UNIVERSIDADE FEDERAL DO MARANHÃO
CENTRO DE CIÊNCIAS EXATAS E TECNOLOGIA
ENGENHARIA DA COMPUTAÇÃO

JOÃO FELIPE MELO DA LUZ

Software para otimização de custeio e rastreabilidade do pescado de caranguejo na região de Araioses-MA

São Luís

2025

JOÃO FELIPE MELO DA LUZ

Software para otimização de custeio e rastreabilidade do pescado de caranguejo na região de Araioses-MA

Monografia apresentada ao curso de engenharia da computação da Universidade Federal do Maranhão, como parte dos requisitos necessários para obtenção do grau de bacharel em engenharia da computação.

Orientador: Prof. Dr. Haroldo Gomes

São Luís

2025

Agradecimentos

Primeiramente a Deus por proporcionar saúde e força para vencer todas as dificuldades. A esta universidade, em especial, seu corpo docente que colaboraram diretamente na nossa caminhada. Ao meu orientador Haroldo Gomes, pelo suporte no tempo que lhe coube, pelas suas correções e incentivos. A minha família, em especial ao meus pais, pelo amor, incentivo e apoio incondicional. E a todos que participaram direta ou indiretamente da minha formação.

Software para otimização de custeio e rastreabilidade da pesca de caranguejo na região de Araioses-MA

Resumo: A atividade pesqueira de caranguejo é essencial para a economia local na região de Araioses-MA, mas enfrenta desafios significativos relacionados à sustentabilidade e à gestão eficiente dos recursos. Este trabalho visa o desenvolvimento e a implementação de um software para otimizar o processo de custeio e rastreabilidade da pesca de caranguejo na região. O *software* proposto aborda essas questões ao permitir o rastreio de todo o ciclo de vida dos caranguejos, desde a captura até a comercialização. Utilizando *Angular* para o *front-end* e *PHP* para o *back-end*, o sistema proporciona uma visão objetiva e detalhada das atividades pesqueiras. O desenvolvimento e a eficácia do software serão avaliados com base em testes e *feedback* dos usuários. O estudo contempla a análise das tecnologias envolvidas e a aplicação do sistema, visando aprimorar o controle e a gestão dos recursos pesqueiros.

Palavras chave: *Software* de gestão, Desenvolvimento de *software*; sustentabilidade; Pesca artesanal.

Software for Optimization of Costing and Traceability of Crab Fishing in the Araioses-MA Region

Abstract: The crab fishing activity is crucial to the local economy in the Araioses-MA region but faces significant challenges related to sustainability and efficient resource management. This work aims to develop and implement software to optimize the cost management and traceability of crab fishing in the region. The proposed software addresses these issues by allowing tracking of the entire lifecycle of crabs, from capture to commercialization. Using Angular for the front-end and PHP for the back-end, the system provides an objective and detailed view of fishing activities. The development and effectiveness of the software will be evaluated based on testing and user feedback. The study includes an analysis of the involved technologies and the application of the system, aiming to enhance the control and management of fishing resources.

Keywords: Management software, Software development, Sustainability, Artisanal fishing.

SUMÁRIO

1 Introdução..... 10
1.1 Justificativa..... 11
2 Objetivos..... 12
2.1 Geral..... 12
2.1. Específicos..... 12
3 Referencial teórico..... 13
3.1 Papel Econômico e Subsistência da pesca artesanal..... 13
3.2 Pesca artesanal na Região de Araioses..... 13
3.3 Linguagens e framework..... 14
3.3.1 PHP..... 14
3.3.3 Laravel..... 15
3.3.2 Angular..... 15
3.4 Progressive Web Apps (PWA)..... 16
3.6 Armazenamento de Dados - MySql..... 17
3.7 Hospedagem AWS..... 17
3.8 Padrão de Arquitetura de Software (MVC)..... 18
3.8 Método de avaliação do software..... 20
4 Metodologia..... 22
4.1 Atores..... 22
4.2 Requisitos funcionais..... 22
4.3 Requisitos não funcionais..... 24
4.4 Modelagem do software..... 25
4.4.1 Caso de Uso..... 25
4.4.2 Diagrama de Classes..... 27
4.4.3 Diagrama de Sequência..... 29
4.5 Convenções de Nomenclatura..... 31
4.6 Desenvolvimento do software..... 31
4.6.1 Front-End..... 32
4.6.2 back-end..... 32
4.6.3 Integração back-to-front..... 33
4.7 Hospedagem e configuração..... 34
5 Resultados..... 35
5.1 telas do sistema..... 35
5.2 Avaliação do sistema..... 44
6 Considerações finais..... 49
Referências..... 50
APÊNDICES..... 53

LISTA DE FIGURAS

Figura 1: Arquitetura MVC 19
FIgura 2: Número de participantes de um teste 21
Figura 3 : Diagrama de casos de uso 26
Figura 4: Diagrama de classes 28
Figura 5: Diagrama de sequência do colaborador cadastrando pesca/venda. 30
Figura 6 : Estrutura de uso de uma aplicação web 34
Figura 7: tela de cadastro 36
Figura 8: tela de login 37
Figura 9: tela de atualizar senha 38
Figura 10 : tela inicial do administrador 39
Figura 11 : Tela do mapeamento da pesca/venda 40
Figura 12: Tela de configurações 41
Figura 13: Tela Inicial do colaborador 42
Figura 14: Cadastro de pesca 43
Figura 15: Tela informações pessoais 44
Figura 16 - Você recomendaria esse app para outras comunidades de pesca? 45
Figura 17 - Para a primeira versão de ele atende as necessidades locais? 45
Figura 18 - Como avalia a interface do app ? 46
Figura 19 - Quais funcionalidades de ver melhoradas ou adicionadas ? 46
Figura 20 - Você considera o design do aplicativo intuitivo e fácil navegação? 47
Figura 21 - As funcionalidades atendem às suas necessidades diárias? 48

LISTA DE QUADROS

Quadro 1: Permissões dos usuários 24
Quadro 2: Convenção de nomenclatura PascalCase 31
Quadro 3: Convenção de nomenclatura camelCase 31
Apêndice A Listagem de end point. 53

LISTA DE ABREVIAÇÕES E SIGLAS

MVC	Model-View-Controller
PWA	Progressive Web Apps
AWS	Amazon Web Services
DNS	Domain Name System
EC2	Elastic Compute Cloud
CI/CD	Continuous Integration/Continuous Delivery
CRUD	Create, Read, Update, Delete
REST	Representational State Transfer
IP	Internet Protocol

1 Introdução

Historicamente, a pesca de caranguejo é uma atividade econômica vital para a região de Araioses-MA, desempenhando um papel central na subsistência de muitos habitantes locais e contribuindo significativamente para a economia regional (Lopes et al., 2023) . Essa atividade, tradicionalmente realizada por pequenas embarcações com tripulações reduzidas e utilizando métodos artesanais, enfrenta desafios críticos relacionados à sustentabilidade e à gestão eficiente dos recursos pesqueiros (PEDROSA; LESSA,2017).

A falta de controle adequado sobre a captura e a comercialização dos caranguejos gera preocupações com a sobrepesca e a degradação ambiental, comprometendo a sustentabilidade a longo prazo da atividade pesqueira . A pesca de caranguejo não só é crucial para a segurança alimentar e a economia local, mas também é essencial para a manutenção das tradições culturais e dos meios de subsistência das comunidades envolvidas (GARCIA, 2016).

Este trabalho visa desenvolver e implementar um sistema de *software* para otimizar o processo de custeio e rastreabilidade da pesca de caranguejo na região de Araioses-MA, o nome do aplicativo será denominado CoopDelta. O *software* proposto é projetado para enfrentar os desafios mencionados ao permitir que os usuários acompanhem todo o ciclo de vida dos caranguejos. Sistemas de *software* de gestão, como o proposto, são ferramentas cruciais para melhorar a eficiência operacional e a transparência, facilitando a tomada de decisões.(Laudon & Laudon, 2018).

Para atingir esses objetivos, o sistema utilizará Angular para o desenvolvimento do *front-end* e PHP para o *back-end*. Angular, com suas capacidades de criar interfaces de usuário interativas e responsivas. Por outro lado, PHP fornecerá a flexibilidade necessária para o processamento de dados e a integração eficiente com bancos de dados, o que é crucial para garantir a performance e a escalabilidade do *software* (Thomson, 2017).

1.1 Justificativa

A importância da pesca de caranguejo para a comunidade local de Araioses-MA é inegável, uma vez que esta atividade representa uma fonte importante de subsistência e contribui para a economia regional. No entanto, a falta de um controle eficaz sobre a captura e a comercialização dos caranguejos tem levado a problemas, como a sobrepesca e a degradação ambiental. Esses problemas não apenas ameaçam a sustentabilidade da atividade pesqueira, mas também comprometem a viabilidade econômica e ambiental a longo prazo.

Neste contexto, a implementação de um sistema de *software* para otimizar o custeio e a rastreabilidade da pesca de caranguejo é essencial para enfrentar esses desafios. Um *software* especializado pode oferecer uma solução eficaz para esse problema, fornecendo uma visão detalhada das operações pesqueiras, o que é importante para garantir a sustentabilidade dos recursos e melhorar a eficiência das práticas de pesca.

Além dos benefícios diretos para a gestão pesqueira, este estudo também promove o conhecimento em tecnologia da informação aplicada a áreas específicas, como a pesca. A integração de tecnologias modernas como Angular e PHP. Esta pesquisa contribui para a melhoria da sustentabilidade na pesca, e também abre novas possibilidades para o uso de tecnologias da informação em setores especializados.

2 OBJETIVOS

2.1 Geral

Desenvolver e implementar um sistema de *software* para otimizar o gerenciamento da pesca de caranguejos na região de Araioses-MA, utilizando tecnologias modernas como Angular e PHP, para melhorar a eficiência na coleta de dados sobre a pesca e promover práticas de gestão sustentáveis.

2.1. Específicos

1. Apresentar estudos relacionados ao papel da pesca na região de Araioses, destacando seu impacto na subsistência local e na economia regional.
2. Modelar o sistema por meio de diagramas de classes e sequência para estruturar a interação entre os componentes.
3. Criar uma interface de usuário intuitiva para registro de captura usando Angular.
4. Desenvolver o *back-end* com PHP para gerenciar e armazenar dados de captura.
5. Adicionar geolocalização para registrar a localização das capturas.
6. Avaliar a eficácia do *software* na gestão dos dados de captura e identificar melhorias.

3 Referencial teórico

3.1 Papel Econômico e Subsistência da pesca artesanal

A pesca artesanal é amplamente reconhecida como uma atividade produtiva de pequena escala, caracterizada por um baixo investimento financeiro e certa deficiência na organização. Apesar de operar em uma escala reduzida e com recursos limitados, a pesca artesanal é responsável por mais da metade das capturas globais de pescado. Esse setor desempenha um papel crucial na economia global, ele também é fundamental para a subsistência de milhões de pessoas ao redor do mundo (FAO, 2017).

Ademais, a pesca artesanal contribui de maneira decisiva para a segurança alimentar e nutricional das comunidades. Ao fornecer uma fonte estável e acessível de proteínas, ela ajuda a combater a fome e a pobreza em diversas regiões, particularmente nas áreas costeiras e ribeirinhas.

