UNIVERSIDADE TECNOLÓGICA FEDERAL DO PARANÁ
CURSO SUPERIOR DE TECNOLOGIA EM ANÁLISE E DESENVOLVIMENTO DE SISTEMAS

LEANDRO MAKOTO KOJIMA

DESENVOLVIMENTO DE UM SISTEMA WEB PARA MAPEAMENTO DE PONTOS DE PESCA MARÍTIMA E FLUVIAL

TRABALHO DE CONCLUSÃO DE CURSO

PATO BRANCO
2021

LEANDRO MAKOTO KOJIMA

DESENVOLVIMENTO DE UM SISTEMA WEB PARA MAPEAMENTO DE PONTOS DE PESCA MARÍTIMA E FLUVIAL

DEVELOPMENT OF A WEB SYSTEM FOR MAPPING MARINE AND RIVER FISHING POINTS

Trabalho de Conclusão de Curso de graduação, apresentado à disciplina de Trabalho de Conclusão de Curso 2, do Curso Superior de Tecnologia em Análise e Desenvolvimento de Sistemas, da Universidade Tecnológica Federal do Paraná, Câmpus Pato Branco, como requisito parcial para obtenção do título de Tecnólogo.

Orientador: Prof. Andreia Scariot Beulke

PATO BRANCO
2021

LEANDRO MAKOTO KOJIMA

DESENVOLVIMENTO DE UM SISTEMA WEB PARA MAPEAMENTO DE PONTOS DE PESCA MARÍTIMA E FLUVIAL

Trabalho de Conclusão de Curso de graduação, apresentado à disciplina de Trabalho de Conclusão de Curso 2, do Curso Superior de Tecnologia em Análise e Desenvolvimento de Sistemas, da Universidade Tecnológica Federal do Paraná, Câmpus Pato Branco, como requisito parcial para obtenção do título de Tecnólogo.

Data de aprovação: 16/fevereiro/2022

Robison Cris Brito
Doutor
Universidade Tecnológica Federal do Paraná

Lucilia Yoshie Araki
Mestre das Ciências
Universidade Tecnológica Federal do Paraná

Andreia Scariot Beulke
Mestre das Ciências
Universidade Tecnológica Federal do Paraná

PATO BRANCO
2021

RESUMO

Este trabalho apresenta o desenvolvimento de um sistema com o intuito disponibilizar informações e acesso para os diversos perfis de usuários, que vão desde pescadores mais experientes à iniciantes, incluindo, ainda, as pessoas que procuram opções de turismo, sem um interesse inicial em pesca. O sistema foi desenvolvido para a web, oferecendo uma forma de indicar geograficamente o ponto de pesca para melhor visualização. Também possibilita adicionar diversos tipos de informações sobre o ponto como: preço, endereço, horário de funcionamento, telefone, espécies que já foram capturadas no local, iscas mais utilizadas e qual espécie foi capturada com determinada isca. Para o desenvolvimento do sistema foi utilizado o *framework* AngularJS, o Bootstrap e a linguagem de programação JavaScript para o desenvolvimento das interfaces visuais e para o desenvolvimento *back-end* foi utilizada a linguagem de programação Java. O resultado da implementação do sistema foi satisfatório, atendendo às funcionalidades propostas.

Palavras-chave: Pesca esportiva. Marítima. Fluvial. Sistema web.

ABSTRACT

This work presents the development of a system with the purpose of computerize the monitoring program activities management of UTFPR (Federal Technological University of Paraná). With the application of information technology to the monitoring program management in university becomes easier and more efficient because various statistical data are made available through the reports available in the system. The system was developed to web, allowing registers, queries and reports generation necessary for the monitoring activities management. The system will initially be implanted at the UTFPR Pato Branco campus, but is prepared to be implanted in other campus. For the development of this system, the AngularJS framework was used together NodeJS. Also using Bootstrap and javascript tools for visual interfaces and for the development of the back-end was used the Java programming language. The results of the system implementation was satisfactory, given the proposed features.

Keywords: Sport Fishing. Salt-water. River. Web system.

‘

LISTA DE FIGURAS

Figura 1 – Tela inicial do Eclipse 19
Figura 2 – Tela inicial do Visual Paradigm Community Edition 20
Figura 3 – MySQL Workbench. 21
Figura 4 - Java SE (Standart Edition). 22
Figura 5 – Caso de uso do sistema. 28
Figura 6 – Diagrama entidade-relacionamento. 31
Figura 7 – Tela de cadastro de usuário 33
Figura 8 – Tela de autenticação do sistema 33
Figura 9 - Tela de mapeamento de pontos 34
Figura 10 – Tela de geração de gráficos 34
Figura 11 – Tela cadastro de ponto 35
Figura 12 – Tela cadastro de ponto 36
Figura 13 – Tela de geração de gráficos exibida em um dispositivo móvel 37
Figura 14 – Tela de galeria de fotos 38
Figura 15 – Tela de edição de senha 38
Figura 16 – Tela para edição de dados cadastrais 39
Figura 17 - Tela de cadastro de menu 40
Figura 18 - Tela de cadastro de parâmetros 40
Figura 19 – Organização das pastas. 42
Figura 20 – Estrutura de pastas da camada view. 47
Figura 21 – Estrutura de pastas das páginas 48
Figura 22 – Estrutura de pastas da camada controller 51

LISTA DE QUADROS

Quadro 1 - Lista de ferramentas e tecnologias.....18
Quadro 2 - Requisitos funcionais.....26
Quadro 3 - Requisitos não funcionais.....27
Quadro 4 - Expansão do caso de uso denominado manter ponto de pesca. 29
Quadro 5 - Expansão do caso de uso denominado manter foto à galeria.....29
Quadro 6 - Expansão do caso de uso manter usuário.....30

LISTAGEM DE CÓDIGOS

Listagem 1 - Método para salvar ponto de pesca. 42
Listagem 2 – Método para gerar gráfico de espécies. 43
Listagem 3 - Método para validar um novo usuário 44
Listagem 4 - Método para atualizar as configurações padrões do usuário 45
Listagem 5 - Método para salvar endereço do ponto de pesca. 46
Listagem 6 - Método para carregar total de pontos 48
Listagem 7 - Método de requisição para a camada de controle 49
Listagem 8 - Estrutura do gráfico de quantidade por espécie 50
Listagem 9 - Uso do fusioncharts no template 51
Listagem 10 - Método save para cadastro de usuário 52
Listagem 11 - Método para carregar os pontos do mapa 52

LISTA DE SIGLAS

API	*Application Programming Interface*
CSS	*Cascading Style Sheets*
DTO	*Data Transfer Object*
HTML	*Hyper Text Markup Language*
IDE	*Integrated Development Environment*
JSON	*JavaScript Object Notation*
JVM	*Java Virtual Machine*
POM	*Project Object Model*
RF	Requisito Funcional
RNF	Requisito Não Funcional
SE	*Standard Edition*
SQL	*Structured Query Language*
UML	*Unified Modeling Language*

SUMÁRIO

1 INTRODUÇÃO **11**
1.1 CONSIDERAÇÕES INICIAIS **11**
1.2 OBJETIVOS **12**
1.2.1 Objetivo Geral **12**
1.2.2 Objetivos Específicos **12**
1.3 JUSTIFICATIVA **13**
1.4 ESTRUTURA DO TRABALHO **13**
2 REFERENCIAL TEÓRICO **15**
2.1 PESCA ESPORTIVA **15**
3 MATERIAIS E MÉTODO **18**
3.1 MATERIAIS **18**
3.1.1 Eclipse **19**
3.1.3 MySQL WorkBench **20**
3.1.4 JavaScript **21**
3.1.5 Java SE **22**
3.1.6 AngularJS **22**
3.1.7 Tomcat **23**
3.1.8 Maven **23**
3.2 MÉTODO **23**
4 RESULTADOS **25**
4.1 ESCOPO DO SISTEMA **25**
4.2 MODELAGEM DO SISTEMA **26**
4.3 APRESENTAÇÃO DO SISTEMA **32**
Fonte: Autoria própria. **38**
4.4 IMPLEMENTAÇÃO DO SISTEMA **41**
4.2.1 *Model* **41**
4.2.2 *View* **47**
4.2.3 Controller **51**
5 CONCLUSÃO **53**
REFERÊNCIAS **54**

1 INTRODUÇÃO

Nesse capítulo são apresentadas as considerações iniciais, os objetivos, a justificativa da realização desse trabalho e a estrutura do trabalho que descreve brevemente o conteúdo dos capítulos.

1.1 CONSIDERAÇÕES INICIAIS

No início do século XXI, a pesca esportiva estava apresentando crescimento acentuado. É estimado que cerca de 40 milhões de americanos utilizam pelo menos um dia do ano para pescar e os gastos dessa atividade, incluindo equipamentos, viagens e hospedagem, chegam a 45 bilhões de dólares (BRITANNICA, 2021).

No Brasil, a pesca esportiva vem crescendo de maneira relevante em todos os estados brasileiros, estimulando a pesca marítima, fluvial e em pesqueiros. Com isso, há o aumento da procura de locais e pontos de pesca. Influenciando a economia, não apenas turística de muitas cidades brasileiras, mas também de outras atividades, como: guia de pesca, comércio de iscas naturais e artificiais, além da estrutura de alimentação e hospedagem e de materiais relacionados diretamente à pesca que incluem equipamentos, itens de vestuário e embarcações (FISH TV, 2021).

A preservação da natureza é essencial para a pesca esportiva, pois o peixe e o meio ambiente em que vive precisam estar saudáveis. O peixe precisa de um ambiente ecologicamente equilibrado para que tenha as condições adequadas para alimentação (cadeia alimentar), reprodução, proteção de predadores naturais. Assim, a associação do ecoturismo aos locais de pesca esportiva pode ser uma boa oportunidade para a economia local.

O desenvolvimento sustentável tem como base cinco áreas de importância para a humanidade e para o meio, que são: pessoas, prosperidade, paz, parcerias e planeta. E a pesca esportiva possui relação com todos esses pilares (BRITANNICA, 2021).

Considerando o envolvimento desses pilares e o enorme crescimento da pesca esportiva em todo o mundo, a proposta desse trabalho é o desenvolvimento de um sistema web que seja uma espécie de ponto de encontro virtual de iniciantes

e pescadores experientes de pesca esportiva seja marítima ou fluvial. Um ambiente para que essas pessoas possam trocar experiências e, assim, estarem mais preparadas para uma pescaria e que possam encontrar informações sobre pescaria e pontos de pesca para cada perfil de pescador. Obter informações de espécies, iscas, pontos de pesca, pesqueiros, rios, lagos e tanques são importantes para a melhoria da prática da pesca esportiva.

1.2 OBJETIVOS

O objetivo geral refere-se ao resultado principal da realização deste trabalho e os objetivos específicos complementam o geral em termos de funcionalidades do sistema.

1.2.1 Objetivo Geral

Desenvolver um sistema web para que os praticantes de pesca esportiva possam compartilhar informações relevantes para a prática desse esporte.

1.2.2 Objetivos Específicos

- Disponibilizar informações sobre locais de pesca.
- Obter dicas e informações dos locais de pesca que possam ser relevantes para realizar uma boa pescaria como: iscas, profundidade, horário, espécies, informações de entrada, estadia e preço.
- Disponibilizar informações que possam auxiliar os iniciantes na prática da pescaria esportiva.
- Apresentar informações gráficas de acordo com o local de pesca, espécies existentes, iscas que são mais recomendadas e épocas mais recomendadas para pesca.
- Disponibilizar galeria de fotos, podendo ser filtrada por local sejam imagens para pescadores ou para possíveis interessados em realizar atividades relacionadas ao ecoturismo nos locais de pesca ou nas suas proximidades.

1.3 JUSTIFICATIVA

A possibilidade de expandir, não apenas a pesca individualmente, mas prosperar em diversos setores de economia e cultura, é uma grande evolução socioeconômica e cultural, pois permite desenvolvimento e empregabilidade em muitos setores de atividade humana.

Tendo em vista esta possibilidade, a existência de um sistema web no qual as pessoas possam compartilhar informações como técnicas e procedimentos, locais de pesca, espécies e iscas, por exemplo, pode facilitar a acessibilidade dessas informações para os praticantes da pesca esportiva e para a divulgação desse esporte de maneira que mais pessoas venham a praticá-lo. Além disso, há a possibilidade de divulgação dos locais de pesca e proximidades que podem ser excelentes opções de passeios e viagens de ecoturismo. Atraindo, assim, não somente pessoas para a pesca esportiva, mas os que procuram turismo de aventura, ecoturismo e a realização de atividades que possibilitem maior contato com a natureza e a convivência com culturas e práticas locais como os ribeirinhos, caiçaras e comunidades de pescadores e tradicionais.

O sistema proposto nesse trabalho busca disponibilizar conteúdo de fácil entendimento e acesso para os diversos perfis de usuários, que vão desde pescadores mais experientes a iniciantes, incluindo, ainda, as pessoas que procuram opções de turismo, sem um interesse inicial em pesca. Possibilitando acrescentar e obter informações do sistema como por exemplo: a isca mais usada em determinada região, a espécie que mais é capturada, com qual isca a mesma foi capturada, a localização geográfica do ponto de pesca, entre outras. Um aspecto importante do sistema é a inclusão de imagens sobre os locais, peixes, equipamentos e outros.

1.4 ESTRUTURA DO TRABALHO

O primeiro Capítulo do trabalho apresentou as considerações iniciais, os objetivos e a justificativa do trabalho. O Capítulo 2 apresenta o referencial teórico que descreve sobre a pesca esportiva. No Capítulo 3 estão as ferramentas e tecnologias utilizadas na modelagem e no desenvolvimento do sistema que consiste

na implementação do *front-end* e do *back-end*. No Capítulo 4 é apresentado o resultado do trabalho e, por fim, é apresentada a conclusão e as referências.

2 REFERENCIAL TEÓRICO

Este capítulo apresenta os tópicos principais para apoiar o desenvolvimento deste trabalho de conclusão de curso.

2.1 PESCA ESPORTIVA

A pesca esportiva é entendida como a exploração de animais aquáticos sem uma finalidade econômica e não consumindo como uma fonte primária para alimentos ou artesanais. (FAO, 2012). Sua importância social, econômica e principalmente ecológica está sendo cada vez mais reconhecida em diversos países do mundo (FAO, 2012.). Já no Brasil a pesca esportiva é reconhecida como a modalidade de pesca por lazer.

