

ARTEFATOS DO PROJETO DE SOFTWARE - CRIAÇÃO DE CAMARÃO EM CATIVEIRO COM MONITORAMENTO IOT PARA GASTRONOMIA

SUMÁRIO

DIAGRAMA DE CASO DE USO	3
DIAGRAMA DE CLASSE	5
DIAGRAMA DE OBJETOS	6
DIAGRAMA DE REDES	8
MODELO DE NEGÓCIOS CANVAS	9
MODELO LÓGICO, CONCEITUAL E FÍSICO DO BANCO DE DADOS	10
SCRIPT SQL	14
ANÁLISE DE RECORRÊNCIA DO ALGORITMO MERGE SORT	21
DIÁRIO DE BORDO	23

DIAGRAMAS UML

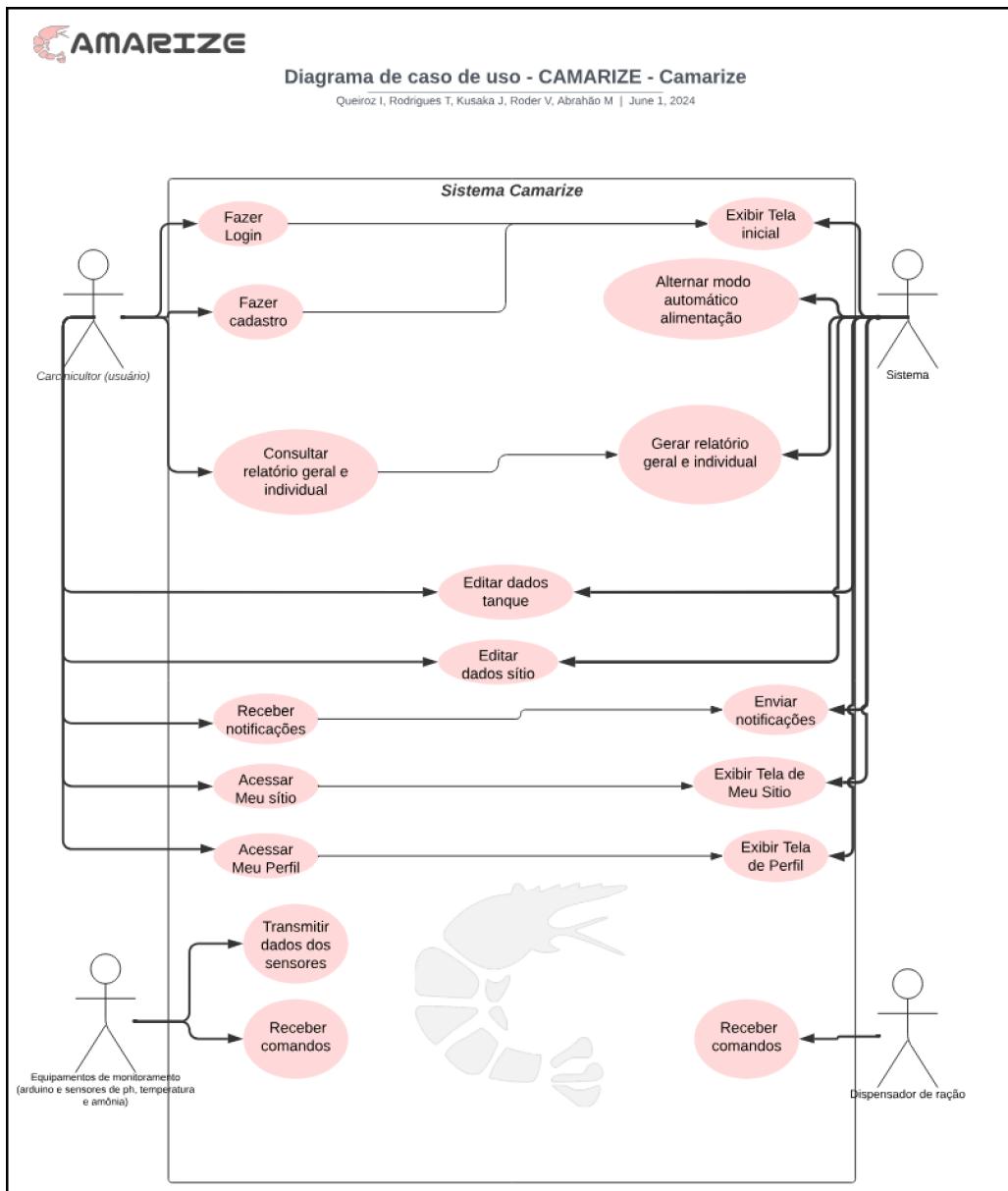
Nesta seção serão apresentados os diagramas da UML utilizados para a modelagem do sistema desenvolvido. Dentre os diagramas utilizados, pode-se citar: Diagrama de Caso de Uso e Diagrama de Redes.

DIAGRAMA DE CASO DE USO

O Diagrama de Caso de Uso foi desenvolvido para demonstrar as funcionalidades do sistema. O Carcinicultor irá fornecer ao Sistema suas informações para realizar o cadastro/login e o sistema deverá fornecer a tela inicial. Dentro do sistema, o carcinicultor deverá editar dados referentes ao tanque e do sítio em que estão os tanques. Além disso, o sistema fornecerá ao carcinicultor o relatório geral e individual de cada um dos tanques que irá contar com temperatura, pH e amônia.

Enquanto o carcinicultor utilizar o sistema, ele receberá notificações a respeito dos tanques e dos camarões, verificar informações o sítio e de seu peril.

Figura 1 – Diagrama de caso de uso



Fonte: Autoria Própria

Os equipamentos de monitoramento irão transmitir os dados de cada sensor e receberão comandos. O dispensador de ração irá receber o comando para liberar a ração para os camarões

DIAGRAMA DE CLASSE

O diagrama de classes define a estrutura e os relacionamentos necessários para monitorar e otimizar as condições de criação de camarões. Ele inclui classes e associações projetadas para gerenciar dietas, parâmetros ambientais, tipos de camarão e sensores, garantindo operações eficientes e integradas.