A sustentabilidade da pesca artesanal é entendida como a capacidade de manter os recursos naturais e a exploração desses recursos a longo prazo, levando em conta fatores socioeconômicos, culturais e políticos que influenciam o sucesso do processo produtivo. A integração dessas variáveis é fundamental para alcançar a sustentabilidade. Esse conceito destaca a importância de aspectos como igualdade, justiça e direitos humanos (PEDROSA; LESSA,2017).

3.2 Pesca artesanal na Região de Araioses

A pesca artesanal em Araioses é predominantemente realizada com a participação de mão de obra familiar, envolvendo 91% dos pescadores, que contam com o auxílio de filhos e cônjuges na atividade. Esta prática garante a renda e o consumo familiar, além também de facilitar a transmissão de conhecimentos pesqueiros de geração em geração, preservando tradições e modos de manejo local (SANTOS; MELO; ROCHA, 2012).

A pesca é a principal ocupação para 60% dos pescadores da região, enquanto os demais complementam sua renda com atividades secundárias, como agricultura, construção civil e comércio, setores que exigem pouca qualificação e revelam um baixo nível de escolaridade entre os trabalhadores. Apesar de sua importância, a renda gerada pela pesca é considerada insuficiente para o sustento completo das famílias, com 95% dos pescadores afirmando que o valor obtido é inadequado para cobrir todas as despesas.(Lopes et al., 2023)

3.3 Linguagens e framework

3.3.1 PHP

PHP (*Hypertext Preprocessor*) é uma linguagem de *script* do lado do servidor amplamente utilizada no desenvolvimento *web*. Desenvolvido inicialmente para criar páginas *web* dinâmicas, PHP permite a integração fluida com bancos de dados como MySQL. Sua capacidade de gerar conteúdo dinâmico e processar formulários de maneira eficiente a torna uma ferramenta essencial para desenvolvedores. Além disso, PHP é conhecido por sua facilidade de aprendizado e pela vasta gama de *frameworks* e bibliotecas disponíveis, o que contribui para acelerar o processo de desenvolvimento e melhorar a produtividade (Welling e Thomson, 2017).

Além de suas funcionalidades básicas, o PHP oferece uma variedade de recursos avançados que suportam operações complexas e escaláveis. A linguagem possui uma comunidade ativa e uma ampla documentação, o que facilita a resolução de problemas e a implementação de melhores práticas. Entretanto, apesar de suas vantagens, a segurança é uma consideração crucial, já que PHP pode estar sujeito a vulnerabilidades se não for corretamente configurado e atualizado. A adoção de práticas de codificação seguras e a utilização de ferramentas de segurança são essenciais para proteger aplicações desenvolvidas com PHP contra ameaças e ataques (Welling & Thomson, 2017).

Dessa forma, o PHP, utilizado em cerca de 78,1% dos sistemas *web*, é essencial para desenvolvedores *back-end*. Com uma grande presença na internet, o PHP é a principal linguagem do *WordPress*, plataforma amplamente adotada. Sua flexibilidade e robustez são ampliadas por *frameworks* como o Laravel, ele se destaca pela sintaxe simples e pela oferta de ferramentas que agilizam a construção

de sistemas modernos. A combinação de PHP e Laravel torna o desenvolvimento mais eficiente e produtivo. (Hostgator, 2024)

3.3.3 Laravel

O Laravel é um *framework* PHP moderno e robusto, projetado para facilitar o desenvolvimento de aplicações *web* escaláveis e seguras no *back-end*. Com sua arquitetura baseada no padrão MVC, o Laravel organiza o código de forma eficiente, o que melhora a manutenção e escalabilidade dos sistemas. Um de seus principais recursos é o *Eloquent*, que simplifica a interação com bancos de dados relacionais e permite uma manipulação eficiente dos dados, essencial em sistemas que demandam alta performance e segurança. A utilização de rotas, autenticação e *middleware* integrados torna a configuração de processos complexos mais simples, permitindo que o desenvolvedor foque no negócio sem se preocupar com detalhes de implementação (ARMEL, 2014).

Além disso, o *framework* também traz recursos como filas para processamento assíncrono e *tasks scheduling*, que são fundamentais para a criação de aplicações de larga escala. Desde seu lançamento, o Laravel tem sido uma escolha consolidada para o desenvolvimento de sistemas de forma eficiente, combinando produtividade, segurança e flexibilidade. (ARMEL, 2014).

3.3.2 Angular

Angular é um *framework* de desenvolvimento de *front-end* desenvolvido pelo Google, projetado para facilitar a criação de aplicações *web* dinâmicas e complexas. Ele adota uma arquitetura baseada em componentes, que permite a construção de interfaces de usuário modulares e reutilizáveis. Com o uso de *TypeScript*, Angular oferece uma tipagem estática que ajuda a detectar erros durante o desenvolvimento e melhora a manutenção do código (Angular, 2024).

Entre suas principais características, destaca-se o data *binding* bidirecional, uma funcionalidade essencial que mantém a interface de usuário sincronizada com o modelo de dados. Essa abordagem permite que quaisquer alterações na interface

sejam refletidas automaticamente nos dados subjacentes e vice-versa, simplificando a manutenção da coerência entre a UI e os dados.

A arquitetura modular do Angular é baseada em componentes e módulos, que facilitam a criação e gestão de partes da interface de maneira modular. Os componentes permitem a construção de interfaces reutilizáveis, enquanto os módulos organizam e encapsulam funcionalidades, promovendo uma estrutura de desenvolvimento mais escalável e organizada.

Outrossim, Angular proporciona diretivas e serviços que ampliam suas capacidades. As diretivas permitem estender o HTML com comportamentos personalizados, enquanto os serviços encapsulam a lógica de negócios e facilitam a comunicação com APIs. Isso contribui para a criação de aplicações *web* mais dinâmicas e interativas, e melhora a separação de preocupações, mantendo a lógica de negócios e a manipulação de dados separados da apresentação da interface (Hevery, 2018).

3.4 *Progressive Web Apps* (PWA)

Progressive Web Apps (PWAs) representam uma abordagem moderna para o desenvolvimento de aplicativos *web*, oferecendo uma experiência de usuário que se assemelha à de aplicativos nativos em dispositivos móveis e *desktops*. Utilizando tecnologias avançadas, os PWAs combinam o melhor da *web* e dos aplicativos móveis, proporcionando uma série de vantagens notáveis. Entre essas vantagens está a capacidade de funcionar *offline*, possibilitada pelos *Service Workers*, que permitem o armazenamento em cache de recursos e dados essenciais para o funcionamento do aplicativo, mesmo quando não há conexão com a internet (Google Developers, 2024).

Outra característica fundamental dos PWAs oferecer a vantagem de instalação direta no dispositivo do usuário a partir do navegador, sem a necessidade de passar por lojas de aplicativos. Isso é feito ao adicionar um ícone na tela inicial do dispositivo, proporcionando uma experiência semelhante à de aplicativos nativos. Essa instalação simplificada contribui para uma integração mais suave com a

plataforma do usuário e facilita o acesso ao aplicativo, melhorando a conveniência e a acessibilidade para os usuários (Nielsen, 2021).

3.6 Armazenamento de Dados - MySql

MySQL é um sistema de gerenciamento de banco de dados relacional de código aberto que se destaca por sua performance e eficiência no tratamento de grandes volumes de dados. Amplamente utilizado em conjunto com PHP, MySQL oferece uma solução robusta para armazenar e gerenciar dados de forma eficaz, suportando desde pequenas aplicações até grandes sistemas empresariais. A capacidade de realizar operações rápidas de leitura e escrita, combinada com a facilidade de escalabilidade, faz do MySQL uma escolha popular para desenvolvedores que buscam um banco de dados confiável e de alto desempenho (Kroenke & Auer, 2019).

Adicionalmente, MySQL proporciona várias funcionalidades de segurança que são essenciais para proteger dados sensíveis. Entre essas funcionalidades estão a autenticação de usuários, o controle de acesso granular e a criptografia de dados, que ajudam a garantir a integridade e a confidencialidade das informações armazenadas. A integração direta com PHP permite uma manipulação eficiente dos dados e facilita o desenvolvimento de aplicações *web* dinâmicas e seguras. Com sua flexibilidade e suporte a múltiplas plataformas, MySQL se consolidou como uma ferramenta indispensável no desenvolvimento moderno (Kroenke & Auer, 2019).

3.7 Hospedagem AWS

A hospedagem de sistemas e aplicações na nuvem tem se tornado uma prática cada vez mais comum devido à sua flexibilidade, escalabilidade e custos reduzidos. A nuvem oferece vantagens como a facilidade de escalabilidade, a redução da necessidade de manutenção física de servidores e a capacidade de atender a um público global. A computação em nuvem é descrita como um modelo de fornecimento de serviços acessíveis sob demanda, onde os usuários podem acessar e utilizar recursos pela internet, sem precisar de infraestrutura própria (ERL; PUTTINI; MAHMOOD, 2013).

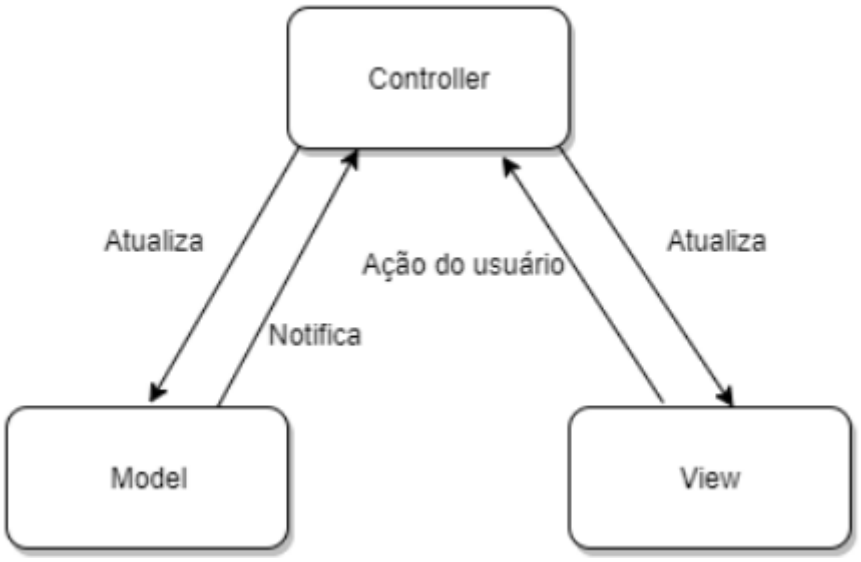
A AWS (*Amazon Web Services*) é uma plataforma de computação em nuvem amplamente adotada por sua escalabilidade e flexibilidade. O AWS *Amplify* é uma ferramenta que facilita o desenvolvimento e a hospedagem de aplicações *web* e móveis, oferecendo uma integração simples com serviços como *Amazon S3*, *Amazon CloudFront*. Ele proporciona uma solução completa para o *front-end*, permitindo que os desenvolvedores se concentrem na criação de funcionalidades sem se preocupar com a infraestrutura. (Amazon, 2024)

Para serviços *back-end*, o *Amazon* EC2 (*Elastic Compute Cloud*) oferece instâncias de servidores virtuais altamente escaláveis, adaptáveis às necessidades específicas de cada aplicação.O uso combinado do AWS *Amplify* e EC2 cria uma arquitetura de sistema eficiente, onde cada parte pode ser escalada de maneira independente. A flexibilidade do AWS permite que o sistema se ajuste automaticamente à demanda, melhorando a performance e reduzindo custos operacionais. (Amazon, 2024)

3.8 Padrão de Arquitetura de Software (MVC)

O padrão arquitetural MVC é amplamente utilizado em aplicações *web*, sendo responsável por organizar a estrutura da aplicação em três componentes principais: a camada de negócios (*Model*), a camada de controle (*Controller*) e a camada de visualização (*View*). De acordo com *Fowler* et al. (2006), cada uma dessas camadas tem a responsabilidade de gerenciar de maneira independente tanto a lógica de negócios quanto a interação com o usuário. Dessa forma, o MVC promove uma separação clara de responsabilidades, facilitando a manutenção e a escalabilidade das aplicações.