É amplamente praticada apresentando 76% da Zona Econômica Exclusiva Mundial (MORA et al.,2009) em ambientes marinhos como também está presente em ambientes de água doce. Em quase todo o Brasil, a atividade é mais praticada em águas doces mesmo sendo um país com uma grande extensão de terras litorâneas, abrangendo mais de sete mil quilômetros de extensão (MUNDO EDUCAÇÃO, 2021).

A pesca esportiva é uma atividade complexa, existindo muitas modalidades e técnicas diferentes, apresentando desde tipos de equipamentos (molinetes, carretilhas, tipos de varas, etc.), locais (costeira, praia, oceano, rios, mangues, lagos, represas, pesque-pague entre outros), formas de acesso (terra firme ou embarcado), tipos de iscas (iscas naturais, iscas industrializadas e artificiais) e outros fatores. Tudo isso faz total diferença dependendo de onde, que modo e até quando está sendo realizado a pescaria (estação do ano, profundidade, pressão atmosférica, horário, entre outros).

De acordo com Cisneros-Montemayor & Sumaila (2010), a pesca esportiva possui grande importância econômica que diz, de acordo com dados coletados no mundo, que dezenas de milhões de pescadores esportivos movimentam mais de 40 bilhões de dólares por ano com gastos em todo tipo de apetrechos, como: equipamentos de pesca, vestimenta, iscas, combustível, passagens aéreas, estadia, entre outros, além de gerar milhares de empregos. Tal importância econômica ainda

não é bem reconhecida em países emergentes, inclusive no Brasil (FREIRE et al., 2016).

Diversas mudanças socioambientais têm relação com o crescimento econômico como alterações de hábitos de consumo nos países, que na pesca, vêm reduzindo a pesca comercial e uma das consequências disso é o crescimento da pratica da pesca esportiva. As pessoas acabam podendo investir mais tempo no lazer ao invés de colher matérias primas para o próprio consumo (FAO, 2012.). No Brasil, muitos pescadores que realizavam a pesca em grande escala onde o objetivo era comercial, atualmente se tornaram guias de pesca esportiva (HANAZAKI et al., 2007; RAMIRES & BARRELLA, 2003).

A Pesca esportiva pode gerar grandes benefícios individuais e familiares, sendo entre eles: a prática de exercícios físicos, relaxamento mental, alimentar, melhoria do relacionamento familiar, compartilhar conhecimentos e valores entre gerações sendo categorizada como uma atividade comum com a família e amigos (LANG, 2014; AUSTRALIA, 2010).

Pode também trazer benefícios ecológicos, por meio do turismo e da própria pesca esportiva pois os guias fornecem informações sobre aspectos biológicos e ecológicos gerando dados de mudanças comportamentais, por exemplo, para pesquisadores e a própria comunidade. Além de que os guias e os pescadores são contra a deterioração do ambiente explorado, incentivando-os a atuarem positivamente na conservação do meio ambiente, mantendo-o ou até melhorando (PARKKILA et al., 2010). Mas, apesar de todos os benefícios a pesca esportiva não é isenta de impactos ambientais. A alta seletividade de espécies e tamanhos, faz com que algumas espécies mais procuradas tenham um elevado declínio em sua população (LEWIN; MCPHEE; ARLINGHAUS, 2008).

A pesca esportiva no Brasil apresenta um grande potencial, pois existe uma grande diversidade de peixes. Apesar disso, os dados sobre a pesca esportiva no Brasil são escassos, principalmente em ambientes marinhos, que interfere na mensuração dos benefícios e impactos desta atividade. Estima-se que menos de 15% dos pescadores brasileiros sejam registrados oficialmente (FREIRE; MACHADO; CREPADLDI, 2012).

Em Mongaguá e Itanhaém, ambas cidades no litoral de São Paulo, a pesca esportiva e amadora são umas das principais atividades de lazer praticadas na região existindo a possibilidade de pesca de plataforma, pesca embarcada com a

existência de clubes náuticos na região, pesca costeira e pesca na praia. Apesar de existir tanta diversidade entre os locais de pesca, não há dados sobre o perfil de pescadores que frequentam, não apenas regiões litorâneas, mas também a grande abrangência territorial aquática do país, bem como informações sobre a pesca e as espécies existentes em cada local.

3 MATERIAIS E MÉTODO

Este capítulo apresenta os materiais e os métodos utilizados para efetuar a análise e o desenvolvimento do sistema proposto.

3.1 MATERIAIS

Esta seção apresenta as ferramentas e as tecnologias utilizadas no desenvolvimento deste trabalho. No Quadro 1 estão as principais ferramentas e tecnologias utilizadas na modelagem e no desenvolvimento do sistema.

Quadro 1 - Lista de ferramentas e tecnologias

Ferramenta / Tecnologia	**Versão**	**Finalidade**
AngularJS	1.3.13	Desenvolvimento *front end*
Bootstrap	3.3.2	Biblioteca de componentes
CSS	3	Estilização de páginas
Eclipse	4.15.0	IDE de desenvolvimento
FusionCharts	3.17.0	Modelagem de gráficos
HTML	5	Linguagem de Estruturação de Conteúdo
Java	8	Desenvolvimento *back end*
Javascript	1.6	Desenvolvimento *front end*
Maven	3.0.0	Gerenciamento de dependências
MySQL	8	Banco de dados
Sourcetree	3.3.9	Versionamento do projeto
Tomcat	8.5	Servidor
Visual Paradigm	16.2	Modelagem UML
MySQL Workbanch	8	IDE para codificação SQL

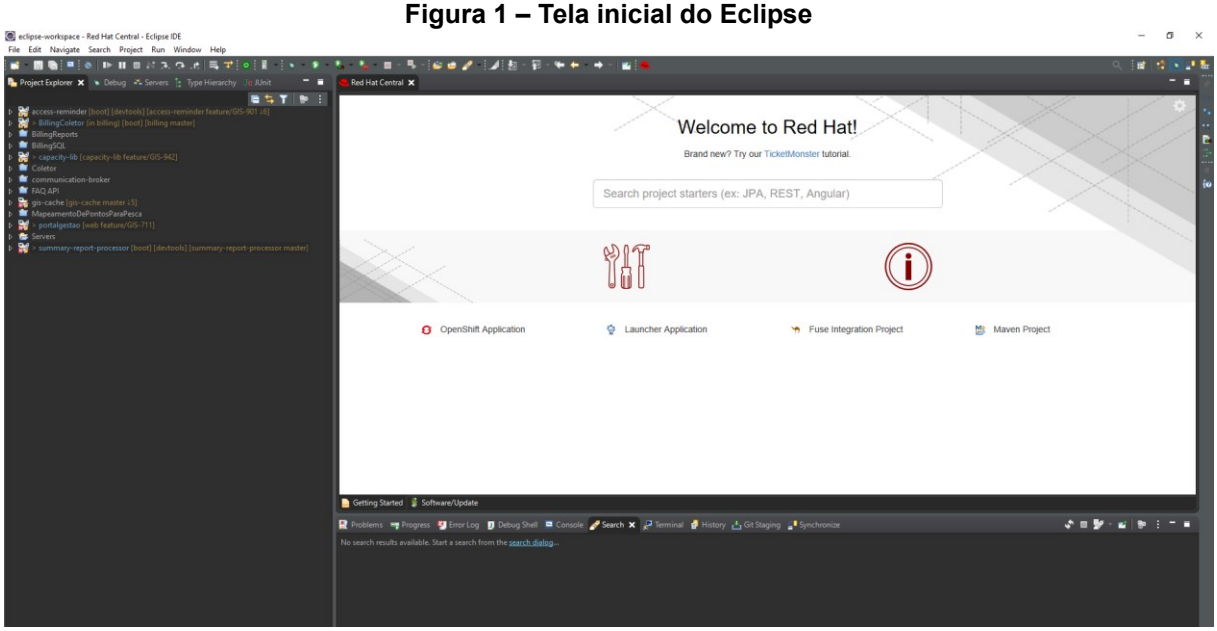
Fonte: Autoria Própria.

A seguir seguem algumas das principais ferramentas utilizadas para a implementação do sistema proposto e um breve descritivo sobre as principais ferramentas utilizadas.

3.1.1 Eclipse

O Eclipse é uma *Integrated Development Environment* (IDE) que possui código livre, focada no desenvolvimento Java, porém, com adição de alguns *plug-ins* e que suportar outras linguagens de programação como: C, C++, PHP, entre outras. Também é possível realizar a conexão com banco de dados. A Figura 1 ilustra a tela inicial da IDE Eclipse.

Figura 1 – Tela inicial do Eclipse

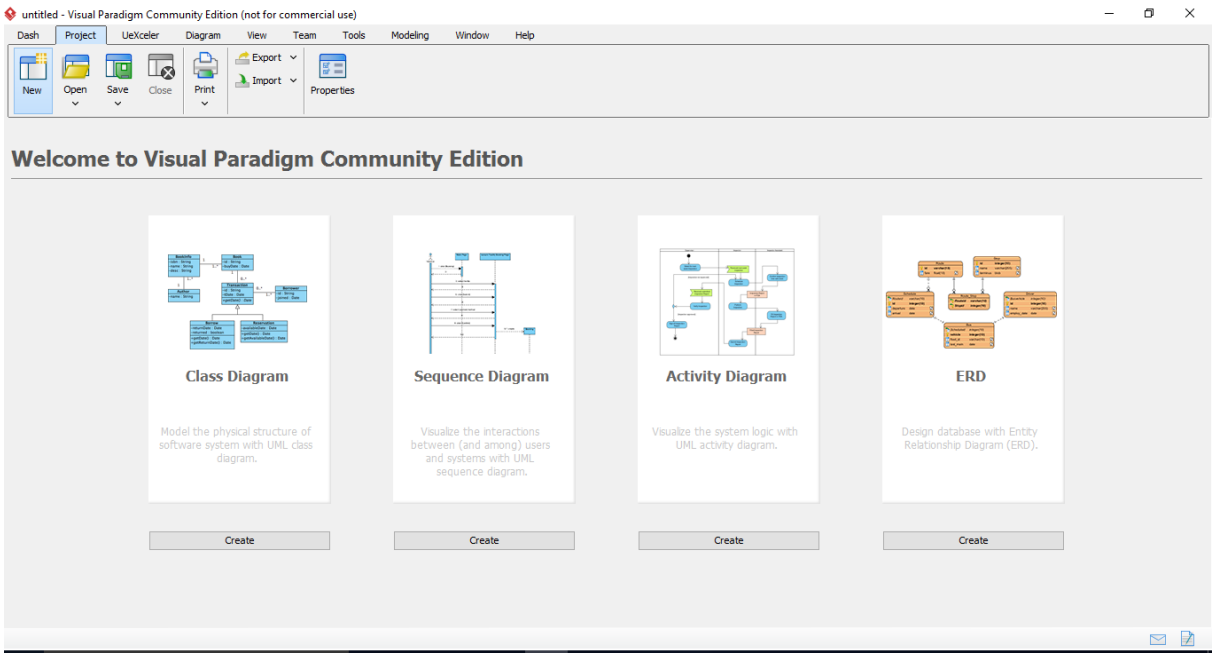


Fonte: Autoria Própria.

3.1.2 Visual Paradigm Community

A ferramenta Visual Paradigm permite a modelagem de diagramas *Unified Modeling Language* (UML) e foi utilizada para modelagem do diagrama de caso de uso. A mesma exige pagamento de licença para seu uso, porém, existe a possibilidade de se usar a versão *Community* que é disponibilizada para testes por 30 (trinta) dias após a instalação. Essa foi a versão utilizada no desenvolvimento da modelagem do trabalho. A tela inicial da ferramenta é apresentada na Figura 2.

Figura 2 – Tela inicial do Visual Paradigm Community Edition

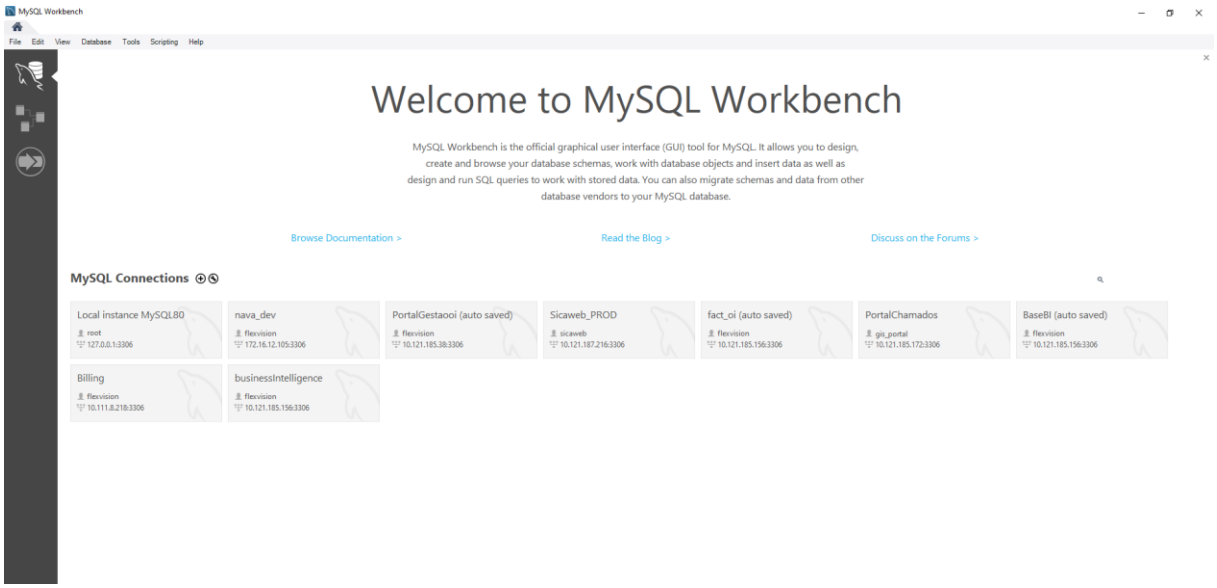


Fonte: Autoria Própria.

3.1.3 MySQL WorkBench

O MySQL WorkBench é uma ferramenta gratuita que serve para modelagem de banco de dados, que permite abrir e fechar conexões, executar códigos *Standard Query Language* (SQL), importar e exportar base de dados, além de outras funções. É uma ferramenta que possibilita gerenciar uma base de dados com suas principais necessidades atendidas. A Figura 3 representa um exemplo de modelagem utilizando a ferramenta MySQL WorkBench.

Figura 3 – MySQL Workbench.



Fonte: Autoria Própria.