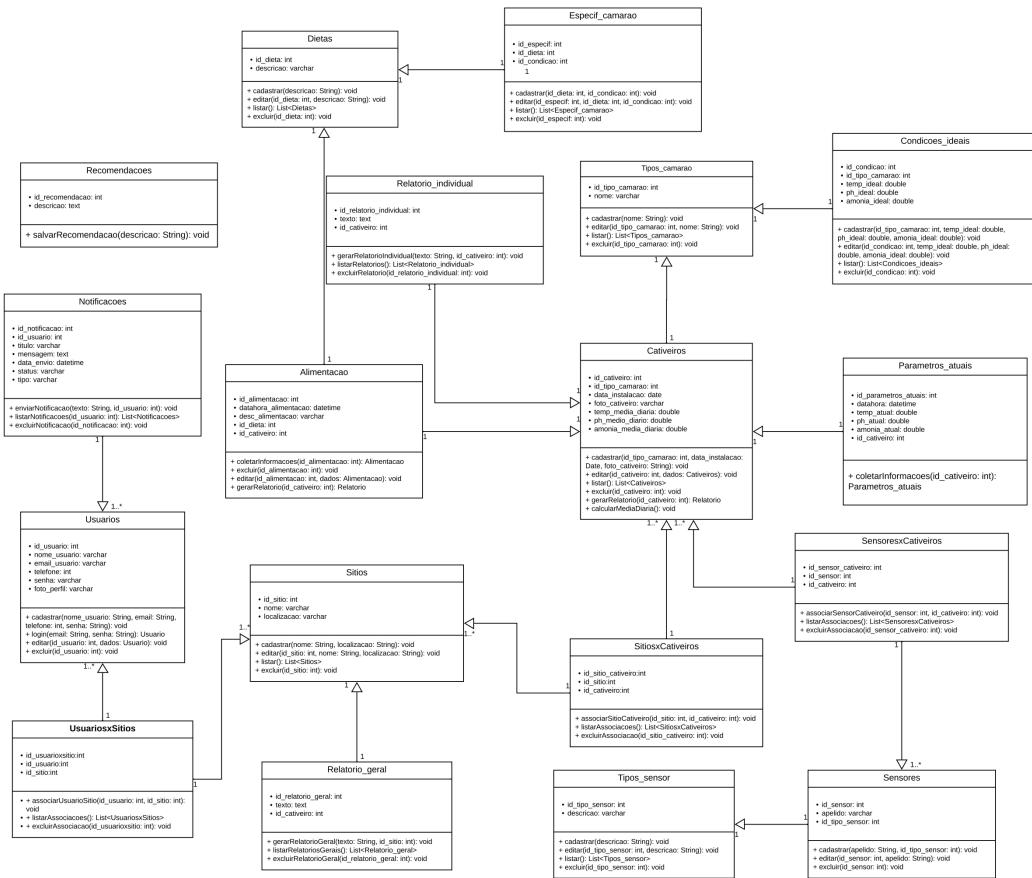
Entre as principais classes, **Tipos_camarao** categoriza espécies de camarões por meio de um identificador único e um nome. **Dietas** gerencia as alimentações ideais para cada espécie, enquanto **Condições_ideais** define parâmetros ambientais recomendados, como temperatura, pH e amônia, associados a cada tipo de camarão. **Parâmetros_atuais** registra as condições ambientais em tempo real de cada cativeiro, e **Especif_camarao** conecta condições ideais às dietas recomendadas. Já **Cativeiros** representa as instalações de criação, relacionando aos tipos de camarão e armazenando dados como médias diárias de temperatura e pH.

O sistema também inclui classes como **Alimentação**, que documenta horários e detalhes nutricionais, vinculando as dietas aos cativeiros. **Sítios** registra informações sobre locais onde estão os cativeiros, enquanto **Sensores** monitoram dados ambientais e estão associados aos cativeiros. Usuários gerenciam o sistema por meio de autenticação e dados pessoais, e os relatórios individuais e gerais detalham os desempenhos dos cativeiros com base nos dados coletados.

Os relacionamentos são fundamentais no sistema. Por exemplo, cada cativeiro está associado a um único tipo de camarão, enquanto um tipo de camarão pode estar ligado a vários cativeiros. Os **Parâmetros_atuais** monitoram condições específicas de cativeiros, e a alimentação é registrada de forma vinculada a dietas e cativeiros específicos. Além disso, **Sítios** podem agregar múltiplos cativeiros, enquanto os sensores mapeiam as condições de cativeiros individuais. Usuários gerenciam sítios específicos e recebem notificações com recomendações baseadas nos dados monitorados.

O uso de multiplicidade (como `1..*`, `0..1`) garante a flexibilidade e consistência dos relacionamentos, permitindo que o sistema opere de forma robusta e organizada. Este diagrama reflete um design completo, garantindo o monitoramento eficiente das condições ambientais, o controle das dietas e o gerenciamento dos cativeiros de camarões.

Figura 2 – Diagrama de classe



Fonte: Autoria Própria

DIAGRAMA DE OBJETOS

O diagrama de objetos apresenta uma visão prática do sistema de gerenciamento de cativeiros, demonstrando as instâncias específicas das classes e os valores associados em um momento determinado. Ele detalha o estado de cada entidade e suas interações, facilitando a compreensão da configuração atual.

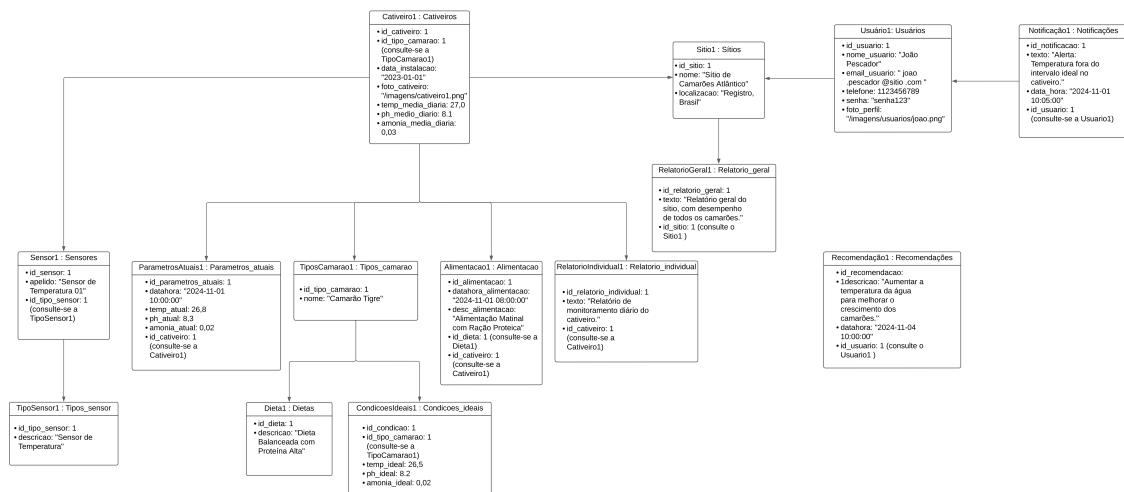
Entre os objetos principais, temos o Sensor1, que monitora parâmetros ambientais como temperatura, pH e amônia, categorizado por TipoSensor1, que descreve seu tipo, como "Sensor de Temperatura". Esses dados são registrados em ParametrosAtuais1, com valores como temperatura de 26,8°C, pH de 8,3 e amônia de 0,02. O Cativeiro1 armazena informações detalhadas sobre o

local onde os camarões, como o "Camarão Tigre" descrito por TipoCamarao1, são criados, incluindo médias diárias de parâmetros ambientais e dados de instalação. Esse cativeiro faz parte do Sítio1, chamado "Sítio de Camarões Atlântico", localizado em Registro, Brasil, que é gerenciado por um usuário, como João Pescador, representado pelo Usuario1.

O sistema também gerencia atividades como Alimentação1, associada a uma dieta específica (Dieta1), como "Dieta Balanceada com Proteína Alta", projetada para otimizar o crescimento dos camarões. CondicoesIdeais1 define os parâmetros ambientais ideais, enquanto RelatorioGeral1 e RelatorioIndividual1 registram informações detalhadas sobre os cativeiros, fornecendo uma visão ampla ou específica para análise e gestão. Recomendacoes1 oferece sugestões práticas, como ajustes na temperatura da água, baseadas nos dados coletados. Notificação1 alerta os usuários sobre situações críticas, garantindo uma resposta rápida para manter as condições ideais.

