MINISTÉRIO DA EDUCAÇÃO
SECRETARIA DE EDUCAÇÃO PROFISSIONAL E TECNOLÓGICA
INSTITUTO FEDERAL DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA
FARROUPILHA *CAMPUS* PANAMBI

SISTEMA DE GERENCIAMENTO DA TERMOMETRIA E AUTOMAÇÃO DA AERAÇÃO PARA UNIDADES ARMAZENADORAS DE GRÃOS

TRABALHO DE CONCLUSÃO DE CURSO

BRUNO PAUTZ DE OLIVEIRA

Panambi, RS, Brasil
2021

SISTEMA DE GERENCIAMENTO DA TERMOMETRIA E AUTOMAÇÃO DA AERAÇÃO PARA UNIDADES ARMAZENADORAS DE GRÃOS

por

Bruno Pautz de Oliveira

Trabalho de conclusão de curso de graduação apresentado ao Instituto Federal Farroupilha Campus Panambi como requisito parcial para a obtenção do título de Tecnólogo em Sistemas para Internet.

Orientador: Prof. Me. Cléber Rubert

Panambi, RS, Brasil

2021

MINISTÉRIO DA EDUCAÇÃO
SECRETARIA DE EDUCAÇÃO PROFISSIONAL E TECNOLÓGICA
INSTITUTO FEDERAL DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA
FARROUPILHA *CAMPUS* PANAMBI

A Comissão Examinadora, abaixo assinada, aprova a Monografia

SISTEMA DE GERENCIAMENTO DA TERMOMETRIA E AUTOMAÇÃO DA AERAÇÃO PARA UNIDADES ARMAZENADORAS DE GRÃOS

elaborada por

Bruno Pautz de Oliveira

como requisito parcial para obtenção do título de
Tecnólogo em Sistemas para Internet

COMISÃO EXAMINADORA

_____

Cléber Rubert
(Orientador)

_____

Rosana Wagner, Dr. (IFFar)

_____

Rodrigo Luiz Antoniazzi, Me. (IFFar)

Conceito Final: _____

Panambi, ... de ..... de 2021.

RESUMO

O processo de modernização da agricultura no Brasil ao longo das últimas décadas tem levado ao aumento da produção de alimentos, principalmente o milho, o trigo, a soja e a cevada. Além de produzir, é preciso armazenar os produtos com eficiência, a fim de preservar para que possa garantir o menor desperdício possível dos mesmos e maiores lucros com suas vendas. Caso a umidade ou a temperatura se mantenham alta por um longo período de tempo, serão formadas condições favoráveis ao aparecimento de fungos, microflora, insetos, germinação, levando a perda de qualidade e consequentemente a redução do valor do produto. O armazenamento adequado da produção é uma questão de segurança, desta forma, quando conduzido adequadamente, a qualidade do produto é preservada e as perdas são evitadas ou minimizadas. Este trabalho consiste no desenvolvimento de um sistema para controle e gerenciamento de unidades armazenadoras de grãos, onde através de uma interface web, banco de dados, os usuários poderão acompanhar por meio de relatórios a situação da sua armazenagem e tomar decisões. Após a implementação do sistema desenvolvido, testes foram realizados, conceitos foram revistos e mudanças foram implantadas, tornando o sistema robusto, confiável e de baixo custo, completando as exigências do mercado atual.

Palavras-chave: armazenagem; sistema; qualidade.

ABSTRACT

The modernization process of agriculture in Brazil over the last decades has led to an increase in food production, mainly corn, wheat, soy and barley. In addition to producing, it is necessary to store the products efficiently, in order to preserve them so that you can guarantee the least possible waste of them and greater profits from their sales. If the humidity or temperature remains high for a long period of time, favorable conditions will be created for the appearance of fungi, microflora, insects, germination, leading to a loss of quality and, consequently, a reduction in the value of the product. Proper storage of production is a matter of safety, thus, when carried out properly, product quality is preserved and losses are avoided or minimized. This work consists in the development of a system for control and management of grain storage units, where, through a web interface, database, users will be able to monitor the status of their storage and make decisions with reports. After the implementation of the developed system, tests were carried out, concepts were revised and changes were implemented, making the system robust, reliable and low-cost, completing the requirements of the current market.