3.1.4 JavaScript

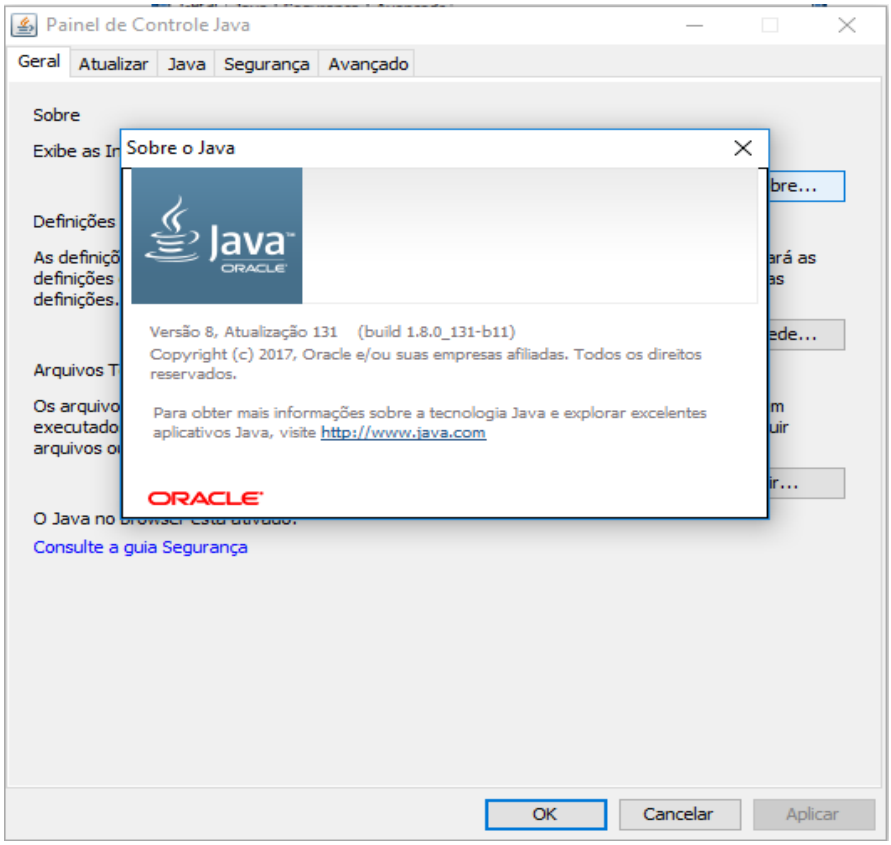
O JavaScript é uma linguagem de *script* orientada a objetos, usada para tornar páginas *web* interativas, como por exemplo: o uso de animações, eventos de cliques em botões, etc. possui uma biblioteca padrão de objetos como *arrays, date, string e integer*.

Quando utilizada para programação do lado do cliente permite respostas a eventos do usuário, como eventos de clique, entrada de dados em um formulário e navegação entre páginas. Do lado do servidor o JavaScript permite a interação e troca de dados com o banco de dados (MDN Web Docs, 2021).

3.1.5 Java SE

O Java *Standard Edition* (SE) é uma ferramenta para o desenvolvimento de programas na linguagem de programação Java. O Java SE contém um ambiente para o desenvolvimento e execução de aplicações Java, incluindo a *Java Virtual Machine* (JVM) e um compilador Java, ilustrado na Figura 4.

Figura 4 - Java SE (Standart Edition).



Fonte: Autoria Própria.

3.1.6 AngularJS

O AngularJS é um *framework* estrutural para aplicações web que permite entender a sintaxe da linguagem *Hyper Text Markup Language* (HTML) da sua aplicação para expressar os componentes de maneira clara. A vinculação de dados e a injeção de dependências do AngularJS elimina muito código que seria necessário escrever. E isso acontece dentro do navegador, facilitando a implementação das aplicações de servidores.

O AngularJS tenta minimizar a incompatibilidade de dependência entre o HTML e o que o aplicativo precisa, criando novas estruturas em tempo de execução dentro do próprio navegador. Faz com que o navegador aprenda uma nova sintaxe por meio de uma construção de diretivas (ANGULARJS, 2021).

3.1.7 Tomcat

O Apache Tomcat é um servidor de aplicativos Java de código-fonte aberto. Ele implementa as tecnologias Java Servlet, JavaServer Pages, Java Expression Language e Java WebSocket (ORACLE, 2021).

3.1.8 Maven

Apache Maven é uma ferramenta de gerenciamento e compreensão de projetos de software. Baseada no conceito de modelo de objeto de projeto, *Project Object Model* (POM), podendo gerenciar a construção, o relatório e a documentação de um projeto a partir de uma informação central.

Seu objetivo principal é permitir que um desenvolvedor compreenda o status do desenvolvimento no menor período de tempo, reunir o princípio da melhor prática de desenvolvimento e facilitar a orientação de um projeto.

3.2 MÉTODO

Nesta seção são apresentadas as principais atividades do desenvolvimento do sistema baseando-se na proposta de Pressman (2002) para o modelo sequencial linear que são análise, projeto, codificação e testes. A seguir as fases definidas para este trabalho foram:

a) Levantamento de requisitos: há diferentes maneiras de se recolher requisitos, sendo por meio de observação direta do problema, pesquisando em fontes bibliográficas, realizando entrevistas e aplicando questionários.

Neste trabalho, o levantamento de dados foi por meio de pesquisas em fontes bibliográficas e observações feitas diretamente ao problema, com intuito de identificar suas maiores necessidades.

b) Análise e Projeto: com os requisitos devidamente coletados, foram definidos os casos de uso. Nessa fase foram identificados os atores e algumas funcionalidades para a solução. Além disso foi criado o diagrama de entidade e relacionamentos do banco de dados. Para a modelagem do diagrama de caso de uso foi utilizada a ferramenta Visual Paradigm Community Edition, e para a modelagem do diagrama de entidade e relacionamentos foi utilizado o MySQL Workbanch. Também foram criados alguns casos de uso expandidos para detalhar as operações do sistema proposto.

d) Implementação e testes: foi utilizado a IDE Eclipse para a implementação na linguagem Java, utilizando o *framework* Spring para o mapeamento das classes, para administração da base de dados foi utilizado o MySQL Workbench para SQL. Os testes do sistema foram realizados informalmente com objetivo de encontrar possíveis erros e *bugs* no código implementado.

4 RESULTADOS

Este capítulo apresenta os resultados obtidos com a realização deste trabalho, que é o levantamento de requisitos, a modelagem, o projeto do sistema para mapeamento de pontos de pesca e o desenvolvimento do sistema que consiste na implementação do *front-end* e do *back-end*.

4.1 ESCOPO DO SISTEMA

O sistema pode ser acessado por pessoas que possam contribuir com informações essenciais para uma pescaria esportiva. As informações são compartilhadas entre todos os usuários cadastrados. Há dois níveis de usuário, sendo: pescador e administrador. O pescador pode adicionar pontos de pesca e fotos na galeria, também pode editar e excluir apenas as informações por ele incluídas. O administrador tem, além das permissões do tipo pescador, o acesso a todas as informações cadastrais de usuário, podendo adicionar, editar e excluir os dados.

O sistema visa facilitar o recolhimento de informações sobre pontos de pesca, que pode ser no rio, mar ou pesqueiros, visando facilitar o acesso à informação do local bem como a sua divulgação, e contribuir para a expansão da modalidade da pesca esportiva no Brasil.

É muito comum pessoas que se interessam pela pesca esportiva, especialmente os principiantes, não saberem da existência de determinados pontos de pesca que podem ser de deslocamento fácil a partir da sua residência. Ou, então, ir até um pesqueiro, por exemplo, despreparado para as condições do local e o tipo de pesca que é realizado nesse local o que pode exigir equipamentos e iscas, por exemplo, específicas. Essas pessoas, muitas vezes, acabam não voltando para o mesmo ponto com o pensamento de que o local é ruim ou que não tem uma quantidade grande de peixes, porém, apenas não utilizou a isca certa. Para as pessoas que precisam de informações essenciais para realizar uma pescaria, como: iscas mais utilizadas, espécies mais capturadas, horário mais ativo, por exemplo, obter esses dados por meio de um sistema web, pode auxiliar na realização de uma

pescaria com resultados mais satisfatórios, além de aumentar o número de pessoas interessadas por esse esporte e a movimentação econômica que pode proporcionar provenientes de hospedagem e alimentação, por exemplo.

O sistema web proposto, como resultado desse trabalho, oferece uma forma de indicar geograficamente o ponto de pesca para melhor visualização. Também permite adicionar diversos tipos de informações sobre o ponto como: preço, endereço, horário de funcionamento, telefone, espécies que já foram capturadas no local e iscas mais utilizadas, por exemplo.

O sistema também disponibiliza gráficos que podem ser filtrados por ponto de pesca. Pode ser consultado, por exemplo, quantos peixes foram capturados em determinado ponto de pesca. Também foi disponibilizado uma galeria de fotos, que pode ser filtrada por isca, local e espécie. As fotos podem ser relevantes, além de para a atividade de pescaria em si, para divulgação do local para ecoturistas e turismo de aventura, tornando-se uma fonte alternativa de renda para a localidade.

4.2 MODELAGEM DO SISTEMA

Nos Quadros 2 e 3 são apresentados, respectivamente, os requisitos funcionais e não funcionais do sistema proposto a fim de representar as funcionalidades do sistema. Por padrão, foi usado a sigla RF para representar os requisitos funcionais do sistema e a sigla RNF para representar os requisitos não funcionais.

Quadro 2 - Requisitos funcionais

Identificação	Nome	Descrição
RF01	Manter usuário	O cadastro de usuário será efetuado por um administrador do sistema.
RF02	Manter Ponto de Pesca	Ao acessar o formulário de cadastro de pontos de pesca, será adicionado um marcador com a localização do dispositivo que sendo utilizado para entrar no sistema, o usuário poderá alterar o posicionamento do marcador selecionando um ponto no mapa. Deve preencher os campos de cadastro e clicar no botão para salvar o cadastro.
RF03	Gerar gráficos	O usuário poderá gerar gráficos para obter informações por ponto de pesca, como: iscas

		mais utilizadas, espécies mais capturadas e uma linha do tempo para saber em que período do ano há mais capturas.
RF04	Manter galeria de fotos	Será permitido o cadastro de fotos à galeria. Acessando o formulário de cadastro de fotos o usuário deverá selecionar a foto que deseja cadastrar e preencher os campos.
RF05	Manter dicas do ponto de pesca	Haverá uma tela que fornecerá dicas sobre o ponto, iscas mais utilizadas, horários, espécies, etc.
RF06	Manter lista de acesso	A lista de acesso é uma forma para limitar o acesso aos módulos do sistema. Apenas o usuário do tipo administrador poderá ter acesso ao formulário de acessos.
RF07	Manter menu	Os menus são cadastrados pelo usuário com perfil de administrador, que podem ser menus de configuração e de produto.
RF08	Manter parâmetro	Os parâmetros do sistema podem ser cadastrados via tela, por um usuário com perfil de administrador.
RF09	Filtrar fotos	Na galeria de fotos será permitido a filtragem de fotos por tipo do ponto de pesca, nome do local, isca e espécie.
RF10	Filtrar informações e dicas	Na tela de informações e dicas de pontos de pesca, será filtrado por ponto de pesca. Deverá ser selecionado um ponto de pesca para carregar as informações da tela.
RF11	Abrir ponto de pesca no Google Maps	O sistema permitirá o redirecionamento para o Google *Maps*. Selecionando o marcador do mapa na página principal do sistema, o usuário poderá ser redirecionado à página do Google *Maps* com as coordenadas referentes ao ponto de pesca selecionado.
RF12	Alterar dados	O sistema possibilitará a alteração de dados cadastrais dos usuários, como: senha, email e contato.

Fonte: Autoria própria.

Quadro 3 - Requisitos não funcionais.

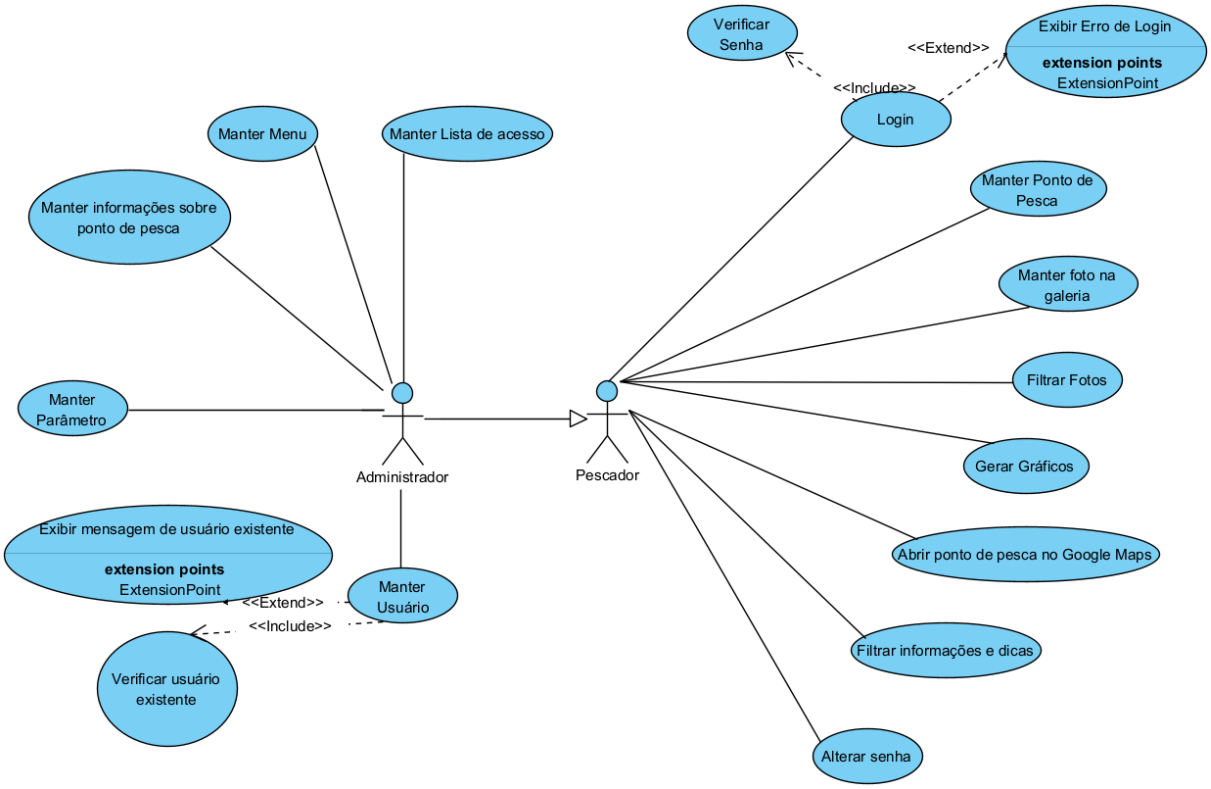
Identificação	**Nome**	**Descrição**
RNF01	Autenticação e acesso	O usuário deverá acessar o sistema por meio de *login* e senha.
RNF02	Uso de *design* responsivo nas interfaces gráficas	O sistema foi desenvolvido para executar em ambiente web e possui um leiaute responsivo, independente do *front-end* que será utilizado para acesso: navegador *web*, *smartphone* ou *tablet*.
RNF03	Controle de	Cada tipo de usuário terá acesso apenas aos

	acessos às páginas	módulos permitidos no sistema.
RNF04	Validação de cadastro de usuário	Para o cadastro de usuários, serão validados os dados preenchidos para garantir que não haverá duplicidade, campos em branco e dados preenchidos corretamente.