O diagrama demonstra como os sensores coletam dados que, integrados ao sistema, geram relatórios, notificações e recomendações para o manejo eficiente dos cativeiros, promovendo o bem-estar dos camarões criados. Ele destaca a importância das associações entre os objetos para garantir que todas as atividades e parâmetros sejam monitorados e ajustados de forma precisa.

Figura 3 – Diagrama de objetos



Fonte: Autoria Própria

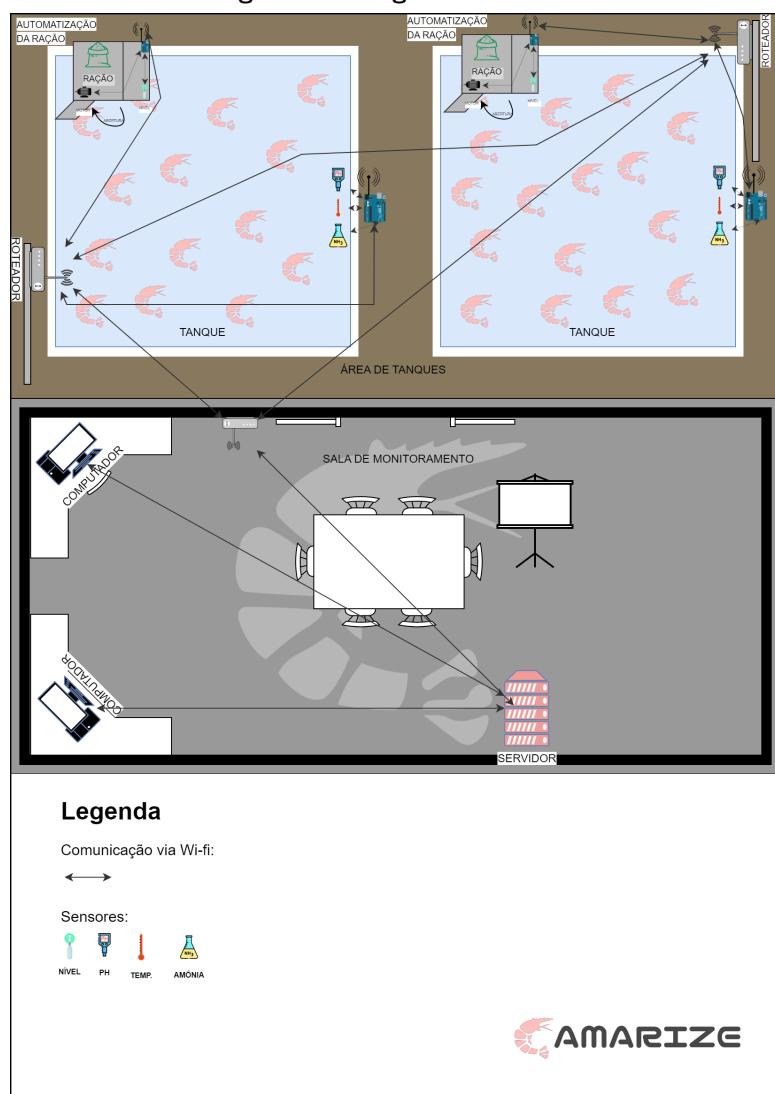
DIAGRAMA DE REDES

O Diagrama de Redes foi desenvolvido com o objetivo de ilustrar o funcionamento do projeto Camarize, um Sistema de Monitoramento de Cativeiro de Camarões para Fins Gastronômicos.

Na etapa de monitoramento, cada tanque estará equipado com um Arduino conectado a três sensores: medidor de temperatura, de pH e amônia. Que se mostram fundamentais para a avaliação da salubridade da água do cativeiro para os camarões. O dispositivo transmitirá continuamente os dados dos sensores por meio do Wi-Fi para o servidor central encontrado na sala de monitoramento.

Na etapa de automatização da alimentação, será utilizado um dispensador de ração com uma abertura na parte inferior, equipado com um motor para controle da abertura e um sensor de nível com os parâmetros pré-definidos anteriormente para determinar a quantidade ideal de ração a ser liberada no tanque. Em que, ambos os componentes estarão conectados ao Arduino, também se comunicando via Wi-Fi com o servidor central da sala de monitoramento. Os dados serão então compartilhados com os computadores que têm acesso à plataforma.

Figura 4 – Diagrama de Rede



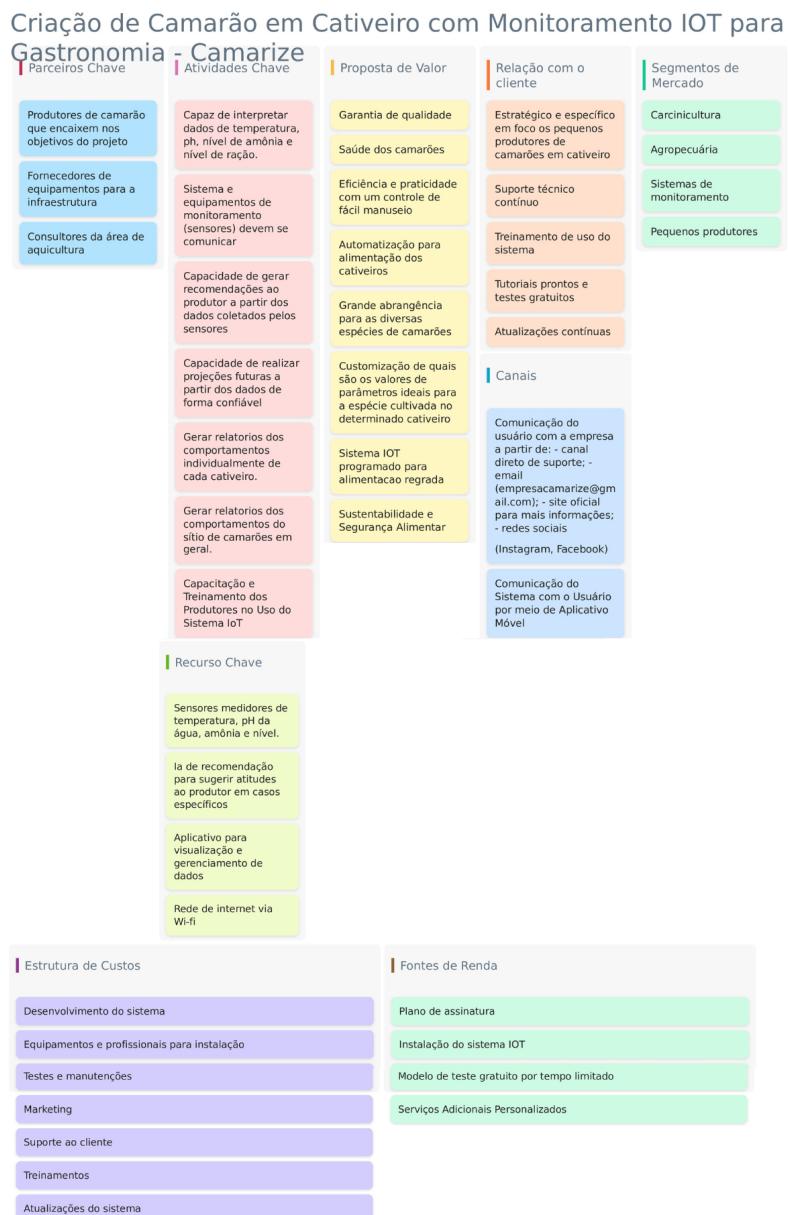
Fonte: Autoria Própria

É essencial pontuar que a comunicação entre os elementos ocorrerá via Wireless, que pode ser amplificada com uma maior quantidade de roteadores a depender da localização.

