialidade dos dados dos usuários, bem como a proteção contra ameaças cibernéticas.

Por fim, concluímos que o desenvolvimento web é um campo em constante evolução, com infinitas possibilidades e desafios. Através deste projeto, obtivemos uma visão prática de como criar um sistema web completo, desde a concepção até a implementação. Ao aplicar os conceitos aprendidos ao longo do curso, pudemos criar um sistema funcional e útil, que atende às necessidades dos usuários e contribui para a causa da adoção responsável de animais de estimação.

Este trabalho nos proporcionou um valioso aprendizado sobre o processo de desenvolvimento web, bem como o trabalho em equipe e a aplicação de metodologias adequadas. Estamos satisfeitos com o resultado alcançado e esperamos que este sistema possa fazer a diferença, promovendo a adoção de animais e contribuindo para o bem-estar dos animais de estimação e das pessoas envolvidas.

Em suma, o desenvolvimento web é uma área repleta de oportunidades e desafios, e este projeto nos permitiu adquirir habilidades valiosas para enfrentar esses desafios. Com dedicação, aprendizado contínuo e paixão pela criação de soluções web, estamos prontos para enfrentar novos projetos e contribuir para o avanço da tecnologia web.