Fonte: Autoria própria.

O diagrama de casos de uso apresentado na Figura 5, representa a descrição de uma unidade funcional, no qual são envolvidos os atores de cada funcionalidade do sistema.

Figura 5 – Caso de uso do sistema.



Fonte: Autoria própria.

A seguir são apresentadas algumas expansões de casos de uso que apresentam uma descrição mais detalhada das funcionalidades do sistema.

No Quadro 4, é apresentado a expansão de caso de uso para o cadastro de ponto de pesca.

Quadro 4 - Expansão do caso de uso denominado manter ponto de pesca.

Caso de uso	**Manter ponto de pesca**
Atores	Pescador e Administrador
Descrição geral	O caso de uso inicia-se quando o Pescador ou Administrador deseja manter um ponto de pesca.
Pré-condições	Pescador ou Administrador autenticado no sistema.
Fluxo principal	1. O Usuário deseja manter um ponto de pesca. 2. O sistema solicita o cadastro de um novo ponto de pesca. 3. O usuário informa as informações do novo ponto de pesca. (Exc) 4. O sistema verifica duplicidade do ponto de pesca informado. 4.1. O ponto de pesca não é duplicado. 4.1.1. O sistema segue adiante para o passo 5. 4.2. O ponto de pesca é duplicado. 4.2.1. O sistema gera mensagem de aviso ao usuário. 4.2.2. Retorna para o passo 3. 5. O sistema cadastra o novo ponto de pesca e atualiza o mapa.
Tratamento de exceções	3a. Horário inválido 3a.1 Retorna ao passo 1. 3b. Campo em branco 3b.1 Retorna ao passo 1.

Fonte: Autoria própria.

A expansão de caso de uso para cadastrar uma nova foto ao sistema é mostrada no Quadro 5.

Quadro 5 - Expansão do caso de uso denominado manter foto à galeria.

Caso de uso	**Manter foto na galeria**
Atores	Administrador e Pescador
Descrição geral	O caso de uso inicia-se quando o Usuário deseja adicionar uma foto à galeria de fotos.
Pré-condições	Usuário autenticado no sistema.
Fluxo principal	1. O Administrador ou pescador deseja adicionar uma foto. (Exc) 2. O Administrador ou pescador carrega a foto e preenche as informações. 3. O sistema mantém a foto e as informações preenchidas.
Tratamento de exceções	3a. Campos obrigatórios em branco 3a.1 Retorna para o passo 1.

Fonte: Autoria própria.

Para realizar o cadastro de um novo usuário, é necessário que o usuário que irá realizar o cadastro tenha privilégios de administrador do sistema. O processo

possui validações de dados preenchidos pelo usuário para garantir a integridade e singularidade do cadastro. O Quadro 6 representa a expansão do caso de uso deste processo de cadastro.

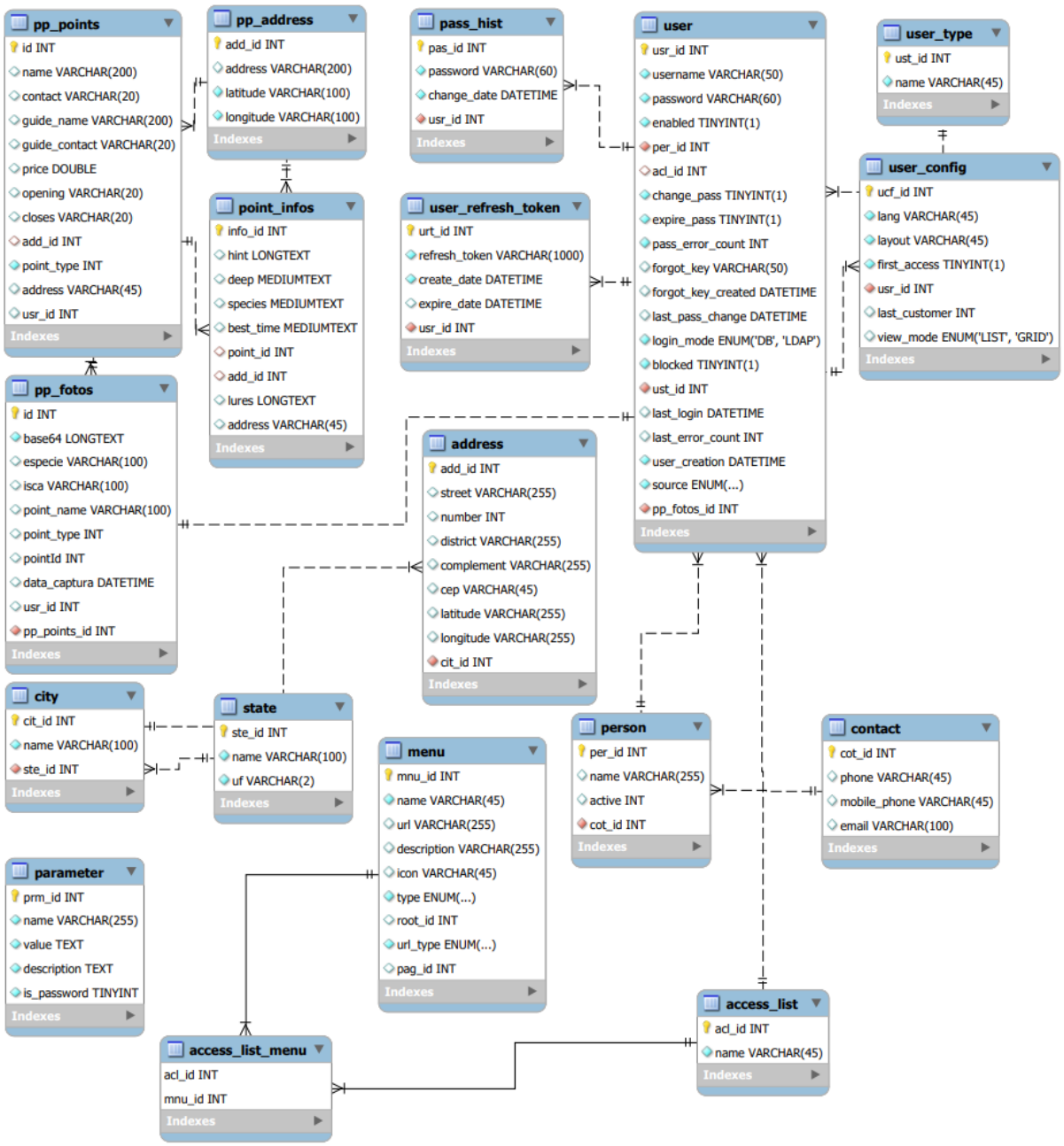
Quadro 6 - Expansão do caso de uso manter usuário.

Caso de uso	**Manter Usuário**
Atores	Administrador
Descrição geral	O caso de uso inicia-se quando o Usuário administrador deseja cadastrar um novo usuário ao sistema.
Pré-condições	Estar autenticado com um usuário do tipo administrador.
Fluxo principal	1. O Administrador deseja realizar o cadastro no sistema. 2. O Administrador preenche os campos cadastrais. 3. O sistema verifica duplicidade nas informações fornecidas. 3.1. As informações são duplicadas. 3.1.1 Volta para o passo 2 3.2. As informações não são duplicadas. 3.2.2. O sistema solicita cadastro para o novo usuário. 3.2.3. Mensagem de cadastro realizado.
Tratamento de exceções	2a. Campos obrigatórios em branco. 2ª.1 Retorna para o passo 2

Fonte: Autoria própria.

O diagrama de entidade-relacionamento é apresentado na Figura 6, que representa o banco de dados da aplicação.

Figura 6 – Diagrama entidade-relacionamento.



Fonte: Autoria própria.

No diagrama da Figura 6, a entidade denominada “pp_pontos”, é responsável pelo armazenamento de pontos de pesca cadastrados no sistema, contendo o nome do ponto de pesca, o contato, os horários de funcionamento e outros dados. Essa entidade tem relações com as entidades denominadas “pp_infos”, que é responsável por armazenar as informações e dicas de um ponto de pesca e “pp_address” que é responsável por armazenar o endereço do ponto de pesca cadastrado.

A entidade denominada *user* é armazena os dados do usuário, com as suas informações básicas de *login* e senha, o tipo de usuário, que é representado pela entidade *user_type*, e outras informações que podem ser consultadas na Figura 6.

A entidade denominada “menu” é a representação do cadastro de menus no sistema, no qual um usuário com perfil de administrador cadastra um novo menu ao sistema e configura o acesso ao menu pelos tipos de usuário. Esta relação entre menu e acesso é representada pelas entidades denominadas “*access_list_menu”* e “*access_list”.*

A entidade responsável pelo armazenamento das fotos cadastradas no sistema é a entidade denominada “pp_fotos”, que contém o ponto de pesca onde a foto foi tirada, a isca usada, o nome da espécie, além dos campos de chave estrangeira que são “*usr_id”* e “*pointId”*.

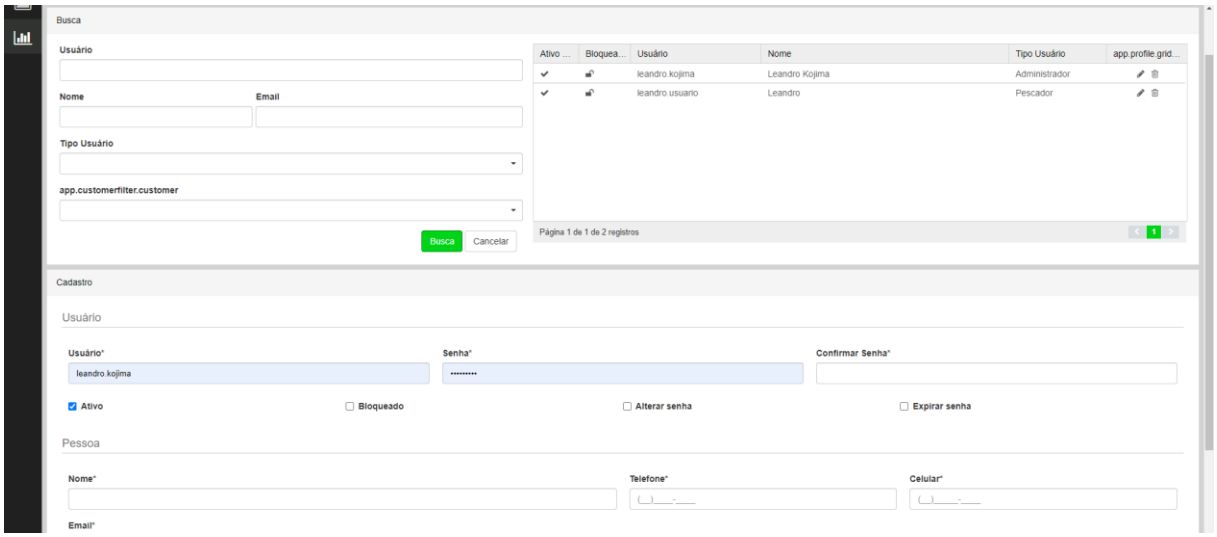
As informações e dicas do sistema são registradas na entidade denominada “point_infos” que contém informações e dicas de um ponto de pesca, como: profundidade, iscas mais utilizadas, dicas e espécies encontradas no ponto. Essa entidade tem relacionamento com a entidade que representa o ponto de pesca e seu endereço.

4.3 APRESENTAÇÃO DO SISTEMA

Após o levantamento de requisitos do sistema, foram desenvolvidas as telas utilizando as tecnologias Bootstrap, AngularJS e Fusioncharts.

Para criar um novo usuário é necessário que o usuário com perfil de administrador realize essa operação. A Figura 7 contém a tela de cadastro de um novo usuário no sistema.

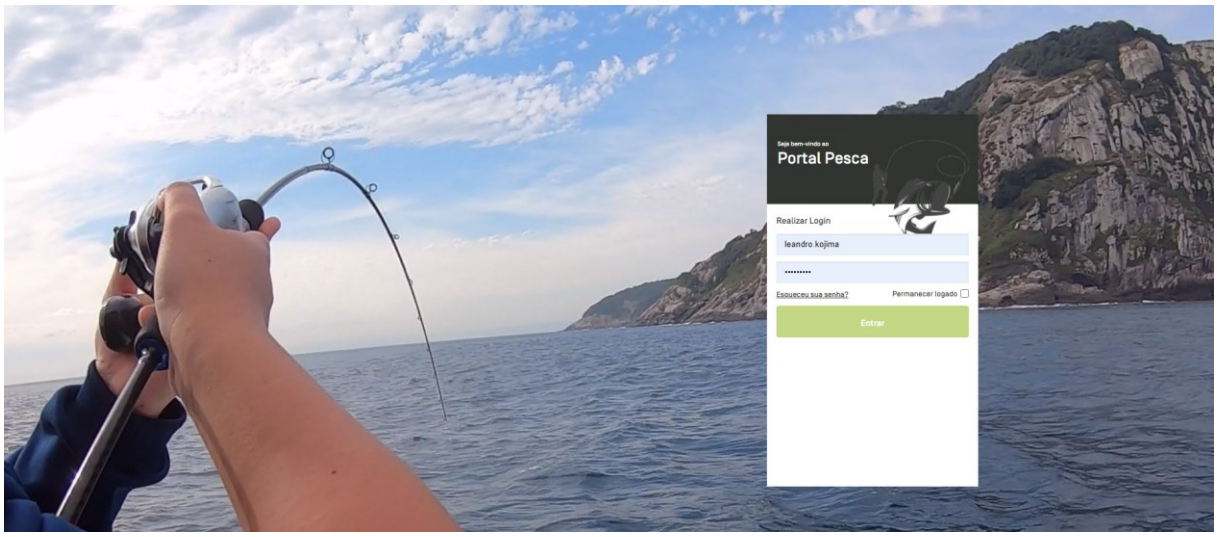
Figura 7 – Tela de cadastro de usuário



Fonte: Autoria própria.