Figura 1: Arquitetura MVC



Fonte: (Adriely, 2022).

A camada *Model* é onde reside a maior parte da lógica de negócios da aplicação. Ela também interage diretamente com o banco de dados ou outras fontes de dados, sendo a camada que contém os processos centrais que governam o funcionamento do sistema. Segundo Richards e Ford (2024), o *Model* lida com a lógica de negócios, garantindo que a integridade dos dados e as operações necessárias sejam realizadas corretamente, sem envolver aspectos de apresentação ou controle de fluxo.

Por sua vez, a *View* é a camada responsável pela interface com o usuário. Ela recebe os dados processados pelo *Model* e os exibe de forma compreensível e visualmente agradável. A *View* não deve conter lógica de negócios ou manipulação de dados, seu papel é exclusivamente a apresentação.Nesse contexto a *View* é projetada para exibir os dados ao usuário de forma clara, focando na interação e usabilidade da aplicação, sem se envolver diretamente com o processamento dos dados (Sommerville, 2011).

O *Controller* atua como intermediário entre a *View* e o *Model*. Ele recebe as ações do usuário, processa essas informações e decide qual *Model* deve ser utilizado para manipular os dados, além de determinar qual *View* deve ser renderizada.O *Controller* é essencial para o funcionamento da aplicação, garantindo

que as interações do usuário sejam corretamente processadas e que a interface de usuário seja atualizada de acordo.(Fowler, 2006)

O padrão MVC, diagramado na figura 1, oferece várias vantagens no desenvolvimento de sistemas modernos. Sua principal vantagem é a separação clara de responsabilidades, o que facilita a manutenção e a evolução das aplicações. Com as camadas bem definidas, os desenvolvedores podem focar em aspectos específicos do sistema sem afetar outras áreas. Isso torna a arquitetura mais escalável, permitindo que novas funcionalidades sejam adicionadas com menos risco de afetar a estabilidade do sistema (Adriely, 2022).

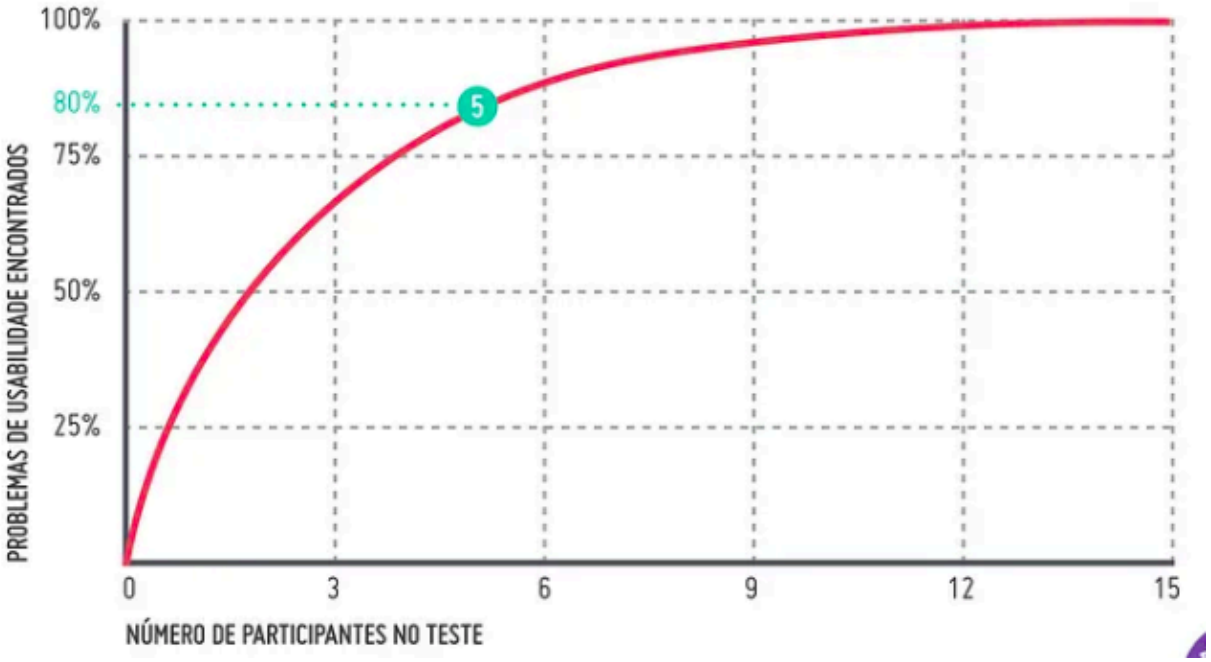
Outro benefício importante é a facilidade de manutenção. Mudanças na interface do usuário podem ser feitas sem impactar a lógica de negócios, e o *model* pode ser alterado sem a necessidade de modificar a *view*. Isso proporciona maior flexibilidade e reduz o tempo de desenvolvimento.

3.8 Método de avaliação do *software*

A avaliação de sistemas de *software* é essencial no desenvolvimento de aplicativos, especialmente para interfaces de usuário. Segundo Volpato (2017), é possível identificar até 80% dos problemas de usabilidade com apenas 5 participantes. Isso ocorre porque os primeiros participantes geralmente revelam os problemas mais comuns, e com o aumento do número de participantes, esses problemas começam a se repetir, proporcionando um aprendizado significativo com uma amostra reduzida, conforme o gráfico da figura 2.

No entanto, o número de participantes pode variar dependendo da complexidade da tarefa e dos perfis dos usuários testados. Quando há diferentes perfis de usuário ou tarefas mais complexas, pode ser necessário incluir mais participantes para cobrir as variações no comportamento.

FIgura 2: Número de participantes de um teste



Fonte: (Volpato,2017).

Nesse contexto, a pesquisa qualitativa e quantitativa são duas abordagens essenciais na avaliação de sistemas. A pesquisa quantitativa coleta dados objetivos e numéricos, como o tempo de uso ou a taxa de sucesso nas interações, permitindo uma análise estatística sobre o comportamento dos usuários. Já a pesquisa qualitativa se concentra nas percepções e experiências subjetivas, buscando entender os sentimentos e motivações dos usuários.

4 METODOLOGIA

O presente capítulo descreve a abordagem metodológica adotada para o desenvolvimento. O objetivo é explicar o processo de desenvolvimento, desde as especificações das regras de negócios na seção de requisitos funcionais até a modelagem e desenvolvimento do sistema, tanto do *back-end* quanto do *front-end*.

4.1 Atores

- **Administrador**

  O Administrador é o responsável pela gestão geral do sistema. Ele pode cadastrar novos usuários, redefinir senhas, gerenciar dados financeiros e gerar relatórios. Além disso, o administrador tem acesso completo a todos os registros de pesca e pode filtrar dados necessários.

- **Colaborador (Pescador)**

  O colaborador é um pescador que usa o sistema para registrar suas atividades de pesca e visualizar seus próprios registros. Ele pode adicionar novos registros, consultar informações sobre suas capturas e atualizar seus dados pessoais.

4.2 Requisitos funcionais

Os requisitos funcionais definem as operações e capacidades que cada tipo de usuário deve ter dentro do sistema. Eles detalham as funcionalidades específicas que o sistema deve oferecer para cumprir com as necessidades dos usuários (Sommerville, 2011).

Cadastro de Usuário (RF1): O sistema deve permitir o cadastro de novos usuários, que podem ser administradores da cooperativa ou colaboradores pescadores. O administrador será responsável por inserir e gerenciar os dados dos colaboradores no sistema. Este processo é essencial para garantir que todos os usuários possam acessar as funcionalidades apropriadas ao seu papel.

Redefinição de Senha (RF2) : O administrador pode redefinir a senha de qualquer colaborador, gerando uma senha temporária. Após o login com a senha temporária, o colaborador pode alterar a senha para uma de sua escolha.

Autenticação (RF3):O sistema deve fornecer uma tela de login onde os usuários inserem suas credenciais (CPF e senha). O acesso será concedido de acordo com o papel do usuário, seja administrador ou colaborador. Isso garante que cada usuário tenha acesso apenas às funcionalidades pertinentes ao seu papel.

Edição de Dados Pessoais (RF4): Usuários poderão atualizar seus dados pessoais, como endereço e telefone, dentro da área logada no sistema. Isso permite que os usuários mantenham suas informações atualizadas, facilitando a comunicação e o gerenciamento das informações pessoais.

Adição de Registro de Pesca (RF5) : Colaboradores poderão adicionar registros de pesca, incluindo detalhes como quantidade , data e local da captura. Este recurso é fundamental para manter um banco de dados preciso e atualizado sobre as atividades de pesca.

Listagem de Registro de Pesca (RF6): Colaboradores poderão visualizar apenas seus próprios registros de pesca, enquanto administradores poderão acessar todos os registros, filtrando por pescador e data. Isso facilita o acompanhamento das atividades pesqueiras.

Detalhamento do Registro de Pesca (RF7): O sistema deve permitir a visualização detalhada de cada registro de pesca, mostrando todos os dados inseridos. Isso possibilita uma análise completa dos registros, garantindo transparência e controle sobre as informações de captura.

Adição de controle financeiro (RF8): Administradores poderão cadastrar itens específicos na área financeira, como despesas e receitas relacionadas à pesca. Este recurso é essencial para monitorar e gerenciar a situação financeira da cooperativa.

Listagem de Administradores e Colaboradores (RF9): O administrador pode visualizar uma lista de todos os administradores e colaboradores cadastrados

no sistema. Isso ajuda no gerenciamento eficaz da equipe e facilita a coordenação das atividades.

Alteração de senha de Usuário (RF10) Administradores podem alterar a senha de qualquer colaborador, redefinindo-a para uma senha temporária, e permitindo que o colaborador a modifique posteriormente. Este recurso é crucial para manter a segurança das contas de usuário.

Geração de Relatórios de Pesca (RF11): O sistema deve gerar relatórios detalhados sobre as atividades de pesca, como volume de captura e custos. Isso fornece uma visão quantitativa e analítica das operações, auxiliando na tomada de decisões e na avaliação de desempenho.

A Tabela abaixo representa as permissões de cada funcionalidade para determinado ator do sistema.

Quadro 1: Permissões dos usuários

	Administrador	Colaborador
Redefinir Senha	**Sim**	**Não**
Auteticado	**Sim**	**Sim**
Editar Dados	**Sim**	**Sim**
Adicionar Pesca	**Não**	**Sim**
Listar Pesca	**Sim**	**Sim**
Detalhar Registro	**Sim**	**Sim**
Adicionar Financeiro	**Sim**	**Não**
Listar Usuários	**Sim**	**Não**
Alterar Senha	**Sim**	**Não**
Gerar Relatórios	**Sim**	**Não**

Fonte: Autor próprio.