MODELO DE NEGÓCIOS CANVAS

O modelo de negócios Canvas foi utilizado para análise econômica do sistema. Por meio do Canvas foi possível identificar os Parceiros, Recursos e Atividades Chaves, nossas Propostas de Valor, Segmento de Mercado, nossa Fonte de Renda e Estrutura de Custos.

Figura 5 – Canvas



Fonte: Autoria Própria

Foi realizada uma análise completa e incluído os possíveis dados.

MODELO LÓGICO, CONCEITUAL E FÍSICO DO BANCO DE DADOS

A modelagem conceitual e lógica demonstra a funcionalidade do programa e do que será aplicado no banco de dados. A visão gráfica do banco de dados se demonstra algo fundamental, por conta que proporciona o máximo de detalhes com extrema clareza, apresentando as relações entre as entidades.

No modelo conceitual, a entidade Cativeiros representa os tanques, com dados como data_instalacao e temp_media_diaria. Ela está associada a Tipos_camarao, que define a espécie criada, e a Condições_ideais, que especifica parâmetros recomendados, como temp_ideal e ph_ideal.

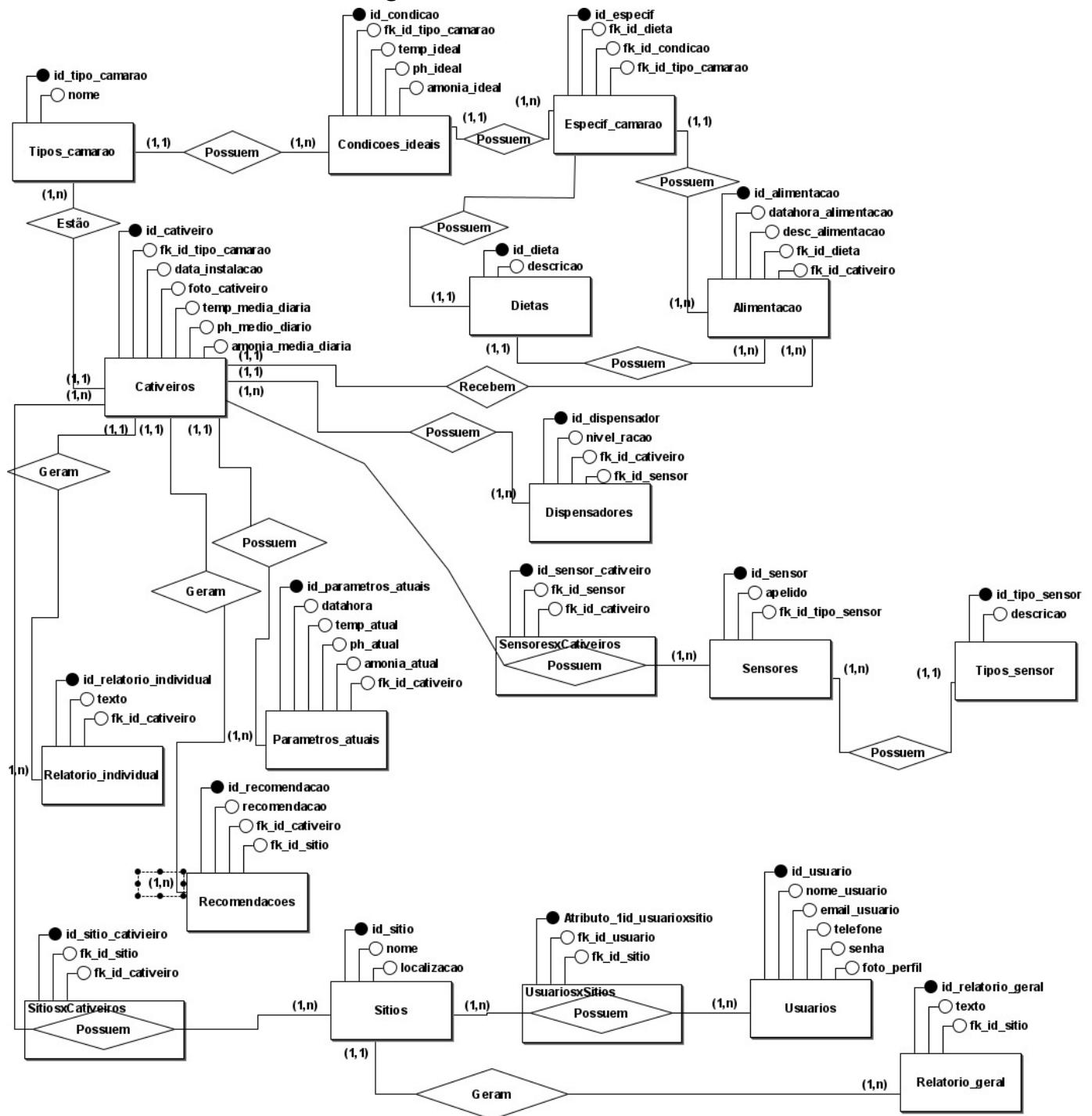
Sensores, registrados em Sensores, monitoram os cativeiros, coletando dados armazenados em Parametros_atuais, como temp_atual e ph_atual. A alimentação é controlada por Dietas e eventos específicos registrados em Alimentacao. Cativeiros estão localizados em Sitos, com atributos como nome e localizacao, e geram relatórios gerais em Relatorio_geral.

Usuários, descritos em Usuarios, têm acesso aos dados dos cativeiros e recebem notificações e recomendações por meio das entidades Notificacoes e Recomendacoes. Os relacionamentos entre as entidades integram dados de sensores, alimentação, localização e usuários para garantir o controle eficiente do ambiente de criação.

No modelo lógico, é apresentado uma representação mais concreta do banco de dados, demonstrando as tabelas, entidades, chaves primárias e estrangeiras e seus relacionamentos.