Na Figura 8 está representado a tela de autenticação do usuário, onde deverá fornecer o nome do usuário e senha para acessar o sistema.

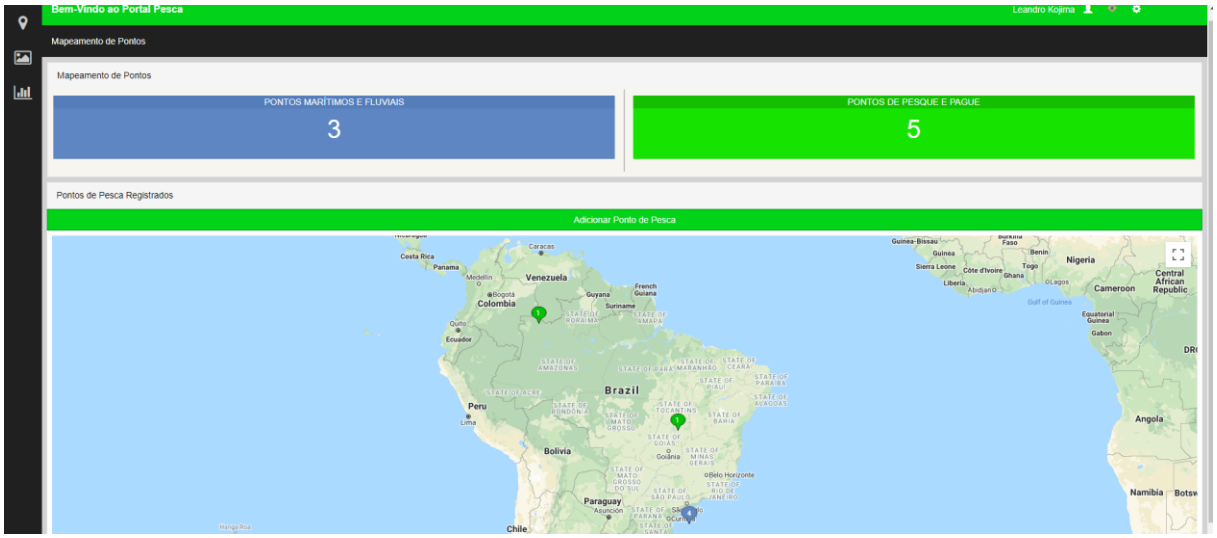
Figura 8 – Tela de autenticação do sistema



Fonte: Autoria própria.

A Figura 9 representa a tela inicial do sistema, que serve para mapear pontos, onde estão fixados com um marcador no mapa todo os pontos cadastrados no sistema. No mapa, o usuário pode obter mais informações sobre um ponto clicando no marcador e será exibido algumas informações básicas sobre o mesmo. Também poderá acessar o formulário de cadastro de pontos por meio dessa tela.

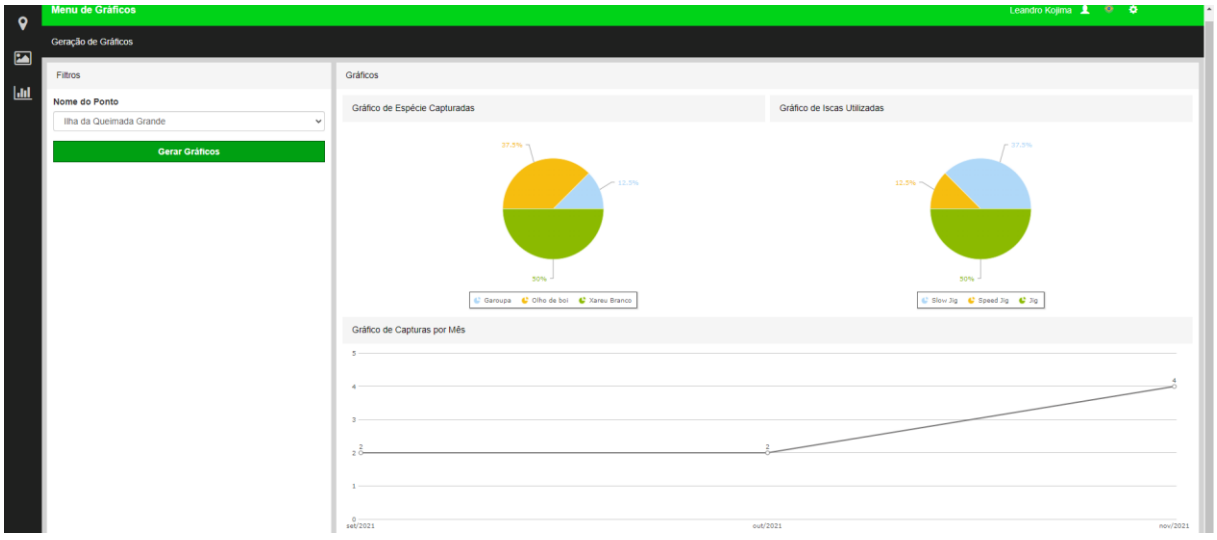
Figura 9 - Tela de mapeamento de pontos



Fonte: Autoria própria.

A Figura 10 representa a tela de gráficos, na qual o usuário poderá escolher um ponto de pesca e gerar gráficos mostrando as espécies mais capturadas, as iscas mais utilizadas no determinado ponto e uma linha do tempo mostrando em qual período do ano é capturado mais peixes no ponto escolhido.

Figura 10 – Tela de geração de gráficos

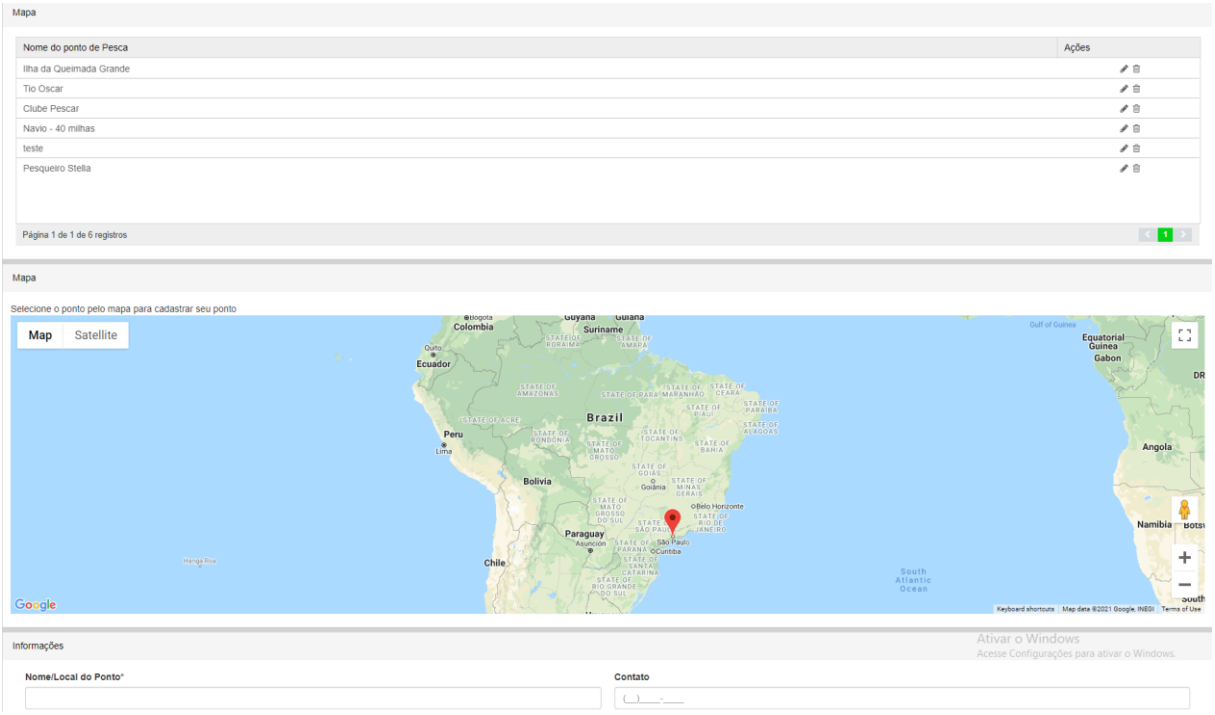


Fonte: Autoria própria.

O formulário de cadastro de um novo ponto de pesca é exibido na Figura 11, onde ao inicializar é adicionado um marcador no mapa da localização atual do dispositivo usado para acessar o sistema, podendo alterar a posição do marcador com um clique no mapa. O usuário deverá preencher os dados do formulário para

realizar o novo cadastro de ponto de pesca. Será exibida uma lista de pontos de pesca no início da página, possibilitando a edição dos pontos. Para o usuário com perfil de administrador, serão exibidos todos os pontos de pesca cadastrados e para o usuário com perfil de pescador, serão listados apenas os pontos de pesca cadastrados por ele mesmo.

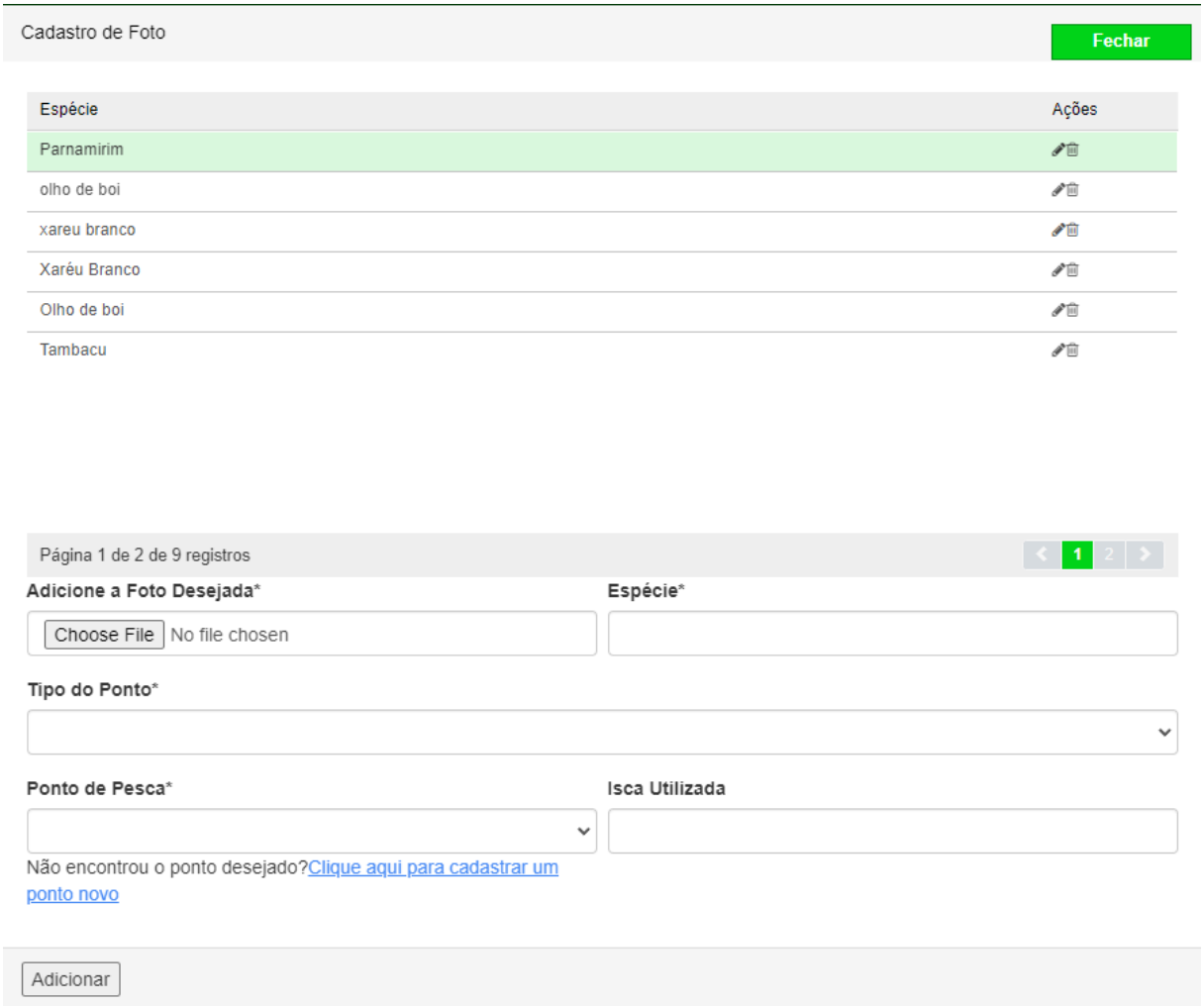
Figura 11 – Tela cadastro de ponto



Fonte: Autoria própria.

A Figura 12, representa a tela de cadastro de fotos para a galeria, na qual o usuário deve escolher o arquivo para fazer o *upload* da foto no banco de dados. Nesta sessão para o usuário com perfil de administrador serão exibidas todas as fotos cadastradas, sendo possível editá-las. Para o usuário com perfil de pescador serão exibidas apenas as fotos cadastradas por ele mesmo.

Figura 12 – Tela cadastro de ponto



Fonte: Autoria própria.