4.3 Requisitos não funcionais

Requisitos não funcionais especificam critérios que julgam o desempenho, a usabilidade, a segurança e outras qualidades do sistema, esses requisitos determinam como o sistema deve operar e garantem que ele atenda a certos padrões e requisitos de qualidade (Sommerville ,2011).

Acessos simultâneos (RNF1): O sistema deve poder ser acessado por várias pessoas simultaneamente. Para que os usuários possam acessar e atualizar dados de forma eficiente e sem atrasos significativos.

Usabilidade (RNF2): A interface do sistema deve ser intuitiva e fácil de usar, com suporte para navegação clara e acessível para todos os usuários, incluindo usuários com pouca experiência em tecnologia.

Segurança (RNF3): O sistema deve implementar criptografia para proteger dados sensíveis. Além disso, deve ter mecanismos de autenticação para garantir que apenas usuários autorizados tenham acesso às funcionalidades apropriadas.

Disponibilidade (RNF4): O sistema deve ter uma disponibilidade mínima de 99% durante o horário comercial, garantindo que esteja acessível para os usuários a maior parte do tempo.

Escalabilidade (RNF5): O aplicativo deve ser escalável para suportar um aumento no número de usuários e dados sem degradação significativa no desempenho.

Compatibilidade dispositivos móveis (RNF6): O sistema deve ser compatível com os principais navegadores da web e dispositivos móveis, sem problemas de funcionalidade.

4.4 Modelagem do software

A modelagem do *software* define a estrutura e os componentes necessários para atender aos requisitos do sistema. Utilizando diagramas de fluxo e arquitetura em camadas, detalha a interação entre *front-end* e *back-end*, assegurando comunicação eficiente e segura. A modelagem também considera escalabilidade e manutenção, permitindo ajustes futuros.

4.4.1 Caso de Uso

Essa seção abordará o Diagrama de Caso de Uso, que é uma modelagem em UML que visa representar as funcionalidades observáveis do sistema, bem como os elementos externos que interagem com ele. Ele oferece uma visão mais geral das diferentes maneiras pelas quais os usuários, ou outros sistemas, podem interagir e se beneficiar das funcionalidades oferecidas pelo sistema em questão.

Essencialmente, o diagrama de caso de uso destaca os diversos cenários de utilização, identificando os atores, que são representações de elementos externos que interagem com o sistema durante sua execução (Bezerra, 2015). Esses atores podem ser usuários ou entidades externas, e o relacionamento entre atores e caso de uso.

Segundo a aplicação Lucidchart, o diagrama de caso de uso deve seguir três principais pontos (Lucidchart, 2023):

1. Cenários onde o sistema ou aplicativo interage com pessoas, organizações ou sistemas externos.
2. Metas que o sistema ou aplicativo ajuda essas entidades (conhecidas como atores) a atingir.
3. O escopo do sistema.

Figura 3 : Diagrama de casos de uso

Fonte: Autor próprio.

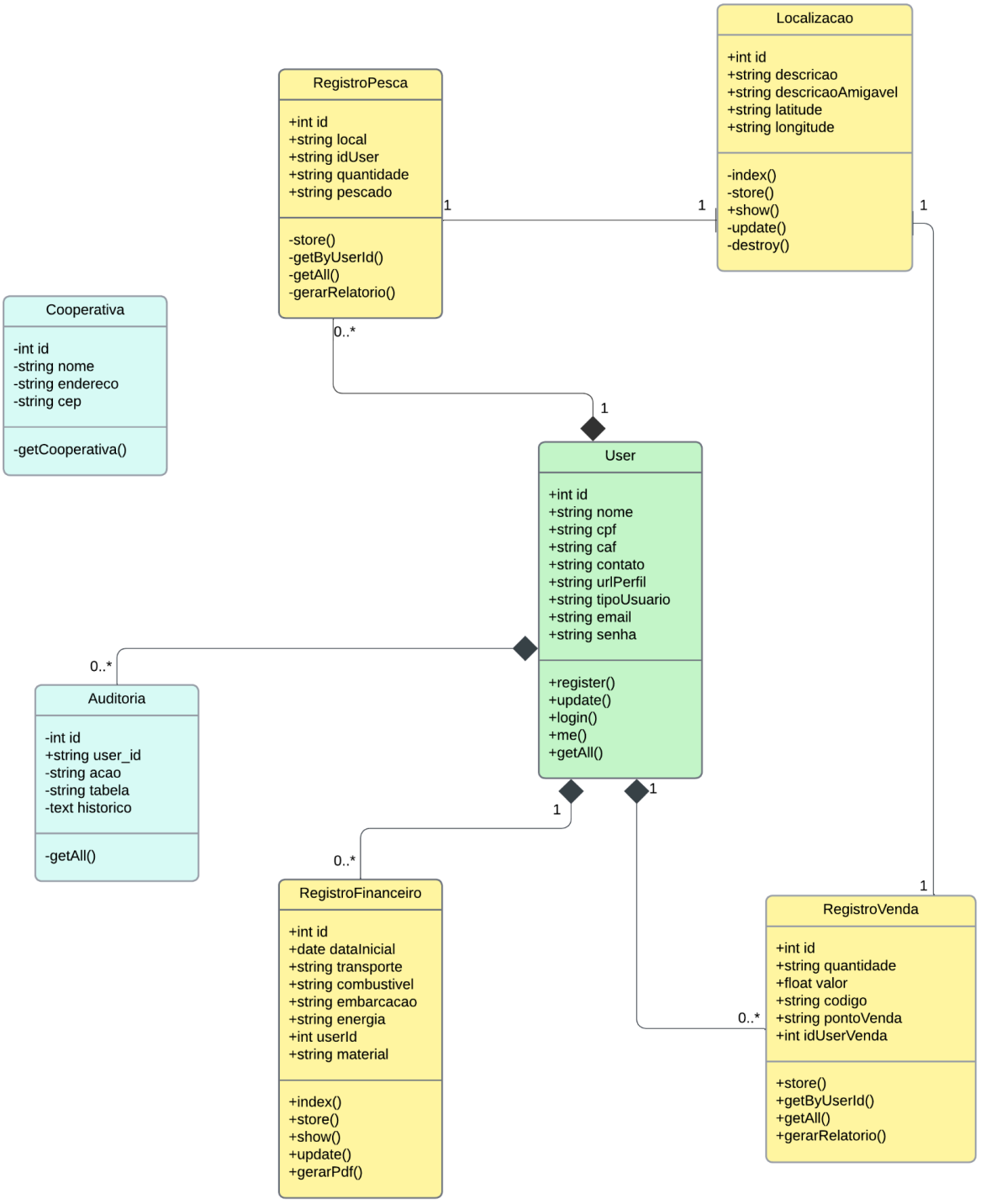
O diagrama ilustrado acima representa as funcionalidades do sistema, destacando as ações dos atores. O administrador possui acesso a funções como gerenciamento de usuários, localidades e registros financeiros, além de ações gerais como *login*, *logout* e edição de perfil. O colaborador, por sua vez, pode cadastrar o pescado e vendas, visualizar detalhes dessas atividades e também gerenciar seu perfil. Por meio deste diagrama mostra claramente as permissões de cada ator e como interagem com o sistema.

4.4.2 Diagrama de Classes

Esta parte do trabalho apresenta a estrutura e os relacionamentos das classes, abordando a modelagem da organização e das interações entre as classes do sistema. O diagrama de classes é uma representação visual essencial para sistemas orientados a objetos, pois descreve como as entidades do sistema se conectam e se comportam.

A partir dessa modelagem, é possível entender melhor a distribuição das responsabilidades no sistema e como os dados e comportamentos estão interligados .O diagrama de classes é uma das ferramentas mais importantes durante o desenvolvimento de sistemas, pois fornece uma visão clara das entidades, seus atributos e os métodos que elas podem executar.

Figura 4: Diagrama de classes



Fonte: Autor próprio.

A imagem 4 representa o diagrama de classes, a entidade User representa os usuários do sistema, armazenando informações como id, nome, email, senha e telefone. Essa classe é central e conecta-se a outras funcionalidades importantes, como *RegistroPesca, RegistroVenda* e *RegistroFinanceiro*, indicando que um usuário pode realizar múltiplas pescas, vendas e registros financeiros (relação 1 para N). Os métodos da classe *User* incluem operações como re*gistrar(), atualizar() e excluir(),* permitindo a gestão de dados dos usuários no sistema. Essa centralidade reflete o papel essencial dos usuários na operação do sistema.

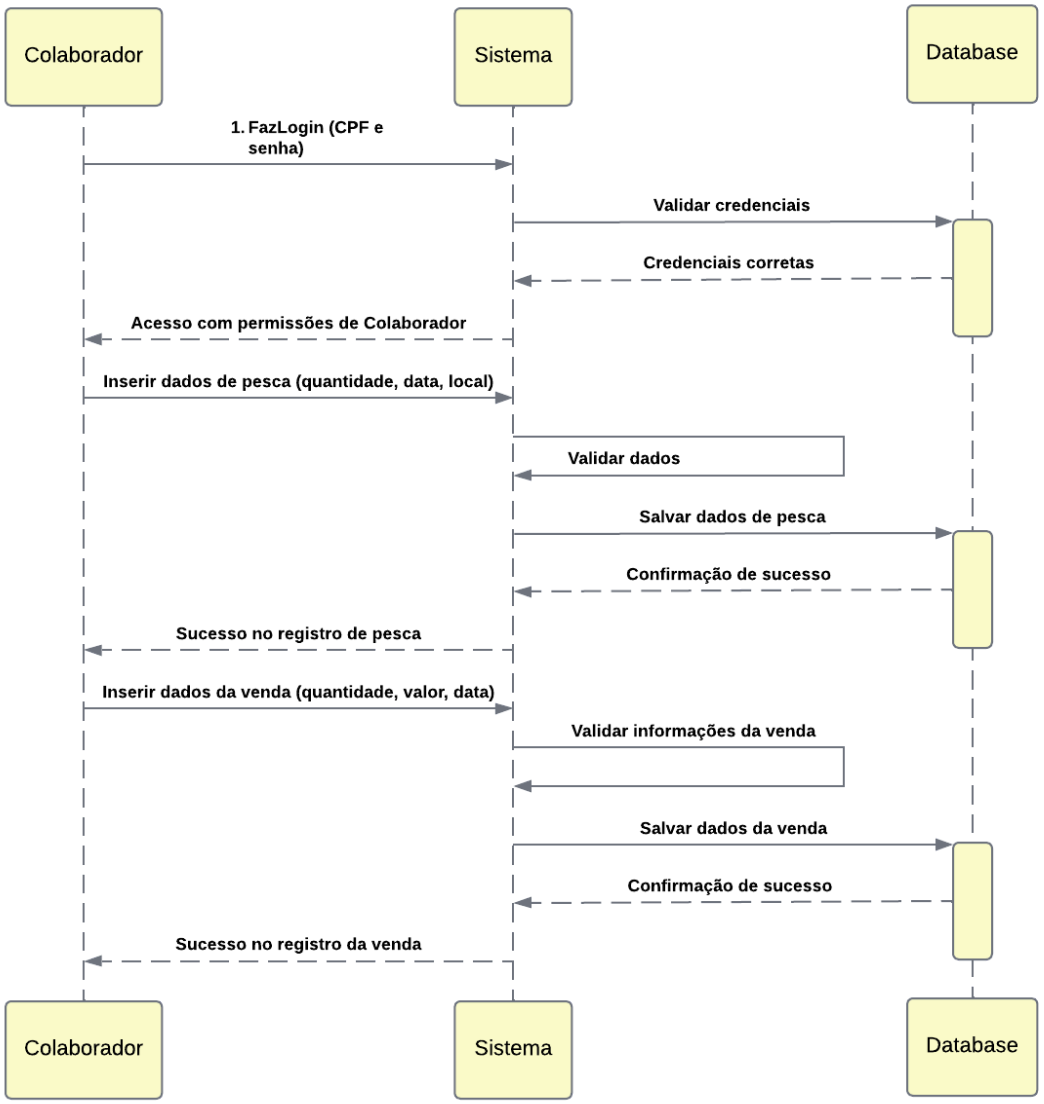
4.4.3 Diagrama de Sequência

Esta seção aborda os Diagramas de Sequência, que ilustram a ordem cronológica das interações entre objetos durante um processo específico. Cada objeto é representado por uma linha de vida, e as mensagens trocadas entre esses objetos são indicadas por setas, que mostram a direção da comunicação. Esses diagramas fornecem uma visão dinâmica e interativa do comportamento do sistema, destacando as interações entre suas diferentes partes.