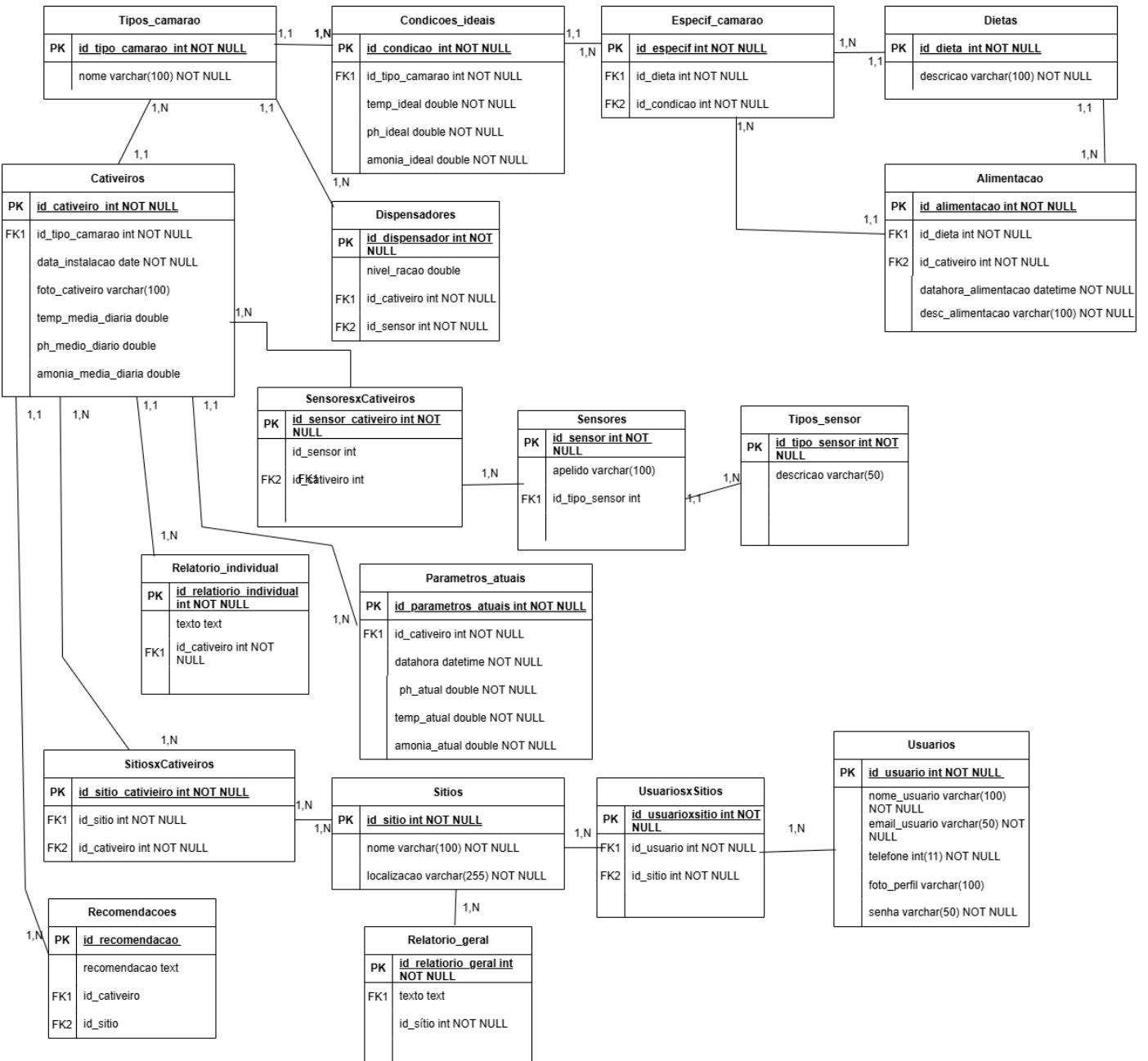
No modelo físico, demonstra uma versão refinada do modelo conceitual, representando restrições de dados, nomes de entidades e relacionamentos para implementação de forma independente.

Figura 6 – Modelo Conceitual



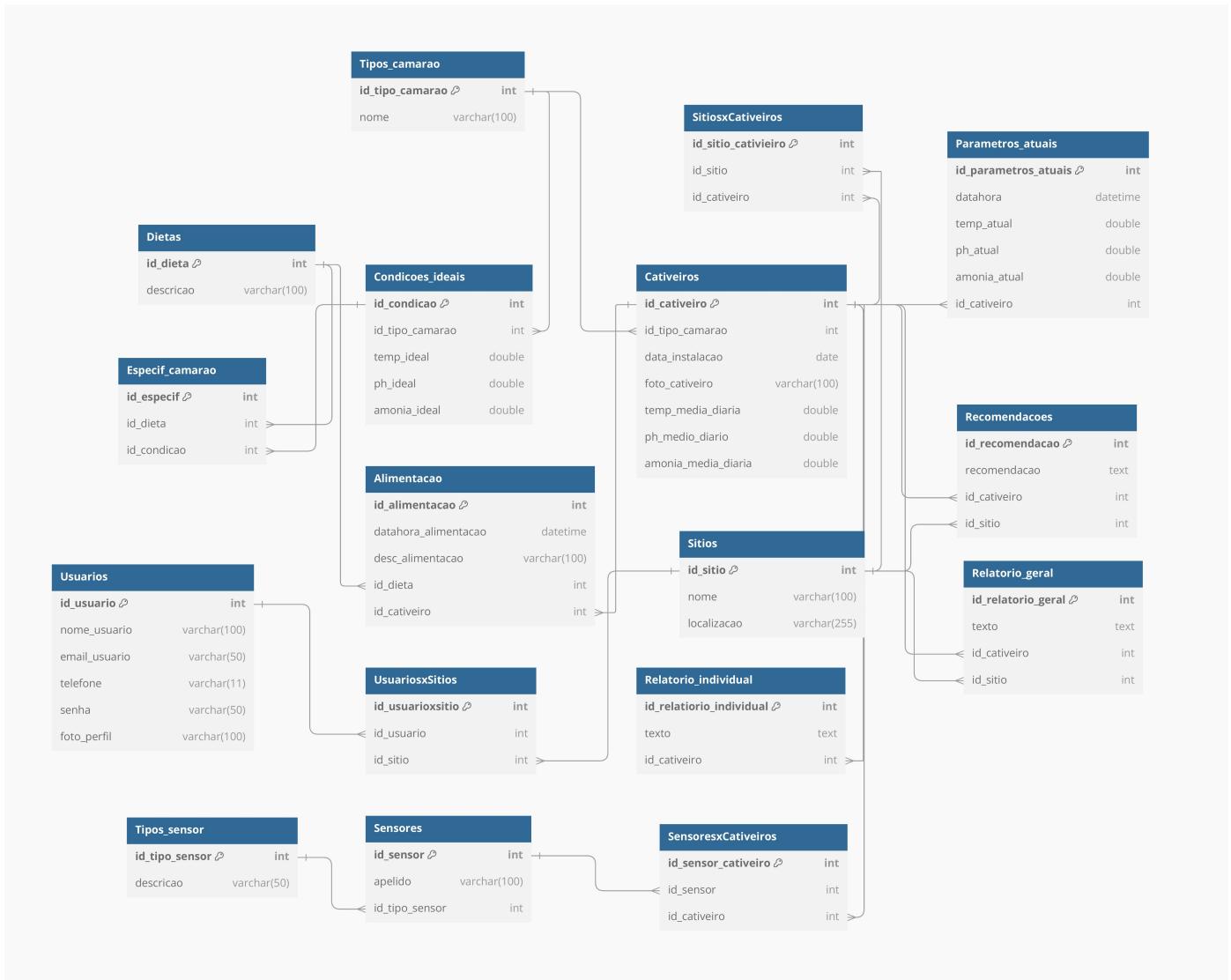
Fonte: Autoria Própria

Figura 7 – Modelo Lógico



Fonte: Autoria Própria

Figura 8 – Modelo Físico



Fonte: Autoria Própria

O banco de dados tem um papel de extrema importância para as informações, sendo necessário para guardar, organizar e recuperar dados. O Modelo Lógico e Conceitual são maneira de demonstrar a eficiência de um banco de dados.