Keywords: storage; system; quality.

LISTA DE FIGURAS

SUMÁRIO

1 INTRODUÇÃO

O processo de modernização da agricultura no Brasil ao longo das últimas décadas tem levado ao aumento da produção de alimentos, principalmente o milho, o trigo, a soja e a cevada. Conforme dados divulgados pela Empresa Brasileira de Pesquisa Agropecuária (EMBRAPA, 2017), o Brasil é o segundo maior produtor mundial de soja, ficando atrás apenas dos Estados Unidos.

Aliado ao acréscimo na produção, existe uma grande demanda por locais adequados para armazenar esses produtos por um período de tempo e conservar as propriedades dos grãos colhidos, uma vez que além de produzir, é preciso armazenar os produtos com eficiência, a fim de preservar para que se possa garantir o menor desperdício possível dos mesmos e maiores lucros com suas vendas.

A aeração realizada sem controle pode reduzir a umidade do produto a níveis muito abaixo do valor de comercialização reduzindo assim a lucratividade por perda de peso. Caso a umidade ou a temperatura se mantenham alta por um longo período de tempo, serão formadas condições favoráveis ao aparecimento de fungos, microflora, insetos, germinação, levando a perda de qualidade e consequentemente a redução do valor do produto. (Weber, 2005).

Nesse sentido o presente trabalho tem o objetivo de apresentar o desenvolvimento de um sistema web, onde o qual se conecta aos painéis termocenter fabricados pela empresa Winckieel, coletando as informações e armazenando em um banco de dados possibilitando o histórico, análise e monitoramento da termometria, aeração e estação meteorológica, com isso, auxiliando os usuários na tomada de decisões.

1.1 OBJETIVOS

1.1.1 Objetivo geral

O objetivo geral deste trabalho é desenvolver um sistema web para o monitoramento da termometria e controle da aeração nos grãos armazenados em silos e armazéns.

1.1.2 Objetivos específicos

- Desenvolver algoritmos para comunicar e coletar informações dos painéis termocenter;

- Implementar um banco de dados para o armazenamento das informações;
- Desenvolver uma nova interface web responsiva;
- Implementar recursos no sistema de envio de relatórios de termometria;
- Implementar telas de analise gráficas dos dados coletados;
- Implementar telas de análise de custo de aeração;
- Implementar recursos de automatização da aeração via interface;
- Desenvolver algoritmos para análise dos grãos com objetivo de auxiliar os usuários na tomada de decisões.

1.2 JUSTIFICATIVA

O objetivo da armazenagem de grãos é manter por um período de tempo, a qualidade e características dos grãos, dessa forma, os sistemas de termometria possibilitam o monitoramento das temperaturas dentro dos silos, relatórios de temperaturas que possibilita na tomada de decisões e verificação de risco de pragas. Alinhado a isso, automação da aeração possibilita parametrizar programas de aeração para serem ligados no momento certo com segurança. Dessa forma, ajuda no controle e qualidade dos grãos armazenadoras. Também através da necessidade de redução de custos na empresa Winckieel de Panambi, observou-se o alto custo da aquisição de computadores compactos e suas licenças de *Windows* utilizadas para a coleta das informações nos painéis termocenter.

Além disso, o sistema desenvolvido para sistema operacional *Windows* em 2013 e atualizado em 2016, apresenta uma interface não responsiva o que dificulta o acesso dos clientes que optaram por acessar as informações via *smartphone*.

1.3 ESTRUTURA DO TRABALHO

Este texto está organizado em capítulos. O capítulo 2 é o referencial teórico, no qual explica os fundamentos do sistema desenvolvido para unidades armazenadoras de grãos. O capítulo 3 apresenta o material e o método utilizado para desenvolver a aplicação. O capítulo 4 é apresentado o sistema web desenvolvido, com as telas que representam as funcionalidades implementadas e a forma de interação do usuário com o sistema. Em seguida é apresentado a conclusão do trabalho e na sequência o referencial teórico utilizado no texto.