Participantes:

- **Usuário Admin ou colaborador**: O usuário pode realizar diversas ações relacionadas à consulta ou registro de pesca e venda.
- **Sistema**: A aplicação que gerencia as interações entre os usuários e o banco de dados e as operações relacionadas.
- **Banco de Dados**: Armazena as informações referentes aos usuários, registro de vendas , pescas e localizações .

Figura 5: Diagrama de sequência do colaborador cadastrando pesca/venda.



Fonte: Autor próprio.

4.5 Convenções de Nomenclatura

Para garantir um código bem estruturado e fácil entendimento, foram adotadas convenções de nomenclatura para o projeto. No desenvolvimento, as classes e componentes seguem a convenção *PascalCase*, na qual cada palavra

começa com uma letra maiúscula, sem espaços ou sublinhados (Martin,2008). Por exemplo, no projeto, uma classe para o perfil do usuário é nomeada *PerfilUsuarioComponente*, e um componente para o painel do administrador é denominado *PainelAdministradorComponente*, conforme o quadro 2 demonstra.

Quadro 2: Convenção de nomenclatura *PascalCase*

Tipo	Nome de Exemplo
Classe	PerfilUsuario Componente
Componente	PainelAdministradorComponente

Fonte: Autor próprio.

As variáveis e funções utilizam a convenção *camelCase*, onde a primeira palavra começa com uma letra minúscula e cada palavra subsequente começa com uma letra maiúscula, sem espaços ou sublinhados. O objetivo de usar essa convenção é deixar um código mais limpo e intuitivo (Martin,2008). O quadro abaixo exemplifica o uso da convenção *camelCase*.

Quadro 3: Convenção de nomenclatura *camelCase*

Tipo	Nome de Exemplo
Variável	perfilUsuario
Função	buscarDadosUsuario()

Fonte: Autor próprio.

4.6 Desenvolvimento do software

Para o desenvolvimento da aplicação, foi configurado um ambiente de desenvolvimento local utilizando o sistema operacional Ubuntu, uma distribuição popular do Linux. O primeiro passo consistiu na instalação e configuração das ferramentas necessárias para o funcionamento do projeto. Foram instalados o Angular, utilizando o Node *Package Manager* de forma global, o PHP 8.2, e o Composer para gerenciar as dependências do Laravel. Além disso, foi configurado o banco de dados MySQL, juntamente com a ferramenta DBBeaver, responsável pela análise e visualização das consultas ao banco, auxiliando no desenvolvimento .

4.6.1 *Front-End*

No desenvolvimento do *front-end*, adotaram-se práticas específicas para garantir a modularidade e a reusabilidade do código. Essa abordagem facilita a manutenção e a escalabilidade do sistema, além de otimizar os testes do aplicativo e a geração do PWA tornando esses processos mais eficientes

O *front-end* é estruturado em dois principais componentes: *pages* e *shared*.O componente *pages* é responsável por abrigar as diferentes páginas da aplicação, organizadas de acordo com os papéis dos usuários, como administrador e colaborador. Cada tipo de usuário possui suas próprias páginas e funcionalidades específicas, garantindo uma experiência personalizada. Por exemplo, as páginas para administradores podem incluir funcionalidades administrativas adicionais como cadastro de novas localizações e listagem de todos os usuários, enquanto as páginas para colaboradores são adaptadas para suas necessidades operacionais.

Por outro lado, o componente *shared* armazena elementos que são reutilizáveis em diversas partes da aplicação. Componentes como o menu de navegação, o cabeçalho das páginas. Esses componentes são utilizados por diferentes páginas, tendo uma consistência visual e funcional. Outra parte importante são os serviços que encapsulam a lógica de comunicação com o *back-end*, facilitando a integração e a manutenção do código. Serviços como cadastro de pesca são responsáveis por realizar operações de CRUD e gerenciar a troca de dados entre o *front-end* e o *back-end.*

4.6.2 *Back-end*

Para o *back-end* foi desenvolvido utilizando o *framework* Laravel, aproveitando sua estrutura para a criação da lógica de negócios e comunicação com o banco de dados. O banco de dados foi modelado utilizando *migrations* do Laravel, o que permite uma gestão controlada e incremental das tabelas e esquemas. O uso das migrações auxilia na evolução do esquema do banco de dados de forma estruturada, mantendo um histórico claro das alterações realizadas.

Os *endpoints* da API foram projetados seguindo o formato REST *(Representational State Transfer*), a estrutura REST dos *endpoints* permite a

realização de operações de criar, ler, atualizar e deletar de maneira organizada e facilita a integração entre as camadas da aplicação.

Para a segurança, foi utilizado o Laravel Sanctum, que permite o gerenciamento dos tokens de autenticação, garantindo que apenas usuários devidamente autenticados possam acessar recursos sensíveis da aplicação.

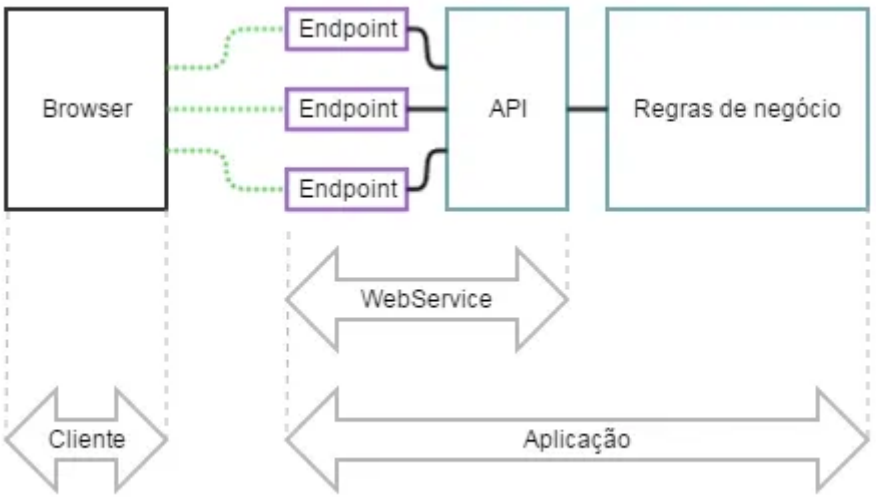
4.6.3 Integração back-to-front

No desenvolvimento da aplicação, foi implementado um sistema de comunicação entre o *front-end* e o *back-end* utilizando APIs (Interfaces de Programação de Aplicações). Diversos *endpoints* foram criados, cada um representando uma funcionalidade específica do sistema, como operações de recuperação e envio de dados. O apêndice A abaixo demonstra algum dos *end points* mais importantes do sistema.

O *front-end faz* requisições HTTP para esses *endpoints*, por meio dos serviços e permite que o *back-end* execute as operações necessárias com base nas regras de negócios definidas. O *back-end* foi configurado para processar essas requisições e retornar respostas no formato JSON. As respostas geradas são então recebidas pelo *front-end*, que as utiliza para atualizar a interface do usuário de forma dinâmica e interativa.

A integração entre o *front-end* e o *back-end* neste sistema segue o padrão da arquitetura MVC , onde a comunicação entre as camadas é claramente definida. Os Controllers são responsáveis por gerenciar as requisições HTTP feitas pelo *front-end,* interagir com os *models* para realizar as operações de negócios (como registro e atualização de dados), e devolver as respostas no formato JSON. Essas respostas são consumidas pela *view*, que atualiza a interface do usuário de forma dinâmica e interativa A figura 6 representa a comunicação desse processo

Figura 6 : Estrutura de uso de uma aplicação web



Fonte:(Alan Vasconcellos,2021).

4.7 Hospedagem e configuração

Para hospedar o *back-end* da aplicação, foi utilizada uma instância Amazon EC2 configurada com o sistema operacional Ubuntu. Nessa máquina, foram instalados o servidor web Apache2, o interpretador PHP e o sistema de gerenciamento de banco de dados MySQL, estabelecendo um ambiente LAMP (Linux, Apache, MySQL, PHP) robusto para o processamento das operações do sistema. A instância EC2 forneceu um endereço IP público, permitindo o acesso externo aos serviços hospedados.

Para gerenciar o tráfego de rede e facilitar o acesso ao sistema, o Amazon Route 53 foi configurado como serviço de DNS. O Route 53 permitiu a criação de um domínio personalizado que aponta para o endereço IP da instância EC2, assegurando que os usuários possam acessar o sistema profissional.

O *front-end* da aplicação foi implantado utilizando o AWS *Amplify*, um serviço que simplifica o desenvolvimento de aplicações web modernas. O *Amplify* oferece integração contínua e entrega (CI/CD), permitindo que atualizações no código sejam automaticamente implantadas, garantindo agilidade no desenvolvimento e manutenção da interface do usuário.

5 Resultados

Este capítulo foi dividido em duas seções, uma que mostra as principais telas desenvolvidas para o funcionamento do sistema e a outra, uma avaliação do sistema. O projeto desenvolvido tem como principal objetivo otimizar o processo de gestão da atividade pesqueira na região e obter informações cruciais para poder contribuir com a sustentabilidade do local.

5.1 telas do sistema

O sistema possui duas seções, sendo a principal, a seção do administrador onde o mesmo poderá realizar consultas para verificar o que foi pescando ou vendido em determinada localização, registro financeiro e ter uma visão geral das informações dos participantes da plataforma. A seção do colaborar é para o cadastro de informações sobre venda ou pesca realizadas pelo mesmo.

5.1.1 Tela de cadastrar usuário

Para realizar o cadastro, o usuário deverá fornecer os seguintes dados obrigatórios: nome, CPF, e-mail, CAP e número de contato (Figura 7) . Após o preenchimento de todos os campos, o sistema realizará uma validação das informações fornecidas. Caso o cadastro seja realizado com sucesso, o usuário será redirecionado para a tela de login.

Figura 7: tela de cadastro

Fonte: Autor próprio.