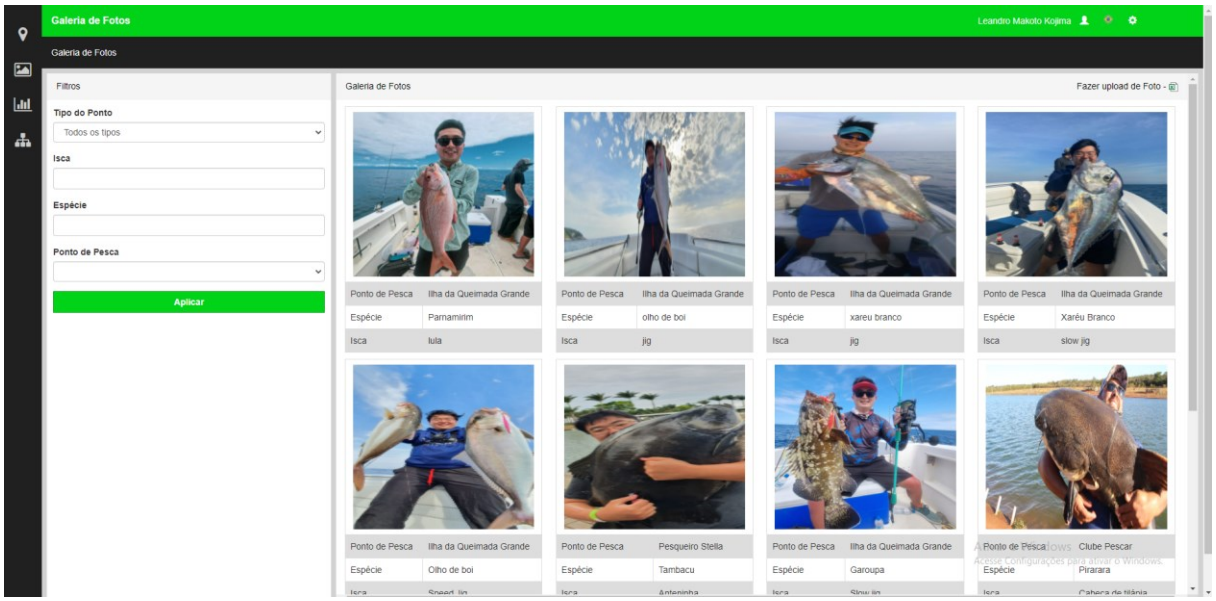
A Figura 13, exibe a tela na qual o sistema é visualizado por um dispositivo móvel. O leiaute foi desenvolvido para se adequar a vários tipos de dispositivos, como: computadores, celulares, *tablets* e notebooks. Para implementar a responsividade foi utilizado o *framework* Bootstrap que contém um sistema de *grid* que utiliza vários *containers*, linhas e colunas para adaptar o conteúdo conforme o dispositivo utilizado pelo usuário.

Figura 13 – Tela de geração de gráficos exibida em um dispositivo móvel

Fonte: Autoria própria.

A Figura 14, apresenta a tela da galeria de fotos do sistema, na qual é exibida todas as fotos cadastradas, sendo possível filtrá-las por tipo do ponto, nome do ponto de pesca, isca utilizada e espécie. Nesta tela, os usuários terão acesso ao formulário de cadastro de uma nova foto.

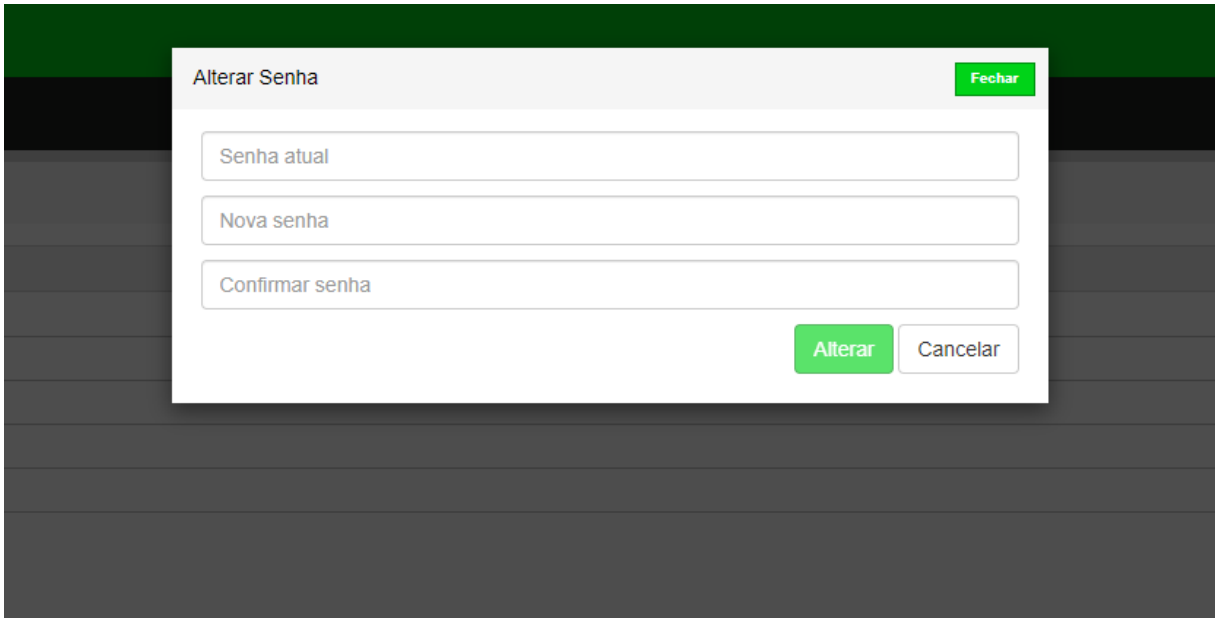
Figura 14 – Tela de galeria de fotos



Fonte: Autoria própria.

Os usuários com perfil de pescador e de administrador, poderão alterar seus dados no sistema. As Figuras 15 e 16 representam, respectivamente, as telas para alterar a senha e os dados cadastrais.

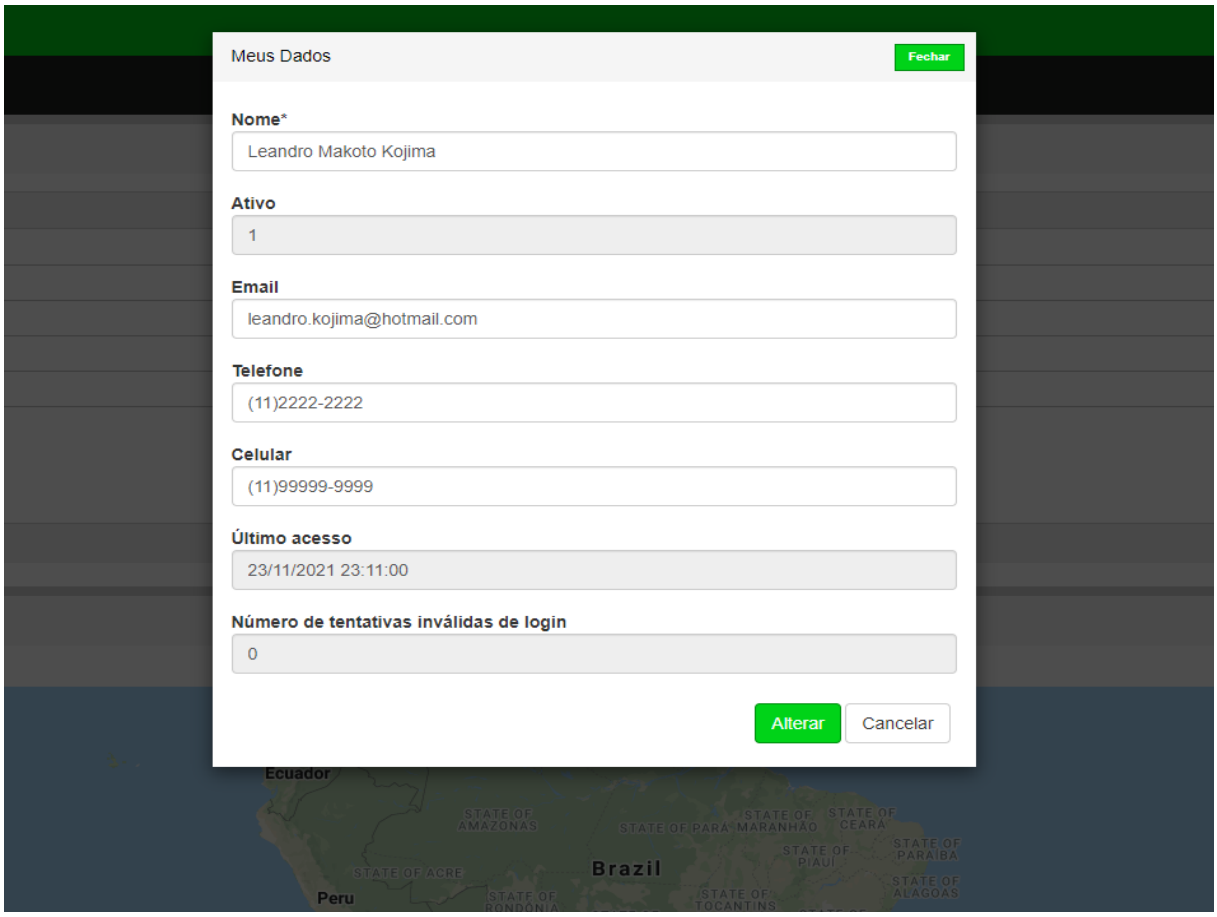
Figura 15 – Tela de edição de senha



Fonte: Autoria própria.

A Figura 16, representa o módulo de alteração de dados cadastrais do usuário, podendo ser editado. O campo denominado “ativo” registra se o usuário está ativo no sistema e o campo denominado “Número de tentativas inválidas de login” representa a quantidade de tentativas sequenciais de *login* sem êxito.

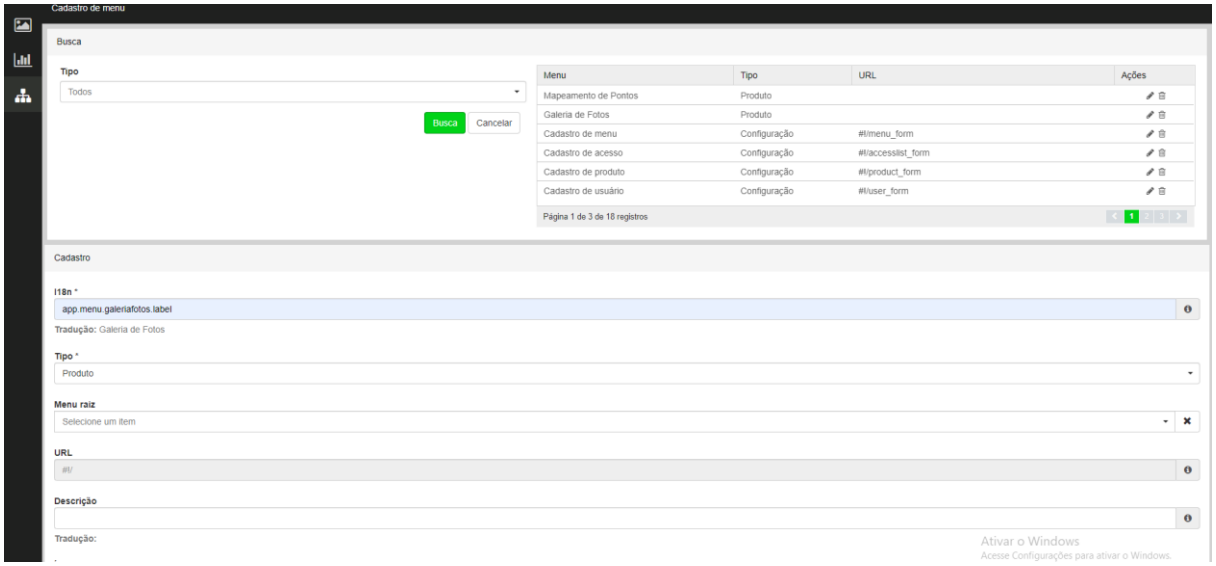
Figura 16 – Tela para edição de dados cadastrais



Fonte: Autoria própria.

Os menus do sistema foram implementados de uma maneira de fácil manutenção, nos quais são cadastrados menus e sub menus a partir de uma tela de cadastro no sistema que pode ser acessada apenas para o perfil administrador. Nesta tela, deve-se preencher os campos de código de tradução para o menu, o caminho de navegação da nova tela e selecionar o tipo do menu que está sendo cadastrado, se é menu de configuração ou menu de produto. Os menus de configuração são exibidos no ícone de engrenagem localizado no canto superior direito e os menus do tipo produto são localizados na lateral esquerda da tela. A tela de cadastro de menu é representada pela Figura 17.

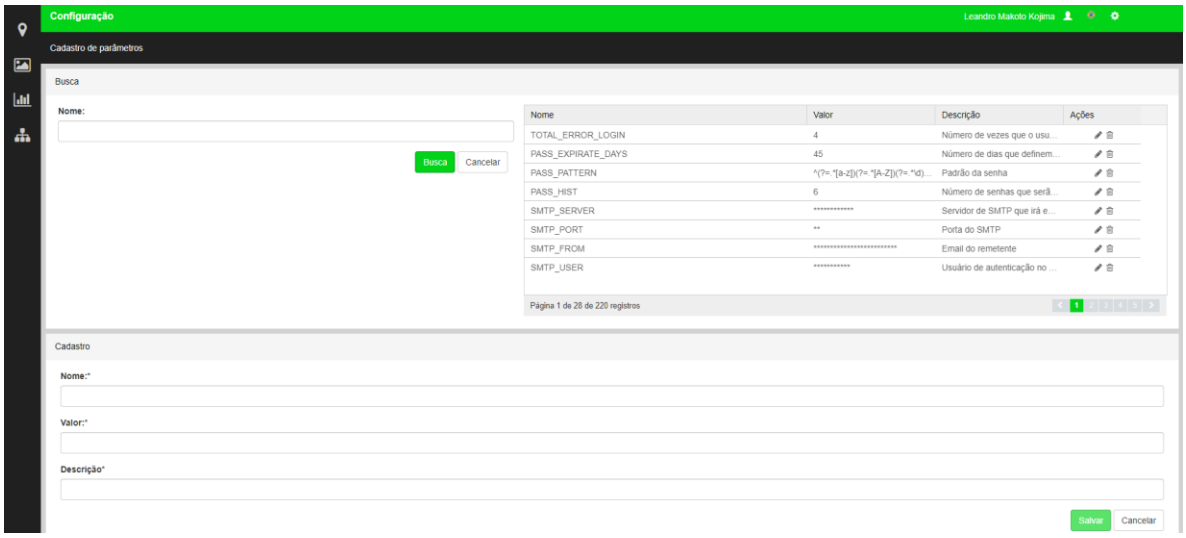
Figura 17 - Tela de cadastro de menu



Fonte: Autoria própria.

Os parâmetros no sistema são valores fixos de configuração do sistema e afins. São usados para resgatarem valores do banco de dados para serem utilizados em locais específicos. O exemplo de um valor usado nos parâmetros é o endereço de uma das APIs do Google Maps. A Figura 18, representa a tela de cadastro de um novo parâmetro.

Figura 18 - Tela de cadastro de parâmetros



Fonte: Autoria própria.

4.4 IMPLEMENTAÇÃO DO SISTEMA

O sistema foi desenvolvido no padrão *Model, View* e *Controller* (MVC), que separa as camadas visuais, lógicas e controladores.