2 REFERENCIAL TEÓRICO

Neste capítulo apresenta o referencial teórico abordando temas essenciais para compreensão do sistema, destacam-se temáticas sobre o armazenamento de grãos, painéis termocenter e programação, abordando linguagens e *framework* no desenvolvimento. Além do sistema de banco de dados usado para o armazenamento de todas as informações.

2.1 ARMAZENAMENTO DE GRÃOS

Segundo Azevedo et al. (2008, p. 2) "[...] o Brasil é um dos países que mais se destaca no cenário mundial da agricultura, devido à sua crescente expansão na produção de grãos".

Para que esses produtos possam ser utilizados no futuro, é necessário um processo de armazenamento, ou seja, uma determinada quantidade de grãos é coletada e armazenada em um determinado período de tempo para manter suas características normais e evitar fungos, pragas, umidade excessiva e entre outras (SILVA et al., 2012).

Existem dois tipos tradicionais de armazenamento: a granel ou em volumes. No armazenamento a granel, os grãos em silos / armazéns (metal, concreto, etc.) ficam soltos. Em volumes, o produto é colocado em sacos e empilhado, geralmente em galpões (AZEVEDO et al., 2008).

No entendimento de Rezende (2009, p. 23) "produzir e conservar com excelência são fatores que determinaram o incremento de receita à cadeia produtiva, maior diferenciação no competitivo mercado, além da disponibilidade de produtos seguros para o consumidor".

Segundo dados da FAO (Organização das Nações Unidas para Agricultura e Alimentação), e do MAPA (Ministério da Agricultura, Pecuária e abastecimento Brasileiro), calcula-se que as perdas de grãos provocadas pelo ataque de pragas no Brasil somam aproximadamente 10% do que é produzido anualmente (LORINI, 2005).

Para Cloud e Morey (1979), a aeração é a transferência de ar através da massa de grãos armazenados, reduzindo assim a taxa de deterioração dos grãos e assim, reduzir as perdas do armazenamento. Os danos causados pela atividade de insetos e de fungos estão relacionados ao índice de umidade e à temperatura dos grãos armazenados. A aeração melhora significativamente as condições de armazenamento de grãos, mantendo um gradiente de temperatura pequeno e uniforme durante todo o período de armazenamento, reduzindo assim o crescimento de fungos, a atividade do inseto e a migração de umidade.

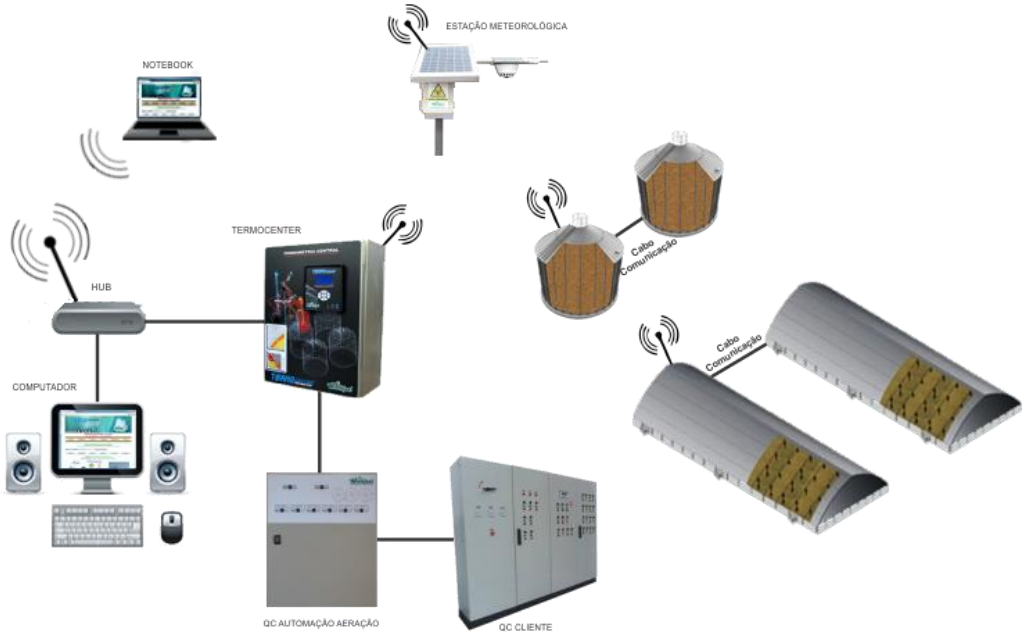
A armazenagem passa por muitas mudanças que se refletem na adição de novos sistemas de informação aplicados ao controle de armazenagem, em sistemas automatizados de controle de ventilação e de temperatura com a principal finalidade de estocar produtos (FLEURY 2000).