É importante destacar que, por padrão, o sistema considera o usuário recém-criado como colaborador. No primeiro acesso, a senha será automaticamente definida como o próprio CPF do usuário. Após esse primeiro login, o sistema encaminhará o usuário para uma tela de alteração de senha, sendo esta modificação obrigatória antes de prosseguir para o uso normal da plataforma.

Essa abordagem foi adotada com o objetivo de garantir flexibilidade tanto para que os usuários possam criar seu perfil de forma autônoma quanto para permitir que administradores cadastrem novos usuários e os repassem às credenciais de acesso necessárias.

5.1.2 Tela de login

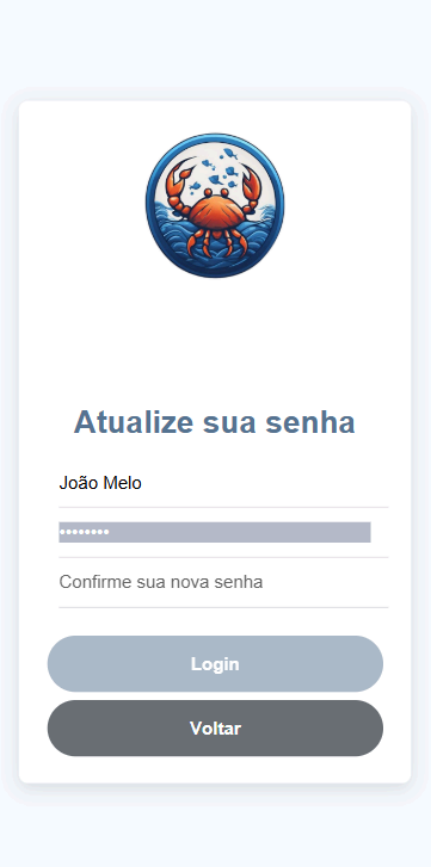
Ao acessar o sistema, o usuário deve informar seu CPF e senha na tela de login (Figura 8) . No caso de ser o primeiro acesso, a senha será automaticamente definida como o próprio CPF informado durante o cadastro.

Figura 8: tela de login

Fonte: Autor próprio.

Após essa autenticação inicial, o usuário será redirecionado para uma tela de alteração de senha (Figura 9) , onde deverá criar uma nova senha antes de prosseguir para o uso do sistema. Esse procedimento visa garantir a segurança das credenciais do usuário. Para acessos subsequentes, o usuário pode utilizar a senha definida na tela de alteração.

Figura 9: tela de atualizar senha



Fonte: Autor próprio.

O sistema identifica automaticamente o perfil do usuário. Caso o usuário seja um administrador, ele será direcionado para a área administrativa, enquanto os

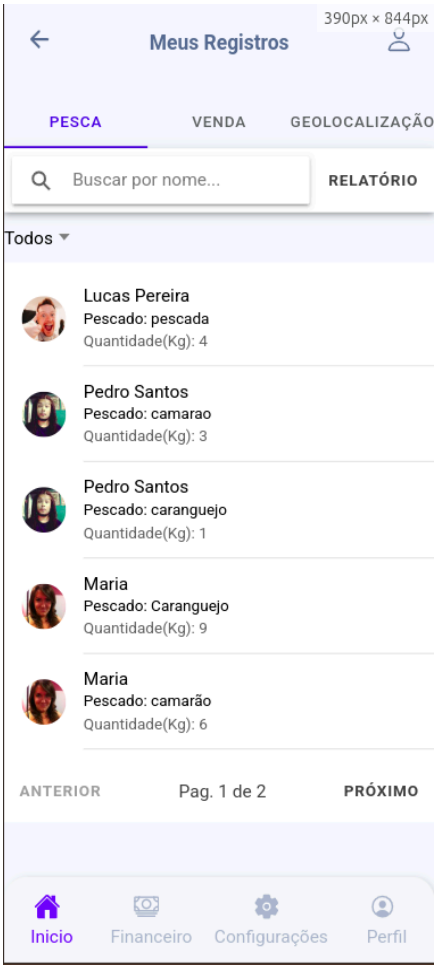
colaboradores terão acesso às funcionalidades padrão, de acordo com as permissões definidas em seu perfil.

5.1.3 tela inicial do administrador

Ao fazer login no sistema, o administrador é direcionado para a tela de pesca (Figura 10), onde pode acessar um registro completo de todas as capturas realizadas. Nesta tela,como se pode ver na captura de tela abaixo , ele pode visualizar detalhes como data, hora, tipo e quantidade de peixe, além de quem realizou a captura.

Da mesma forma, ele também tem acesso a um registro de todas as vendas realizadas. As duas telas, de pesca e venda, possuem funcionalidades de filtro, permitindo ao administrador refinar sua pesquisa por nome de usuário ou região, facilitando a análise e o acompanhamento de ambas as atividades.

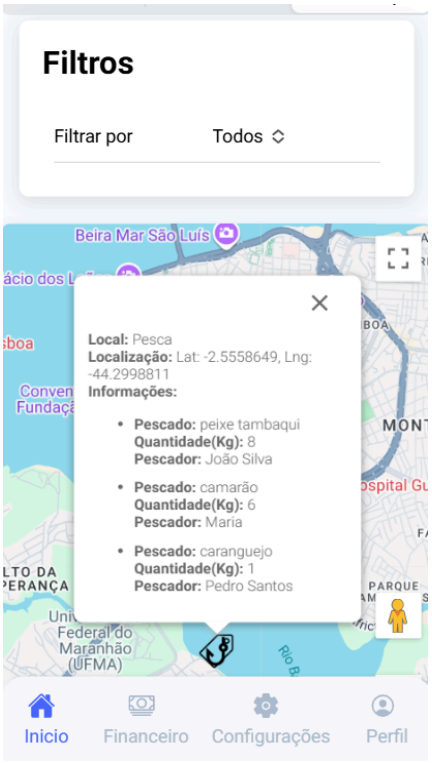
Figura 10 : tela inicial do administrador



Fonte: Autor próprio.

Além disso, o sistema oferece uma funcionalidade de geolocalização, permitindo ao administrador mapear e visualizar, em tempo real, a localização exata das pescas e das vendas. Através de um mapa interativo (Figura 11), é possível identificar as regiões onde as atividades acontecem. Isso ajuda o administrador a monitorar as áreas de maior concentração de pesca e venda, otimizando a gestão e ajudando na tomada de decisões estratégicas.

Figura 11 : Tela do mapeamento da pesca/venda



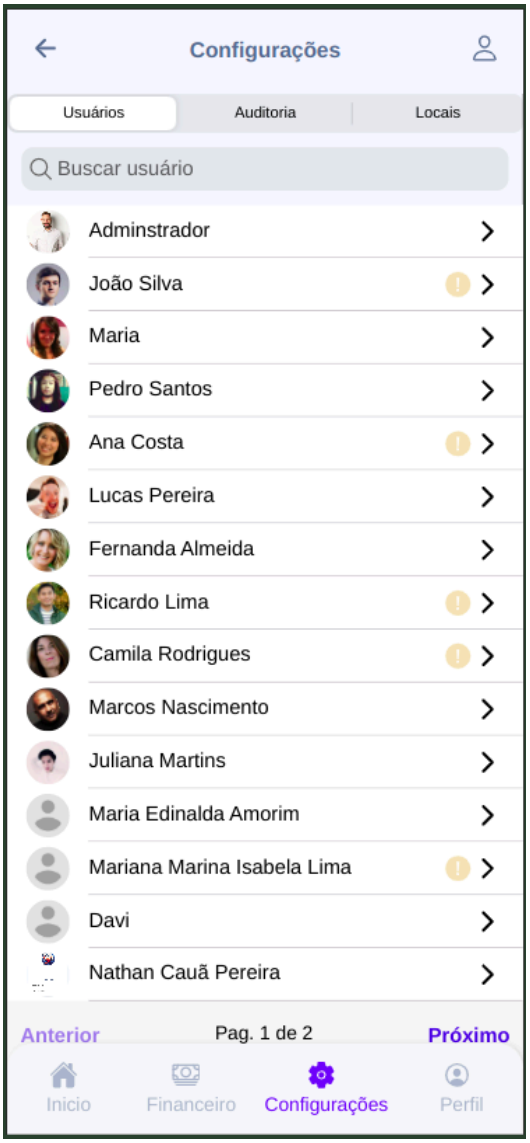
Fonte: Autor próprio.

5.1.4 tela de configurações do administrador

A tela de configurações do administrador, exibida na figura 12, tem como objetivo proporcionar ao usuário responsável pelo gerenciamento do sistema uma visão geral do sistema. Um de seus recursos é a possibilidade de gerenciar a lista de usuários registrados no sistema.

O administrador tem acesso a todas as contas de usuários e pode realizar alterações importantes, como a redefinição de senhas em caso de esquecimento ou necessidade de segurança adicional. Além disso, o administrador tem a autoridade para alterar o nível de acesso dos usuários. Por exemplo, ele pode promover um usuário de colaborador para administrador, atribuindo-lhe permissões completas para gerir o sistema, ou, inversamente, limitar um administrador a uma função mais restrita.

Figura 12: Tela de configurações



Fonte: Autor próprio.

Outro aspecto da tela de configurações é o gerenciamento da localização. O administrador tem a capacidade de editar as informações de locais já cadastrados, como pontos de venda e pontos de pesca, ou mesmo criar novos locais, expandindo o alcance do sistema para diferentes regiões.Além do controle de usuários e locais, a tela de configurações também tem uma área de auditoria essa funcionalidade tem como objetivo registrar novos login feito por usuários um histórico de atualizações no perfil do usuários, como no caso de resetar uma senha.

5.1.6 tela inicial colaborador: cadastro de pesca

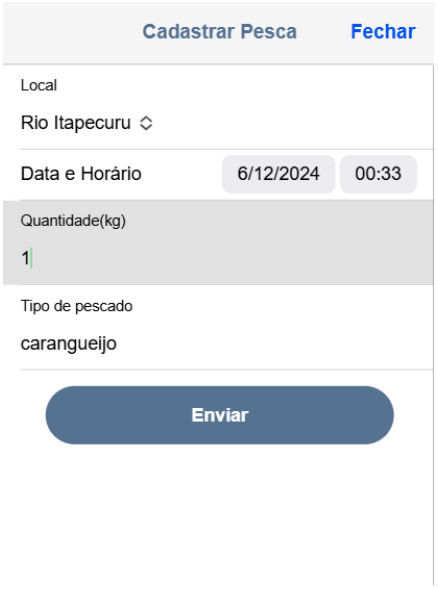
Após o login, a interface do menu para dispositivos móveis está localizada no canto inferior, apresentando as opções Pesca, Venda e Perfil. Ao iniciar o aplicativo ou realizar o login, o usuário será direcionado automaticamente para a tela de "Pesca". Nesta tela, o usuário poderá visualizar os registros das capturas cadastradas, organizados de acordo com a data e hora, sendo exibidos em ordem decrescente conforme ilustrado na imagem abaixo.

Figura 13: Tela Inicial do colaborador

Ao clicar na opção Novo, o usuário tem a possibilidade de cadastrar uma nova pesca. Para isso, uma tela de cadastro é aberta (figura 14), permitindo que ele

preencha os campos obrigatórios, como Local, Data e Hora, Quantidade e Tipo de Pescado. Após preencher todos os campos corretamente, o usuário deve clicar no botão Enviar. Caso os dados sejam salvos com sucesso, o sistema redireciona automaticamente o usuário de volta para a tela de registros, onde a nova pesca cadastrada será exibida na listagem.