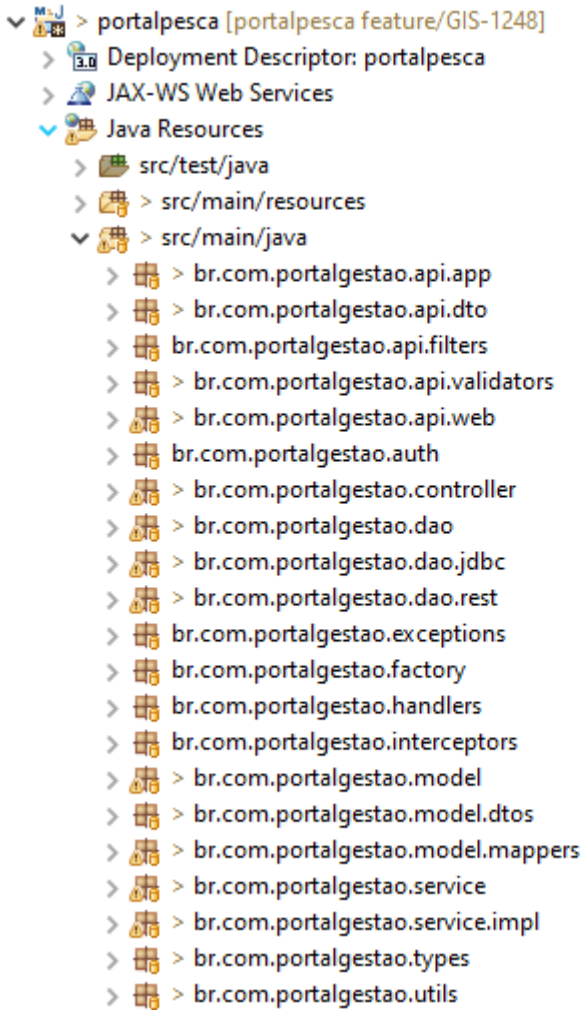
O desenvolvimento do sistema iniciou com a estrutura do banco de dados, seguindo o diagrama de entidade e relacionamentos. Para isso, foi utilizado o MySQL Workbanch para a manipulação do MySQL. Posteriormente, o sistema foi organizado em pacotes que armazenam as classes referentes ao modelo MVC, nas quais, o *model* contém as classes responsáveis por executar a lógica das regras de negócios, validações, inserção no banco de dados e classes de entidades. O pacote denominado *controllers* contém as classes responsáveis por transformar eventos gerados pela interface em ações de negócio, alterando o modelo e, por fim, a pasta *views* contém as classes responsáveis por criar as interfaces da aplicação.

4.2.1 *Model*

A organização dos pacotes da camada de modelo está apresentada na Figura 19. As pastas: *service, dao, jdbc, model, mappers e impl* são pastas pertencentes à camada de modelo.

Os pacotes denominados “*dao*” e “*service*", são interfaces que são implementadas pelas classes que estão dentro dos pacotes denominado “*jdbc*” (implementando as classes no pacote *dao*) e “*impl”* (implementando as classes no pacote *service*). O pacote denominado “*model”*, possui as classes de modelo dos objetos utilizados no sistema, como “*PointModel”*, que espelha informações sobre um ponto de pesca no banco de dados, e o pacote denominado “*mappers”* contém classes que foram usadas como *Data Transfer Object* (DTO) para a transferência de objetos entre as camadas.

Figura 19 – Organização das pastas.



Fonte: Autoria própria.

A seguir, serão apresentados alguns exemplos de código pertencentes à camada de modelo. A Listagem 1, representa o método que insere um ponto de pesca no banco de dados. Foi utilizado a classe *StringBuilder* para escrever o comando SQL responsável por inserir os dados no banco de dados e preparando o código para aceitar parâmetros que representam os dados a serem inseridos. A classe *MapSqlParameterSource* foi utilizada para receber os valores dos parâmetros a serem usados no comando SQL. O comando é executado quando a consulta está pronta e com os valores dos parâmetros preenchidos.

Listagem 1 - Método para salvar ponto de pesca.

```
@Override
public void save(PointModel model) throws ApplicationMessageException {
	StringBuilder query = new StringBuilder("");
```

```
        query.append( "INSERT INTO `pp_points` " );
        query.append( "( " );
        query.append( "`name`, " );
        query.append( "`contact`, " );
        query.append( "`guide_name`, " );
        query.append( "`guide_contact`, " );
        query.append( "`price`, " );
        query.append( "`opening`, " );
        query.append( "`closes`, " );
        query.append( "`add_id`, " );
        query.append( "`point_type` ) " );
        query.append( "VALUES ( " );
        query.append( ":name, " );
        query.append( ":contact, " );
        query.append( ":guide_name, " );
        query.append( ":guide_contact, " );
        query.append( ":price, " );
        query.append( ":opening, " );
        query.append( ":closes, " );
        query.append( ":add_id, " );
        query.append( ":point_type ) " );

        MapSqlParameterSource params = new MapSqlParameterSource();
        params.addValue("name", model.getPointName());
        params.addValue("contact", model.getContact());
        params.addValue("guide_name", model.getGuideContact());
        params.addValue("guide_contact", model.getGuideContact());
        params.addValue("price", model.getPrice());
        params.addValue("opening", model.getOpen());
        params.addValue("closes", model.getCloses());
        params.addValue("add_id", model.getAddress().getAddId());
        params.addValue("point_type", model.getPointType());

        KeyHolder keyHolder = new GeneratedKeyHolder();

        this.getNamedParameterJdbcTemplate().update( query.toString(),
params, keyHolder );

        model.setPntId((int)(long) keyHolder.getKey());

}
```

Fonte: Autoria própria.

Foi utilizada a biblioteca Jackson para serialização e tradução de objetos em forma de *Javascript Object Notation* (JSON) para a transmissão de dados entre as camadas, utilizado para montar um objeto em forma de JSON para alimentar o gráfico de espécies capturadas em determinado ponto. Na Listagem 2, é possível observar um exemplo da utilização da biblioteca, no método *getJsonChartEspecie*.

Listagem 2 – Método para gerar gráfico de espécies.

```
@Override
public ObjectNode getJsonChartEspecie(Integer pointId) throws
```

```
ApplicationMessageException, BusinessMessageException {
    ObjectMapper mapper = jacksonConverter.getObjectMapper();
    ObjectNode node = mapper.createObjectNode();

    List<ChartEspecieDTO> list = new ArrayList<ChartEspecieDTO>();
    list = graficosDAO.getSpeciesByPoint(pointId);

    ArrayNode arrayNodeList = mapper.createArrayNode();
    if(list != null) {
        for(ChartEspecieDTO model : list) {
            ObjectNode obj = mapper.createObjectNode();

            obj.put("label", model.getEspecie());
            obj.put("value", model.getQuantidade());

            arrayNodeList.add(obj);
        }
    }
    node.putPOJO( "data", arrayNodeList );
    return node;
}
```

Fonte: Autoria própria.

Para realizar o cadastro de um novo usuário, foram desenvolvidas algumas validações para garantir a singularidade de um novo acesso. Para isso, foram realizadas validações como: duplicidade de nome de usuário, duplicidade de *email,* campos obrigatórios e padrões de senhas. Na Listagem 3 o método *validateSave* é responsável por fazer as validações necessárias para salvar o usuário, caso contrário, é exibida uma mensagem de erro que informa o usuário sobre o erro ocorrido durante a tentativa de realizar o cadastro.

Listagem 3 - Método para validar um novo usuário

```
/*
* VALIDATORS - INICIO
*/
private void validateSave(UserModel user) throws ApplicationMessageException,
BusinessMessageException {

      if( user == null ) {
          throw new BusinessMessageException(this.messageSource.getMessage(
"app.exception.user.userinvalid", null, LocaleContextHolder.getLocale()));
      }

      // - Não pode haver usuários com o mesmo username ( duplicado )
      this.validateDuplicate( user );

      // - A senha deve seguir o padrão definido
      this.validatePassword( user );

      // - O nome deve estar preenchido
      this.validatePersonName( user.getPerson() );
```

```
        // - O email deve estar preenchido
        // - O email deve ser válido
        // - O email deve ser único
        this.validateEmail( user.getPerson() );

        // - O email deve estar preenchido
        // - O email deve ser válido
        this.validatePhones( user.getPerson() );

        //  - O tipo de usuário deve estar preenchido
        this.validateUserType( user.getUserType() );

        //  - A lista de acesso deve estar preenchida
        this.validateAccessList( user.getAccessListModel() );
}
```

Fonte: Autoria própria.

O usuário possui algumas configurações que são salvas no banco de dados, tais como: idioma que selecionou e modo de visão. Sempre que o usuário alterar alguma dessas configurações, o método *updateUserConfig* é invocado para atualiza as configurações padrões do usuário, conforme é mostrado na Listagem 4.

Listagem 4 - Método para atualizar as configurações padrões do usuário

```
@RequestMapping(value = "/protected/updateUserConfig", method =
RequestMethod.POST)
public @ResponseBody void updateUserConfig(@RequestBody ModelMap userConfig)
throws ApplicationMessageException, BusinessMessageException {
        Object langParam = userConfig.get("lang");
        Object layoutParam = userConfig.get("layout");


        try {
            UserConfigModel retrievedUserConfig =
userConfigService.getByUserId(this.getUserProfile().getUser().getUsrId());

            if (langParam != null) {
                String lang = (String) langParam;
                if (!lang.isEmpty()) retrievedUserConfig.setLang(lang);
            }

            if (layoutParam != null) {
                String layout = (String) layoutParam;
                if (!layout.isEmpty()) retrievedUserConfig.setLayout(layout);
            }

            this.userConfigService.update(retrievedUserConfig,
this.getUserProfile());

        } catch (ApplicationMessageException | BusinessMessageException |
```

```
NullPointerException e) {
            throw e;
        }
    }
```

Fonte: Autoria própria.

Antes do ponto de pesca ser inserido no banco de dados, o endereço do local foi adicionado antecipadamente, utilizando uma *Application Programing Interface* (API) do Google para buscar, por meio das coordenadas, latitude e longitude, o endereço do local selecionado no mapa quando é cadastrado um novo ponto de pesca. A Listagem 5, exibir a utilização da API no método *save.*

Listagem 5 - Método para salvar endereço do ponto de pesca.

```
@Override
public void save(PointModel model, UserProfileDTO userProfile)
      throws ApplicationMessageException, BusinessMessageException {

      try {
            String url =
parameterService.getByName(”GOOGLE_MAPS_API_LATLNG”).getValue() +
model.getLatitude() + ”,” + model.getLatitude();

            HttpClient httpclient = new DefaultHttpClient();
            HttpGet request = new HttpGet(url);
            InputStream in = httpclient.execute(request).getEntity()
                        .getContent();

            BufferedReader br = null;
            StringBuilder sb = new StringBuilder();

            br = new BufferedReader(new InputStreamReader(in));

            String line = br.readLine();

            while (line != null) {
                  sb.append(line);
                  line = br.readLine();

            }
            ObjectMapper mapper = jacksonConverter.getObjectMapper();

            ObjectNode json=(ObjectNode) ObjectMapper().readTree(sb.toString());

            AddressModel adr = new AddressModel();
            adr.setStrAdr(json.findValue("formatted_address").asText());
            adr.setLatitude(model.getLatitude());
            adr.setLongitude(model.getLongitude());


            this.adrDAO.save(adr);
```

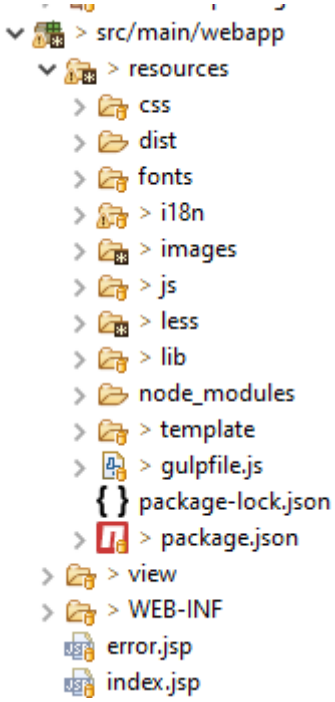
Fonte: Autoria própria.

4.2.2 *View*

A camada *view* contém as classes responsáveis pela interação do aplicativo com o usuário, as quais recebe a entrada de dados fornecidas pelo usuário e a saída desses dados na tela. Nessa camada, o usuário vê o estado do modelo e pode manipular a interface.

A organização das pastas no sistema foi separada das camadas *model* e *controller,* estando dentro da pasta *webapp* que se encontram arquivos de estilização da página, *templates,* arquivos *javascript,* arquivos de tradução, entre outros. A Figura 20 representa a estrutura de pastas da camada *view.*

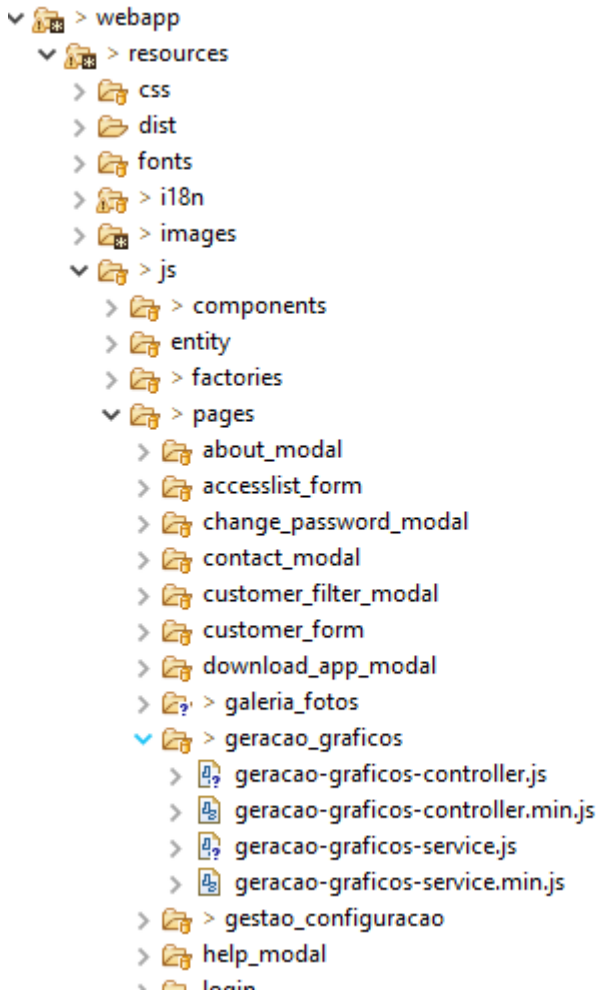
Figura 20 – Estrutura de pastas da camada view.