SCRIPT SQL

O Script SQL foi baseado no modelo lógico e conceitual do banco de dados. Ajuda a entender como o banco de dados deve ser montado, fazendo com que se torne algo prático. O Script conta com a criação das tabelas e se um insert into para cada uma delas.

```
create database camarize_bd;

use camarize_bd;

# Tabela para apenas os registros dos tipos de camarão que são diversos.

create table Tipos_camarao(
    id_tipo_camarao int auto_increment primary key,
    nome varchar(100)
);

# Dieta ideal.

create table Dietas(
    id_dieta int auto_increment primary key,
    descricao varchar(100)
);

# Condições ideais de temperatura, pH e amônia
# para cada tipo específico de camarão.

create table Condicoes_ideais(
    id_condicao int auto_increment primary key,
    id_tipo_camarao int,
    temp_ideal double,
    ph_ideal double,
    amonia_ideal double,
    constraint fk_camarao_condicoes foreign key(id_tipo_camarao)
        references Tipos_camarao(id_tipo_camarao)
);

# Registro do ambiente cativeiro, relacionando com o tipo de camarão que abriga,
# e média diária dos parâmetros.

create table Cativeiros(
    id_cativeiro int auto_increment primary key,
    id_tipo_camarao int,
    data_instalacao date,
    foto_cativeiro varchar(100),
    temp_media_diaria double,
```

```

ph_medio_diario double,
amonia_media_diaria double,
constraint fk_camarao_cativeiro foreign key(id_tipo_camarao)
    references Tipos_camarao(id_tipo_camarao)
);

# Parâmetros atuais de cada cativeiro.

create table Parametros_atuais(
    id_parametros_atuais int auto_increment primary key,
    datahora datetime,
    temp_atual double,
    ph_atual double,
    amonia_atual double,
    id_cativeiro int,
    constraint fk_cativeiro_parametro foreign key(id_cativeiro)
        references Cativeiros(id_cativeiro)
);

# Especificação para cada tipo de camarão
# relacionando as condições ambientais ideais e dieta com o tipo de camarão.

create table Especif_camarao(
    id_especif int auto_increment primary key,
    id_dieta int,
    id_condicao int,
    constraint fk_dieta_especif foreign key(id_dieta)
        references Dietas(id_dieta),
    constraint fk_condicao_especif foreign key(id_condicao)
        references Condicoes_ideais(id_condicao)
);

# Registro de alimentações dadas pelo dispensador de ração
# de acordo com a dieta do cativeiro.

create table Alimentacao(
    id_alimentacao int auto_increment primary key,
    datahora_alimentacao datetime,
    desc_alimentacao varchar(100),
    id_dieta int,
    id_cativeiro int,
    constraint fk_dieta_alimentacao foreign key(id_dieta)
        references Dietas(id_dieta),
    constraint fk_cativeiro_alimentacao foreign key(id_cativeiro)
);

```

```

        references Cativeiros(id_cativeiro)
);

# Registro dos sitios de camarão.

create table Sitios(
    id_sitio int auto_increment primary key,
    nome varchar(100),
    localizacao varchar(255)
);

# Relacionando sitios e cativeiros.

create table SitiosxCativeiros(
    id_sitio_cativieiro int auto_increment primary key,
    id_sitio int,
    id_cativeiro int,
    constraint fk_sitio_sitioxcat foreign key(id_sitio)
        references Sitios(id_sitio),
    constraint fk_cativeiro_sitioxcat foreign key(id_cativeiro)
        references Cativeiros(id_cativeiro)
);
;

# Conta de usuário.

create table Usuarios(
    id_usuario int auto_increment primary key,
    nome_usuario varchar(100),
    email_usuario varchar(50),
    telefone varchar(11),
    senha varchar(50),
    foto_perfil varchar(100)
);
;

# Relacionando cada usuário com seu sitio.

create table UsuariosxSitios(
    id_usuariosx sitio int auto_increment primary key,
    id_usuario int,
    id_sitio int,
    constraint fk_usuario_usuxsitio foreign key(id_usuario)
        references Usuarios(id_usuario),
    constraint fk_sitio_usuxsitio foreign key(id_sitio)
        references Sitios(id_sitio)
);
;
```

```

# Criar registros de relatórios de tanques individuais.

create table Relatorio_individual(
    id_relatorio_individual int auto_increment primary key,
    texto text,
    id_cativeiro int,
    constraint fk_cativeiro_relatorio_i foreign key(id_cativeiro)
        references Cativeiros(id_cativeiro)
);

# Criar registros de relatórios de todos os tanques do sitio.

create table Relatorio_geral(
    id_relatorio_geral int auto_increment primary key,
    texto text,
    id_cativeiro int,
    id_sitio int,
    constraint fk_sitio_relatorio_g foreign key(id_sitio)
        references Sistios(id_sitio)
);

# Tipos de sensor.

create table Tipos_sensor(
    id_tipo_sensor int auto_increment primary key,
    descricao varchar(50)
);

# Registrar sensor.

create table Sensores(
    id_sensor int auto_increment primary key,
    apelido varchar(100),
    id_tipo_sensor int,
    constraint fk_tipo_sensores foreign key(id_tipo_sensor)
        references Tipos_sensor(id_tipo_sensor)
);

# Relacionando quais sensores estão presentes em quais cativeiros.

create table SensoresxCativeiros(
    id_sensor_cativeiro int auto_increment primary key,
    id_sensor int,
    id_cativeiro int,
    constraint fk_sensor_cativeiros foreign key(id_sensor)
        references Sensores(id_sensor),

```

```

constraint fk_cativeiro_sensor foreign key(id_cativeiro)
    references Cativeiros(id_cativeiro)
);

# Recomendações a serem feitas.

create table Recomendacoes(
    id_recomendacao int auto_increment primary key,
    recomendacao text,
    id_cativeiro int,
    id_sitio int,
    constraint fk_cativeiro_rec foreign key(id_cativeiro)
        references Cativeiros(id_cativeiro),
    constraint fk_sitio_rec foreign key(id_sitio)
        references Sitos(id_sitio)
);
;

# Tipos_camaraõ

insert into Tipos_camaraõ (nome) values ('Camarão Branco');
insert into Tipos_camaraõ (nome) values ('Camarão Rosa');
insert into Tipos_camaraõ (nome) values ('Camarão Tigre');

# Dietas

insert into Dietas (descricao) values ('Dieta Proteica');
insert into Dietas (descricao) values ('Dieta Equilibrada');
insert into Dietas (descricao) values ('Dieta Vegetariana');

# Condicoes_ideais

insert into Condicoes_ideais (id_tipo_camaraõ, temp_ideal, ph_ideal, amonia_ideal)
values (1, 27.5, 7.8, 0.1);
insert into Condicoes_ideais (id_tipo_camaraõ, temp_ideal, ph_ideal, amonia_ideal)
values (2, 25.0, 7.5, 0.2);
insert into Condicoes_ideais (id_tipo_camaraõ, temp_ideal, ph_ideal, amonia_ideal)
values (3, 28.0, 7.9, 0.15);

# Cativeiros

insert into Cativeiros (id_tipo_camaraõ, data_instalacao, foto_cativeiro,
    temp_media_diaria, ph_medio_diario, amonia_media_diaria)
values (1, '2024-01-10', 'foto1.jpg', 26.8, 7.7, 0.1);
insert into Cativeiros (id_tipo_camaraõ, data_instalacao, foto_cativeiro,
    temp_media_diaria, ph_medio_diario, amonia_media_diaria)
values (2, '2024-02-15', 'foto2.jpg', 25.2, 7.5, 0.2);