Figura 14: Cadastro de pesca

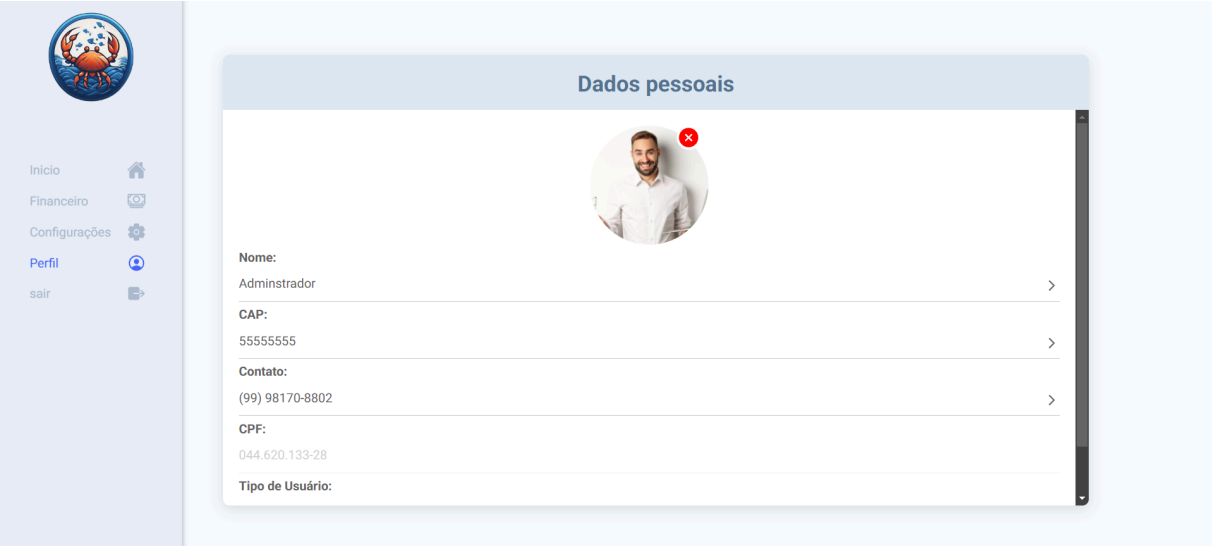


Fonte: Autor próprio.

5.1.7 tela meu perfil

A tela Meu Perfil ,na figura abaixo ,é comum para ambos os tipos de usuários, oferecendo a possibilidade de visualizar e gerenciar informações pessoais. Nela, o usuário pode verificar seus dados cadastrais e editar sua foto de perfil, caso deseje atualizar a imagem associada à sua conta. A interface permite um fácil acesso às opções de edição conforme mostrado na tela abaixo.

Figura 15: Tela informações pessoais



Fonte: Autor próprio.

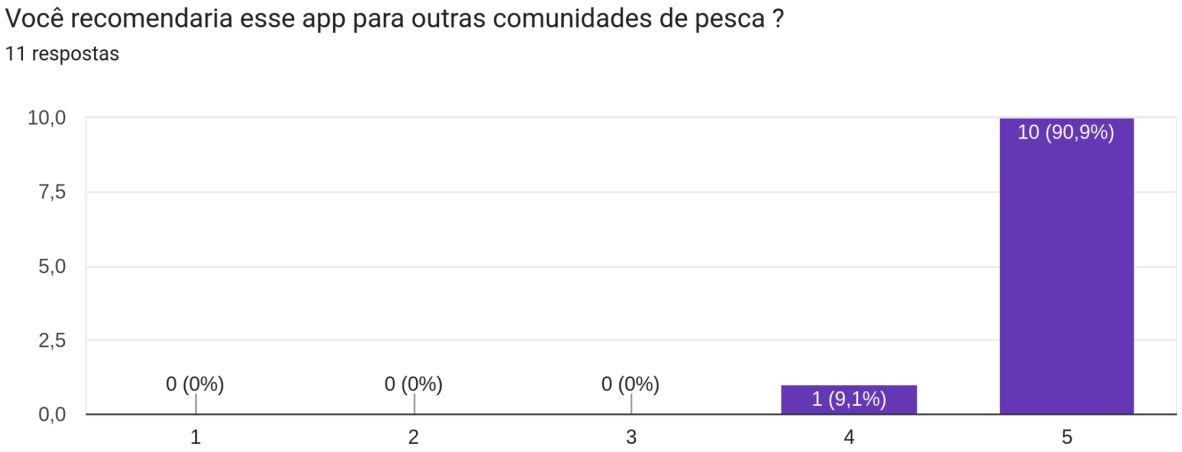
5.2 Avaliação do sistema

Para avaliar o sistema COOPDELTA, foi utilizado um google formulário com 6 perguntas divididas em duas partes: 3 quantitativas e 3 qualitativas. Os participantes da pesquisa foram compostos pela própria comunidade de pescadores e desenvolvedores, com o objetivo de avaliar a aderência do aplicativo às suas necessidades reais.

Após interagirem com o aplicativo, os participantes responderam ao questionário de pós-avaliação. Os resultados obtidos serão apresentados por meio de gráficos, facilitando a visualização das respostas e a análise de dados. Com base nas respostas, será possível verificar a satisfação geral dos usuários, os pontos fortes do COOPDELTA e eventuais áreas que demandam ajustes.

A seguir, a figura 16 apresenta os resultados da pesquisa, que indicaram que 90,9% dos participantes recomendam o aplicativo para outras comunidades de pesca. Esse dado evidencia uma aceitação inicial da ferramenta. Vale ressaltar que apenas um participante atribuiu a nota 4.

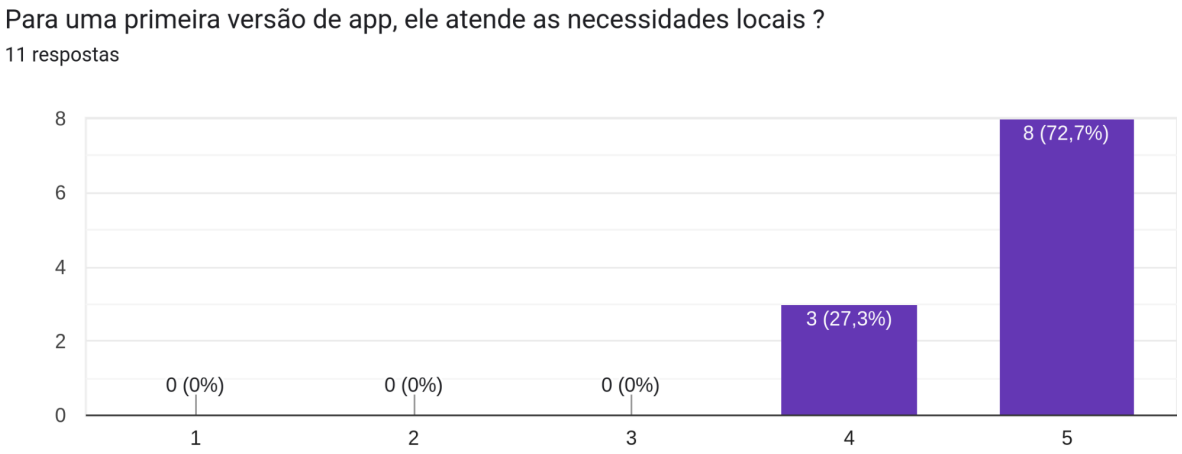
Figura 16 - Você recomendaria esse app para outras comunidades de pesca?



Fonte: Autor próprio.

Conforme ilustrado na figura 17 ,quando questionados se o aplicativo atende às necessidades locais, 72,7% dos usuários atribuíram a nota máxima. Outros 27,3% deram nota 4.

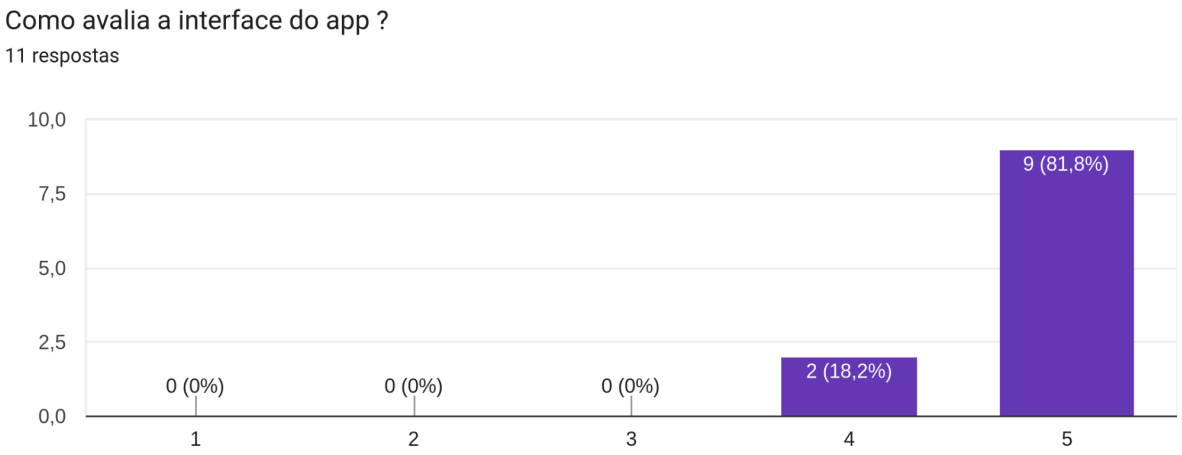
Figura 17 - Para a primeira versão de ele atende as necessidades locais?



Fonte: Autor próprio.

Conforme a figura 18, a interface do CoopDelta recebeu uma avaliação predominantemente positiva, com 81,8% dos usuários dando nota 5 e 18,2% atribuindo nota 4.

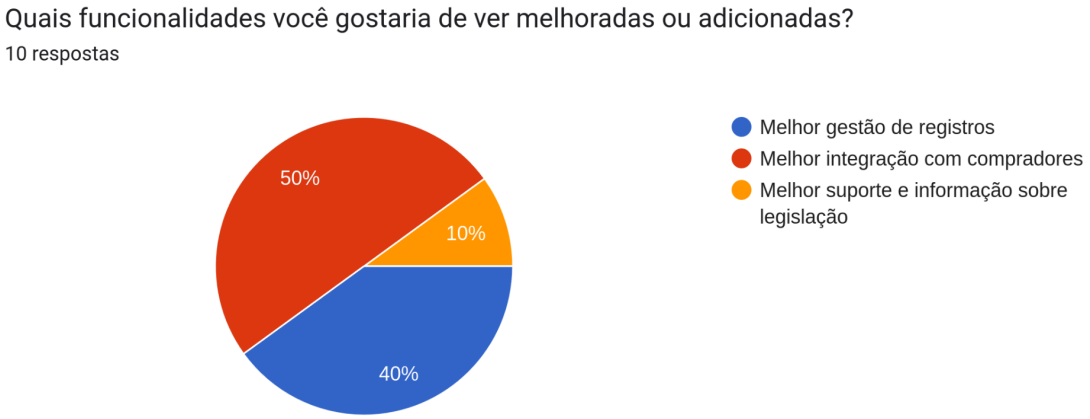
Figura 18 - Como avalia a interface do app ?



Fonte: Autor próprio.