Fonte: Autoria própria.

O pacote denominado “*pages”* possui uma série de pacotes que contém arquivos com código JavaScript responsáveis por controlar as ações do usuário e fazem requisições para a camada *controller.* Cada pacote dentro de *pages* representa uma tela na aplicação, como mostra na Figura 21.

Figura 21 – Estrutura de pastas das páginas



Fonte: Autoria própria.

As páginas que o usuário irá interagir contém funções para exibição de dados na tela, requisições de cadastro, requisições de roteamento para navegar entre as páginas e outros tipos de ações da página. Neste projeto, isso é controlado por controladores e serviços da camada *view.* A Listagem 6, exibe um método para carregar a quantidade de pontos marítimos/fluviais e pontos pesque e pague no qual *PPMapeamentoPontosController* é o nome do controlador e *PPMapeamentoPontosService* é o nome do serviço.

Listagem 6 - Método para carregar total de pontos

```
angular.module('main').controller('PPMapeamentoPontosController',
[
'$scope',
'$translate',
'PNotifyFactory',
```

```
'$interval',
'$rootScope',
'$state',
'$filter',
'$window',
'$modal',
'AccessRestrictionService',
'PPMapeamentoPontosService',
function (
$scope,
$translate,
PNotifyFactory,
$interval,
$rootScope,
$state,
$filter,
$window,
$modal,
AccessRestrictionService,
PPMapeamentoPontosService) {
      PPMapeamentoPontosService.loadTotalPoints().then(function (data) {
        $scope.pontosMaritimosFluviais = data.pontosMaritimos;
         $scope.pontosPesquePague = data.pontosPesquePague;
    });
```

Fonte: Autoria própria.

O controlador da camada *view* é quem chama um método do serviço, que é o responsável para realizar a requisição para a camada *controller.* É possível observar um exemplo de uma requisição do serviço na Listagem 7 utilizando *RequestMapping* na camada de controle.

Listagem 7 - Método de requisição para a camada de controle

```
this.loadTotalPoints = function() {
    var url = $browser.baseHref() +
'protected/api/web/ppMapeamentoPontos/getTotalPoints/';
    return manager.httpGet(url, {}, {}).then(function (response) {
        return response.data;

    });
}
```

Fonte: Autoria própria.

Foi utilizado a *framework Fusioncharts* para a construção dos gráficos, o qual possibilita utilizar opções para customizar o leiaute, as ações e o tamanho. O uso

desse *framework* é dividido em duas partes, sendo: a primeira, contida na Listagem 8, é a implementação do *template* a ser exibido o gráfico e, a segunda, mostrado na Listagem 9, contém a estrutura do gráfico e suas opções de renderização. As Listagens 8 e 9 representam a estrutura do gráfico do tipo pizza de quantidade de espécies capturadas em um determinado ponto de pesca.

Listagem 8 - Estrutura do gráfico de quantidade por espécie

```
$scope.quantityBySpecie = {
                id: "qttyBySpecie",
                type: "pie2d",
                width: "100%",
                height: "309px",
                dataFormat: "json",
                dataSource: {
                    chart: {
                        numberPrefix: "",
                        showLabels: "0",
                        showBorder: "0",
                        bgColor: "#ffffff",
                        showShadow: "0",
                        canvasBgColor: "#ffffff",
                        showCanvasBorder: "0",
                        useRoundEdges: "0",
                        showAlternateHGridColor: "0",
                        showAlternateVGridColor: "0",
                        use3dlighting: "0",
                        showValue: "1",
                        showPercentValues: "1",
                        showPercentInTooltip: "0",
                        decimals: "2",
                        useDataPlotColorForLabels: "1",
                        showLegend: "1",
                        enableMultiSlicing: "0",
                        pieRadius: "80",
                        smartLabelClearance: "0",
                        skipOverlapLabels: "0",
                        theme: "portalpesca",
                        animation: "1"
                    }
                }
};
```

Fonte: Autoria própria.

A Listagem 9, representa a implementação do FusionCharts no código HTML. Conforme exibido na Figura 8, a variável denominada “$scope.quantityBySpecie” é usada para recuperar dados essenciais para a interpretação do código HTML. Quando a tela é acessada, o próprio navegador interpreta o componente e monta a estrutura em tempo de execução.

Listagem 9 - Uso do Fusioncharts no template

```
<fusioncharts id="{{quantityBySpecie.id}}"
type="{{quantityBySpecie.type}}"
width="{{quantityBySpecie.width}}"
height="{{quantityBySpecie.height}}"
datasource="{{quantityBySpecie.dataSource}}"
initialized="onInitialized(chart)">
</fusioncharts>
```

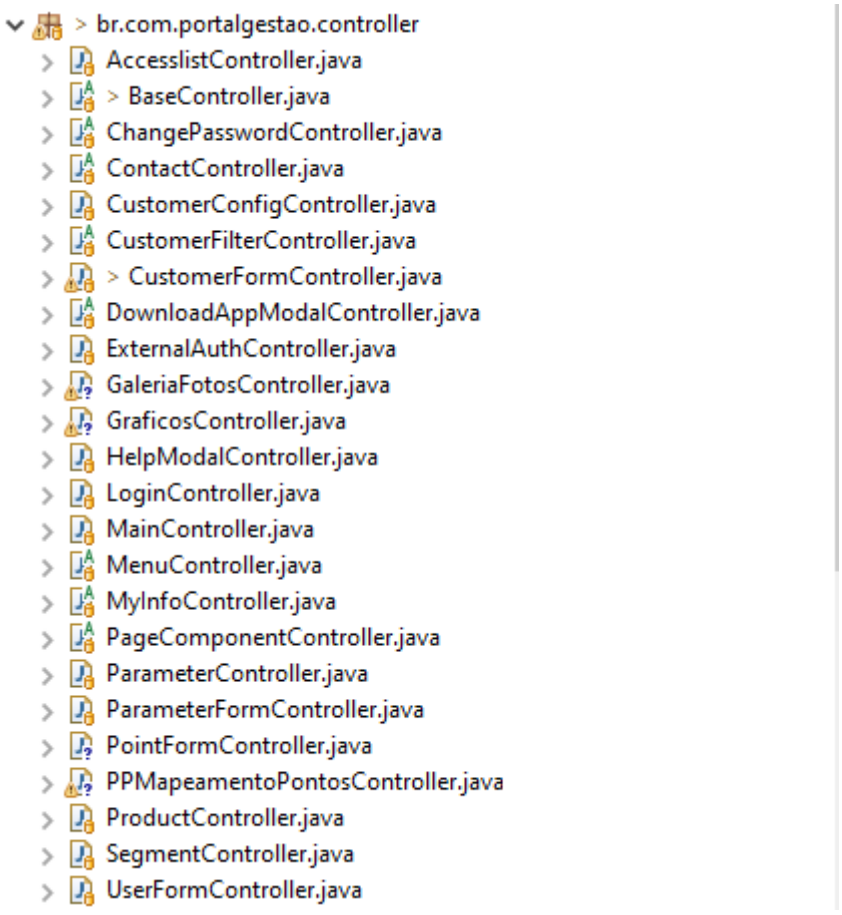
Fonte: Autoria própria.

4.2.3 Controller

A camada *controller* é intermediária entre as camadas *model* e *view* e é responsável pelas requisições realizadas pelo usuário. Essa camada, geralmente retorna algum tipo de resposta para a camada *view,* podendo ser retorno de dados e mensagens informando se a operação realizada foi bem sucedida ou se houve algum erro.

O pacote denominado "*controller",* exibido na Figura 22, é responsável por armazenar os arquivos controladores.

Figura 22 – Estrutura de pastas da camada controller



Fonte: Autoria própria.

O uso de *RequestMapping* no projeto foi utilizado nos *controllers* para manipular requisições do tipo *get* e *post* usadas na aplicação, porém suporta também os métodos *put*, *delete* e *patch*. No sistema, a camada controladora foi usada para intermediar requisições que o usuário faz, por meio da camada *view.* Na Listagem 10 pose-se ver o método *save,* que retorna o status da requisição, se ela foi realizada com êxito ou não quando salvamos um novo usuário no banco de dados.

Listagem 10 - Método save para cadastro de usuário

```
@RequestMapping(value="/save", method=RequestMethod.POST)
public ResponseEntity<Boolean> save(@RequestBody UserModel user) throws
ApplicationMessageException, BusinessMessageException {
	service.save( user, this.getUserProfile() );
	return new ResponseEntity<Boolean>(Boolean.TRUE, HttpStatus.OK);
}
```

Fonte: Autoria própria.

Para carregar os pontos de pesca no mapa para a página inicial foi implementado o método *loadMap* que retorna um objeto do tipo JSON que contém os dados para os marcadores serem exibidos no mapa, mostrado na Listagem 11.

Listagem 11 - Método para carregar os pontos do mapa

```
@RequestMapping(value = "/loadMap", method = RequestMethod.GET)
public ObjectNode loadMap() throws ApplicationMessageException,
BusinessMessageException, Exception {

	return ppMapeamentoPontosService.getPointToMap();
}
```

Fonte: Autoria própria.

5 CONCLUSÃO

O objetivo deste trabalho foi disponibilizar informações sobre pontos de pesca para pescadores iniciantes e experientes, para que conheçam e compartilhem novos pontos de pesca para estarem mais preparados para realizar uma pescaria.

O sistema implementado possibilita que todos os tipos de pesca esportiva sejam compartilhados, sendo eles, marítimos, fluviais e pesqueiro, podendo extrair informações sobre a localização do ponto de pesca, espécies encontradas, iscas mais utilizadas e dicas sobre o ponto de pesca.

A solução pretendida foi desenvolvida como uma aplicação *web* em linguagem *server-side* Java, AngularJS para controlar as ações do usuário, Tomcat como servidor do sistema, MySQL como banco de dados relacional para armazenamento de dados, Google Maps para mostrar geograficamente as localizações dos pontos de pesca e cadastro dos mesmos, IDE Eclipse para o desenvolvimento do sistema. As funcionalidades e implementações atendem aos requisitos levantados. A maior dificuldade encontrada durante o desenvolvimento do sistema foi em relação aos mapas utilizados, porém, com o auxílio de materiais e documentação foi possível desenvolver essa funcionalidade.

Como trabalhos futuros, pretende-se adicionar novas funcionalidades como: disponibilizar loja virtual para a venda de acessórios e equipamentos de pesca, adicionar clientes e equipes de pesca, criar diferentes tipos de produtos no sistema e limitando o acesso aos diferentes tipos de usuário, adicionar a funcionalidade de cadastro, na qual qualquer pessoa pode se cadastrar ao sistema e o administrador pode validar o cadastro. Além disso, pretende-se criar uma versão para ser utilizada em dispositivos móveis.

REFERÊNCIAS

ANGULARJS. **What is AngularJS?.** Disponível em https://docs.angularjs.org/guide/introduction. Acesso em: 07 jul. 2021.

AUSTRALIA. **Recreational Fishing Advisory Committee. Recreational fishing in Australia** - 2011 and beyond: a national industry development strategy. Commonwealth of Australia, jun. 2011.

FAO. **Technical Guidelines for Responsible Fisheries**: Recreational Fisheries. Roma: FAO, 2012.

FREIRE, K. M. F.; MACHADO, M.L.; CREPALDI, D. **Overview of Inland Recreational Fisheries in Brazil**. Fisheries, Abingdon, v. 37, n. 11, p. 484-494. nov. 2012.

FREIRE, K. M. F.; TUBINO R. A; MONTEIRO-NETO, C.; ANDRADE-TUBINO, M. F.; BELRUSS, C.G.; TOMAS, A. R. G.; TUTUI, S. L. S.; CASTRO, P. M. G.; MARUYAMA, L.S.; CATELLA, C. A.; CREPALDI, D.V.; DANIEL, C. R. A.; MACHADO, M. L.; MENDONÇA, J. T.; MORO, P. S.; MOTTA, F. S.; RAMIRES, M.; SILVA, M. H. C.; VIEIRA, J. P. **Brazilian recreational fisheries**: current status, challenges and future direction. Fisheries Management and Ecology, Oxford, v. 23, n. 3-4, p 276-290, jun-ago. 2016.

HANAZAKI, N.; CASTRO, F.D.; OLIVEIRA, V.G.; PERONI, N. **Between the sea and the land: the livelihood of estuarine people in southeastern Brazil**. Ambiente e Sociedade, Campinas, v. 10, n. 1, p 121-136. jan.-jun. 2007.

LANG,T. **The Role of Hatcheries in Ensuring Social and Economic Benefits of Fisheries in the United States**. Fisheries, Abingdon, v. 39, n. 11, p 556- 557,nov. 2014

LEWIN, W.C.; MCPHEE, D.P & ARLINGHAUS, R. Biological Impacts of Recreational Fishing Resulting from Exploitation, Stocking and Introduction In: **Aas, Ø. Global Challenges in Recreational Fisheries**. Oxford: Blackwell Scientific Publications, p. 75-92. 2008

MDN Web Docs. **What is Javascript**. Disponível em https://developer.mozilla.org/en-US/docs/Web/JavaScript/Guide/Introduction#what_is_javascript

MUNDO EDUCAÇÃO. **Zonas litorâneas do Brasil**. Disponível em https://mundoeducacao.uol.com.br/geografia/zonas-litoraneas-brasil.htm. Acesso em: 03 jun. 2021.

ORACLE. **Implantar o apache Tomcat na computação Ampere A1 baseada em Armas conectada a um banco de dados autônomo**. Disponível em https://docs.oracle.com/pt-br/solutions/deploy-tomcat-adb/index.html#GUID-A54F48CD-F9BB-4980-9A50-60ACE2074351. Acesso em: 07 jul 2021.

PARKKILA, K., ARLINGHAUS, R., ARTELL, J., GENTNER, B., HAIDER, W., AAS, BARTON, D., ROTH, E. & SIPPONEN, M.; Methodologies for assessing socio-economic benefits of European inland recreational fisheries. **EIFAC Occasional Paper** n°. 46. Ankara, FAO. 2010

RAMIRES, M.; BARRELLA, W. **Ecologia da Pesca Artesanal em Populações Caiçaras da Estação Ecológica Jureia-Itatins**, São Paulo, Brasil. Interciência, Caracas, v. 28, n. 4, p. 208-213. abr. 2003.