```

```

insert into Cativeiros (id_tipo_camarao, data_instalacao, foto_cativeiro,
temp_media_diaria, ph_medio_diario, amonia_media_diaria)
values (3, '2024-03-20', 'foto3.jpg', 28.1, 7.9, 0.15);

# Parametros_atuais

insert into Parametros_atuais (datahora, temp_atual, ph_atual,
amonia_atual, id_cativeiro)
values ('2024-11-15 08:00:00', 27.0, 7.8, 0.1, 1);
insert into Parametros_atuais (datahora, temp_atual, ph_atual,
amonia_atual, id_cativeiro)
values ('2024-11-15 09:00:00', 25.1, 7.5, 0.2, 2);
insert into Parametros_atuais (datahora, temp_atual, ph_atual,
amonia_atual, id_cativeiro)
values ('2024-11-15 10:00:00', 28.0, 7.9, 0.15, 3);

# Especif_camarao

insert into Especif_camarao (id_dieta, id_condicao) values (1, 1);
insert into Especif_camarao (id_dieta, id_condicao) values (2, 2);
insert into Especif_camarao (id_dieta, id_condicao) values (3, 3);

# Alimentacao

insert into Alimentacao (datahora_alimentacao, desc_alimentacao,
id_dieta, id_cativeiro)
values ('2024-11-15 08:30:00', 'Ração Proteica', 1, 1);
insert into Alimentacao (datahora_alimentacao, desc_alimentacao,
id_dieta, id_cativeiro)
values ('2024-11-15 09:00:00', 'Ração Equilibrada', 2, 2);
insert into Alimentacao (datahora_alimentacao, desc_alimentacao,
id_dieta, id_cativeiro)
values ('2024-11-15 10:30:00', 'Ração Vegetariana', 3, 3);

# Sitos

insert into Sitos (nome, localizacao) values ('Sítio do Camarão 1', 'Local A');
insert into Sitos (nome, localizacao) values ('Sítio do Camarão 2', 'Local B');
insert into Sitos (nome, localizacao) values ('Sítio do Camarão 3', 'Local C');

# SitosxCativeiros

insert into SitosxCativeiros (id_sitio, id_cativeiro) values (1, 1);
insert into SitosxCativeiros (id_sitio, id_cativeiro) values (2, 2);
insert into SitosxCativeiros (id_sitio, id_cativeiro) values (3, 3);

```

```

# Usuarios

insert into Usuarios (nome_usuario, email_usuario, telefone, senha, foto_perfil)
values ('João Silva', 'joao@example.com', '12345678901', 'senha123', 'joao.jpg');
insert into Usuarios (nome_usuario, email_usuario, telefone, senha, foto_perfil)
values ('Maria Santos', 'maria@example.com', '98765432109', 'senha456', 'maria.jpg');
insert into Usuarios (nome_usuario, email_usuario, telefone, senha, foto_perfil)
values ('Pedro Almeida', 'pedro@example.com', '11223344556', 'senha789', 'pedro.jpg');

# UsuariosxSitios

insert into UsuariosxSitios (id_usuario, id_sitio) values (1, 1);
insert into UsuariosxSitios (id_usuario, id_sitio) values (2, 2);
insert into UsuariosxSitios (id_usuario, id_sitio) values (3, 3);

# Relatorio_individual

insert into Relatorio_individual (texto, id_cativeiro)
values ('Parâmetros dentro do ideal.', 1);
insert into Relatorio_individual (texto, id_cativeiro)
values ('pH levemente elevado.', 2);
insert into Relatorio_individual (texto, id_cativeiro)
values ('Temperatura ideal.', 3);

# Relatorio_geral

insert into Relatorio_geral (texto, id_cativeiro, id_sitio)
values ('Todos os tanques estão estáveis.', 1, 1);
insert into Relatorio_geral (texto, id_cativeiro, id_sitio)
values ('Necessário ajustar pH em alguns tanques.', 2, 2);
insert into Relatorio_geral (texto, id_cativeiro, id_sitio)
values ('Temperaturas em níveis ideais.', 3, 3);

# Tipos_sensor

insert into Tipos_sensor (descricao) values ('Sensor de Temperatura');
insert into Tipos_sensor (descricao) values ('Sensor de pH');
insert into Tipos_sensor (descricao) values ('Sensor de Amônia');

# Sensores

insert into SensoresxCativeiros (id_sensor, id_cativeiro) values (1, 1);
insert into SensoresxCativeiros (id_sensor, id_cativeiro) values (2, 2);
insert into SensoresxCativeiros (id_sensor, id_cativeiro) values (3, 3);

# Recomendacoes

```

```

insert into Recomendacoes (recomendacao, id_cativeiro, id_sitio)
values ('Ajustar temperatura para o ideal.', 1, 1);
insert into Recomendacoes (recomendacao, id_cativeiro, id_sitio)
values ('Reducir amônia para níveis aceitáveis.', 2, 2);
insert into Recomendacoes (recomendacao, id_cativeiro, id_sitio)
values ('Manter os parâmetros estáveis.', 3, 3);

```

ANÁLISE DE RECORRÊNCIA DO ALGORITMO MERGE SORT

O Merge Sort é um algoritmo de ordenação que consiste em dividir uma estrutura de subconjuntos e aplicar a ordenação nos elementos que são extraídos da estrutura primária. Após a ordenação dos subconjuntos, é realizada a mistura para um conjunto final ordenado.

O melhor tempo ocorre quando todos os elementos do array já estão ordenados, o Merge Sort sempre executa a divisão e a fusão, independente da ordem dos elementos. O tempo de execução, representado por $O(n \log_2 n)$, pode ser explicado da seguinte forma: $\log_2 n$ corresponde ao número de divisões do array até que cada subarray contenha apenas um elemento, e n refere-se ao número de comparações e fusões realizadas durante o processo de merge.

O tempo médio ocorre quando os elementos estão em ordens aleatórias, em seu comportamento o número de divisões e fusões não dependem da ordem dos elementos, cada nível realiza n operações no total e possui $\log_2 n$ níveis. Sua representação é semelhante com o melhor tempo: $O(n \log_2 n)$.

O pior tempo ocorre quando o array está em ordem oposta a esperada, sua complexidade não é alterada por conta que sua quantidade de execuções não são alteradas. Sua representação se mantém a mesma, sendo ela: $\log_2 n$.