Na figura 19, os participantes apontaram que a principal melhoria desejada é a gestão de registros (50%), seguida da integração com compradores (40%). Apenas 10% indicaram a necessidade de um suporte melhorado e mais informações sobre legislação.

Figura 19 - Quais funcionalidades de ver melhoradas ou adicionadas ?



Fonte: Autor próprio.

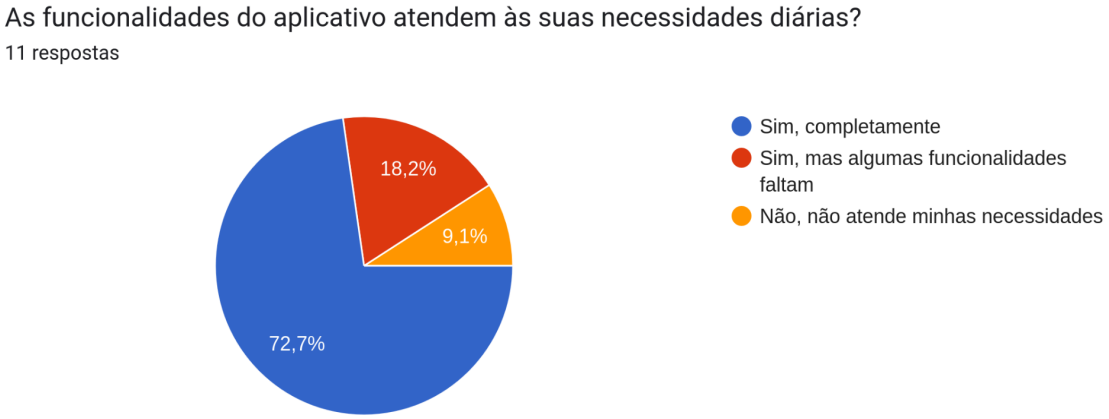
Conforme a figura 20, os entrevistados consideram o design do aplicativo intuitivo e de fácil navegação. Reforçando que os usuários conseguem utilizar suas funções sem dificuldades.

Figura 20 - Você considera o design do aplicativo intuitivo e fácil navegação?

Fonte: Autor próprio.

Por fim, a última questão figura 21 ,A maioria (72,7%) afirmou que o aplicativo atende completamente às suas necessidades diárias. No entanto, 18,2% sentem falta de algumas funcionalidades, e 9,1% relatam que o app ainda não atende suas demandas por completo.

Figura 21 - As funcionalidades atendem às suas necessidades diárias?



Fonte: Autor próprio.

Com base nas respostas obtidas no questionário de Pós-avaliação, pode-se concluir que o sistema teve um desempenho positivo, especialmente no que diz respeito à percepção dos participantes sobre a facilidade de uso e a aceitação do CoopDelta.

6 Considerações finais

Considerando o contexto da pesca artesanal na região de Araioses-MA, observa-se que a sustentabilidade local é um grande desafio enfrentado pelos pescadores. A exploração inadequada dos recursos pesqueiros pode resultar em danos ambientais e comprometer a subsistência da comunidade.

Ao utilizar a tecnologia para rastrear e monitorar as atividades de pesca, é possível garantir práticas mais sustentáveis, além de possibilitar o controle mais preciso das capturas. Dessa forma, o **CoopdeAlta** não só contribui para a melhoria da eficiência da atividade pesqueira, mas também apoia a preservação ambiental e o fortalecimento da economia local.

O presente trabalho visa desenvolvimento do aplicativo, para atender às necessidades da comunidade pesqueira de Araioses-MA. O aplicativo oferece uma interface simples para realizar o registro das capturas de pescado na região. A implementação de uma API entre o *front-end* e o *back-end* assegurará uma comunicação eficiente e segura dos dados, garantindo o bom funcionamento do sistema de forma ágil e precisa. Além disso, ajustes serão realizados com base na avaliação dos usuários, com o objetivo de aprimorar a usabilidade e a funcionalidade do aplicativo.

Como forma de ampliar e melhorar o aplicativo para versões futuras, pode ser implementado dados meteorológicos para auxiliar na previsão das melhores épocas para a pesca, ou a implementação de um sistema de alertas para promover boas práticas de captura. Além disso, a integração com outras plataformas e tecnologias emergentes pode ampliar a capacidade do *software*.

Referências

Lopes, J. M., Pinto, F. E. do N., Gomes, A. M. N., Sousa, A. W. da S., Silva-Matos, R. R. S., & Pereira, A. M. (2023). **Pesca artesanal em Araioses - MA, região do Delta das Américas. In *Ciências Agrárias: Debates Emblemáticos e Situação Perene* (Capítulo 10)**. Universidade Federal do Maranhão (UFMA), Centro de Ciências de Chapadinha.

PEDROSA, B.M.J.; LESSA, R.P.T. **O social como prioridade na pesca artesanal: diretrizes internacionais para a pesca artesanal sustentável.** Arquivos de Ciências do Mar, Fortaleza, v. 50, n.1, p. 7-13, jul./dez. 2017.

GARCIA, Maria Rodrigues; FURTADO, Marivânia Leonor. **A comunidade de pescadores tradicionais de Carnaubeiras-Araioses-MA: Percepções socioambientais e aspectos culturais.** Espaço e Cultura, n. 40, p. 181-202, 2016.

Laudon, K. C., & Laudon, J. P. (2018). ***Management Information Systems: Managing the Digital Firm***. 15ª edição. Pearson.

Welling, L., & Thomson, L. (2017). ***PHP and MySQL Web Development***. 5ª edição. Sams Publishing.

Angular Documentation. (2024). ***Angular - A framework for building client applications in HTML and TypeScript***. Disponível em: <https://angular.io/docs>

acessado em 6 de set. de 2024

Hevery, M. (2018). ***Angular: Up and Running: Learning Angular, Step by Step***. O'Reilly Media.

Google Developers. (2024). ***Progressive Web Apps***. Disponível em: https://developers.google.com/web/progressive-web-apps

Nielsen, J. (2021). ***Progressive Web Apps: A New Approach to Mobile and Desktop Applications*.** Nielsen Norman Group. Disponível em: <https://www.nngroup.com/articles/progressive-web-apps/> acesso em 6 set. de 2024

WELLING, Luke; THOMSON, Laura. ***PHP and MySQL Web Development***. 5. ed. São Paulo: Addison-Wesley, 2017.

KROENKE, David M.; AUER, David J. ***Database Processing: Fundamentals, Design, and Implementation***. 14. ed. São Paulo: Pearson, 2019.

SOMMERVILLE, I. **Engenharia de Software-9a** Edição. Pearson Education, 2011.

Martin, R. C. (2008). ***Clean Code: A Handbook of Agile Software Craftsmanship***. Prentice Hall.

ERSE, Alan Vasconcellos. **Desenvolvimento e análise do backend do projeto Cuidaldoso**. 2021.

SANTOS, C.C.; MELO, F.A.; ROCHA, F.M. **Etnoictiologia praticada pelos pescadores do Delta do Parnaíba, litoral piauiense.** In: GUZZI, A. (org.). *Biodiversidade do Delta do Parnaíba: litoral piauiense*. Parnaíba: EDUFPI, 2012.

FAO. **Diretrizes Voluntárias para Garantir a Pesca de Pequena Escala Sustentável no Contexto da Segurança Alimentar e da Erradicação da Pobreza.** 2017. 34 p.

Bruna B. Barro HOSTINGER. **As 10 linguagens de programação mais usadas em 2024: aprimore suas habilidades de desenvolvimento. *Hostinger Tutorials***, 2024. Disponível em: https://www.hostinger.com.br/tutoriais/linguagens-de-programacao-mais-usadas#:~:text=Al%C3%A9m%20disso%2C%20o%20PHP%20%C3%A9,PHP%20%C3%A9%20f%C3%A1cil%20de%20aprender. Acesso em: 3 dez. 2024.

ARMEL, Jamal. **Web application development with Laravel PHP Framework version 4**. 2014.

GOOGLE DEVELOPERS. ***Geolocation API***. Disponível em: https://developers.google.com/maps/documentation/geolocation/overview. Acesso em: 3 dez. 2024.

ERL, Thomas; PUTTINI, Ricardo; MAHMOOD, Zaigham. **Cloud computing: concepts, technology, security, and architecture**. 2. ed. São Paulo: Pearson, 2021.

Amazon. (2024). ***Amazon Web Services Overview*. Amazon Web Services**. Disponível em: https://aws.amazon.com/what-is-aws/

FOWLER, M; et al. **Padrões de Arquitetura de Aplicações Corporativas**. 1ªed. Porto Alegre: Editora Bookman, 2006

RICHARDS, Mark; FORD, Neal. ***Fundamentos da Arquitetura de Software: uma Abordagem de Engenharia***. 1. ed. Porto Alegre: Bookman, 2024.

BARBOSA, Adriley Samuel Ribeiro et al. **ANÁLISE COMPARATIVA ENTRE OS PADRÕES MVC, MVP, MVVM E MVI NA PLATAFORMA ANDROID**. 2022.

VOLPATO, E. **Quantos participantes chamar para o teste de usabilidade?** 2017 . Disponível em: < https://medium.com/testr/quantos-participantes-chamar-para-um-teste-de-usabilidade-7afc8bd7496#:~:text=Entre%20os%20profissionais%20de%20UX,dos%20problemas%20de%20uma%20interface.>.Acessado em 10 fev. 2025 .

APÊNDICES

Apêndice A Listagem de *end point*.

Método HTTP	Rota	Controlador	Descrição	Autenticado
POST	/usuario/registrar	UserController	Registrar um novo usuário.	Não
POST	/usuario/login	UserController	Realizar login de um usuário.	Não
GET	/busca-usuarios	UserController	Buscar todos os usuários.	Sim
POST	/usuario/editar/{id}	UserController	Editar um usuário específico.	Sim
GET	/usuario/me	UserController	Buscar informações do usuário logado.	Sim
POST	/pesca	RegistroPescaController	Registrar um novo registro de pesca.	Sim
PUT	/pesca/{id}	RegistroPescaController	Atualizar um registro de pesca.	Sim
GET	/pesca/meus	RegistroPescaController	Buscar registros de pesca do usuário logado.	Sim
GET	/pesca	RegistroPescaController	Buscar todos os registros de pesca.	Sim
POST	/pesca/relatorio-pesca	RegistroPescaController	Gerar relatório de pesca.	Sim
POST	/venda	RegistroVendaController	Registrar uma nova venda.	Sim
PUT	/venda/{id}	RegistroVendaController	Atualizar um registro de venda.	Sim
GET	/venda	RegistroVendaController	Buscar todos os registros de venda.	Sim
GET	/venda/meus	RegistroVendaController	Buscar registros de venda do usuário logado.	Sim
POST	/venda/relatorio-pesca	RegistroVendaController	Gerar relatório de vendas.	Sim

GET	/financeiros	RegistroFinanceiroController	Listar todos os registros financeiros.	Sim
POST	/financeiros	RegistroFinanceiroController	Registrar um novo registro financeiro.	Sim
GET	/localizacoes	LocalizacaoController	Listar todas as localizações.	Sim
POST	/localizacoes	LocalizacaoController	Registrar uma nova localização.	Sim
GET	/localizacoes/{id}	LocalizacaoController	Buscar uma localização específica.	Sim
GET	/busca-auditoria	AuditoriaController	Buscar todos os registros de auditoria.	Sim