Na aplicação, foi utilizada na tela de sensores, sendo utilizado para organizar os sensores de forma crescente e decrescente. Seu tempo de execução foram exatos 0.000000 milisegundos, mostrando a eficiência e rapidez do algoritmo. Sua representação matemática se apresenta desta maneira:

$$\text{MergeSort}(n) = \begin{cases} 0, & \text{se } n = 0 \\ 0, & \text{se } n = 1 \\ 2 \cdot \text{MergeSort}\left(\frac{n}{2}\right) + O(n), & \text{se } n > 1 \end{cases}$$

A representação do código:

```

function mergeSort(arr) {
    if (arr.length <= 1) return arr;

    const mid = Math.floor(arr.length / 2);
    const left = mergeSort(arr.slice(0, mid));
    const right = mergeSort(arr.slice(mid));

    return merge(left, right);
}

```

```

function merge(left, right) {
    let result = [], leftIndex = 0, rightIndex = 0;

    while (leftIndex < left.length && rightIndex < right.length) {
        if (left[leftIndex].id_tipo_sensor > right[rightIndex].id_tipo_sensor) {
            result.push(left[leftIndex]);
            leftIndex++;
        } else {
            result.push(right[rightIndex]);
            rightIndex++;
        }
    }

    return result.concat(left.slice(leftIndex), right.slice(rightIndex));
}

```

A representação em pseudocódigo:

```

funcão mergeSort(arr[])
    // Função recursiva para ordenar o vetor usando o algoritmo MergeSort

    se tamanho de arr <= 1 então
        retornar arr
    fimse

    meio <- arredondar_para_baixo(arr.length / 2)
    esquerda <- mergeSort(arr[0, meio])
    direita <- mergeSort(arr[meio, arr.length])

    retornar merge(esquerda, direita)
fimfunção

funcão merge(esquerda[], direita[])
    // Função para combinar dois vetores ordenados

    resultado <- lista_vazia
    indice_esquerda <- 0
    indice_direita <- 0

    enquanto indice_esquerda < tamanho de esquerda e
          indice_direita < tamanho de direita faça
        se
            esquerda[indice_esquerda].id_tipo_sensor > direita[indice_direita].id_tipo_sensor

```

```
então
    adicionar esquerda[indice_esquerda] em resultado
    indice_esquerda incrementa 1
senão
    adicionar direita[indice_direita] em resultado
    indice_direita incrementa 1
fimse
fimenquanto

retornar resultado concatenado com esquerda[indice_esquerda, arr.length]
e direita[indice_direita, arr.length]
fimfuncao
```

DIÁRIO DE BORDO

Nome da Atividade	Data de início	Data de término	Responsável pela atividade	Descrição da atividade realizada
Início dos trabalhos para o projeto	23/08/2024	23/08/2024	João Kusaka; Matheus Abrahão; Tiago Rodrigues; Victor Roder; Isabele Queiroz;	Início dos planejamentos e realizações para o projeto no 2º Semestre
Participação na Feira da SEBRAE	31/08/2024	31/08/2024	João Kusaka; Tiago Rodrigues; Victor Roder; Isabele Queiroz.	Nosso projeto foi selecionado para se apresentar na feira da SEBRAE, realizada no dia 31/08/2024 no RBBC, onde apresentamos para turistas a respeito de nosso projeto
Reunião sobre o P.I	03/09/2024	03/09/2024	João Kusaka; Matheus Abrahão; Tiago Rodrigues; Victor Roder; Isabele Queiroz	Reunião convocada pelos membros para acertar determinados pontos a respeito do projeto. Uma decisão importante foi a troca no nome do projeto a sugestão da professora Thissiany.
Modificações no artigo	01/09/2024	17/11/2024	Tiago Rodrigues.	Com base na correção do artigo feito pelos professores Luis e pela professora Thissiany, foram feitas as alterações em todas as partes do artigos, incluindo novas informações, imagens e atualizações de informações. A plataforma para ser feito o artigo neste semestre foi o VsCode.

Modificações no figma	02/09/2024	15/11/2024	Matheus Ferrari.	Modificações feitas em todas as telas, alterando para deixar mais moderno e adaptável para ser montado para a aplicação web.
Modificações no banco de dados	13/09/2024	03/11/2024	Victor Roder; Isabele Queiroz.	Realizada uma mudança completa no banco de dados com o objetivo de melhorar e deixar mais completo para melhor funcionalidade e entendimento. Os modelos do banco foram alterados de forma completa por meio do BrModelo.
Reunião do P.I	04/10/2024	04/10/2024	Isabele Queiroz; João Kusaka; Matheus Abrahão; Tiago Rodrigues; Victor Roder.	Reunião convocada pelos membros para acertar pontos que estavam em abertos e tomar decisões para o rumo do projeto.
Entrega da Introdução e Objetivo	08/10/2024	08/10/2024	Tiago Rodrigues.	Entrega da Introdução e Objetivo do artigo por solicitação do professor orientador para analisar e apontar melhorias.

Inscrição para o GrandPrix do SENAC	10/10/2024	10/10/2024	João Kusaka; Matheus Abrahão; Tiago Rodrigues; Victor Roder; Isabele Queiroz.	Foi decidido pela maioria dos membros em participar do GrandPrix do Senac utilizando o tema do nosso P.I.
Realização da Aplicação Web	22/10/2024	17/11/2024	João Kusaka; Isabele Queiroz	Foi realizado a aplicação web, feita em Node.js no VsCode. A aplicação está sendo feita com base no Figma e no banco de dados com o objetivo de estar semelhante com o que foi planejado. Toda e qualquer atualização realizada é registrada no GitHub
Realização dos Diagramas de Classe e de Objetos	25/10/2024	03/11/2024	Victor Roder.	A montagem dos diagramas de classe e de objetos se deu por meio do draw.io. Ambos os modelos foram feitos com base no banco de dados com a intenção de manter a fidelidade com o que foi planejado, juntamente com a explicação de cada diagrama.
Modificações no Canvas	22/10/2024	03/11/2024	Isabele Queiroz	Realizada alterações no canvas após revisão e solicitação do integrante Tiago.
Entrega dos requisitos para o GrandPrix	12/11/2024	12/11/2024	João Kusaka; Matheus Ferrari; Tiago Rodrigues; Victor Roder; Isabele Queiroz.	Realizada a entrega dos requisitos solicitado para o GrandPrix do SENAC, sendo eles um pitch e um figma.

Entrega do Estado da Arte, Metodologia e Resultados	03/11/2024	03/11/2024	Tiago Rodrigues.	Entrega do Estado da Arte, Metodologia e Resultados a pedido do professor orientador para melhorias e ajustes.
Modificações no Banner	13/11/2024	17/11/2024	Tiago Rodrigues.	Foram realizadas mudanças no banner para estarem com as informações atualizadas acerca do projeto e dos novos protótipos de tela. Banner atualizado pelo Figma.
Modificações no Pitch	13/11/2024	17/11/2024	Isabele Queiroz.	Foram realizadas mudanças no pitch com objetivo de deixar o pitch atualizado.
Realização do modelo físico	13/11/2024	17/11/2024	Tiago Rodrigues.	O modelo físico do banco de dados foi realizado por meio da plataforma dbdiagram.io, foi moldado com base no modelo conceitual.