As mudanças foram predominantes, segundo Figueiredo (2004), por fatores de resposta rápida, leituras em tempo real, melhor controle de qualidade, redução de desperdício, garantindo um melhor acompanhamento durante o período de estocagem.

2.2 PAINÉIS TERMOCENTER

Os painéis termocenter são painéis centrais de termometria que realizam o chaveamento dos sensores colocados nos silos e armazéns, também realiza o acionamento e desligamento dos motores da aeração através do monitoramento dos dados coletados da estação meteorológica. Além do processamento, o painel tem a capacidade de armazenar em sua memória algumas informações coletadas, permitindo o armazenamento de pequena quantidade de informações que são subscritas com novas informações. Abaixo a Figura 1 com o fluxograma do painel termocenter.

Figura 1- Arquitetura de comunicação dos Painéis Termocenter

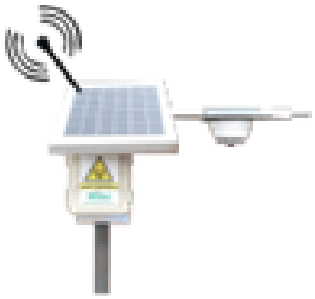


Fonte: Winckieel

2.2.1 Estação Meteorológica

A estação meteorológica contém sensores de temperatura, umidade relativa do ar, detecção de chuva e opcional sensor de volume de chuva (pluviômetro). Possui bateria recarregável e painel solar para o seu funcionamento. É responsável por informar as condições climáticas ao sistema de controle termocenter através de uma rede de comunicação Wi-Fi.

Figura 2 - Estação meteorológica

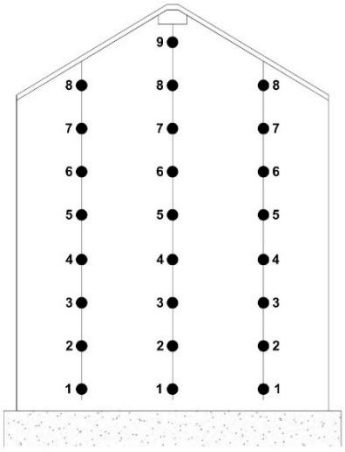


Fonte: Winckieel

2.2.2 Cabos pêndulos

Os cabos pêndulos contêm os sensores de temperatura, os quais serão interligados através de cabos de compensação até as caixas de comutação, que por sua vez realizam a leitura dos sensores.

Figura 3 - Disposição dos sensores no interior do cabo pêndulo



Fonte: Winckieel

Figura 4 - Cabo pêndulo instalado dentro do silo

Fonte: Winckieel

2.2.3 Comunicação entre silos/armazéns e painel termocenter

As caixas de multiplexação são responsáveis pela leitura e chaveamento dos sensores de temperatura, posteriormente enviam estas informações ao sistema de controle termocenter através de uma rede de comunicação Wi-Fi.

Figura 5 – Caixa de multiplexação

Fonte: Winckieel

2.2.4 QC Automação aeração

O QC de automação da aeração é responsável pela automação do sistema de aeração dos silos/armazéns. Possuem saídas para acionamento dos motores e entradas para a confirmação de motor ligado. São interligados ao QC do cliente através de cabos de controle. A comunicação com o painel termocenter é através de uma rede de comunicação no padrão RS-485.

Figura 6 - QC de automação

Fonte: Winckieel

2.2.5 QC Cliente

Um quadro de comando é uma caixa ou estrutura que distribui todos os disjuntores, interruptores, temporizadores, relés, CLPs e equipamentos usados para controlar o sistema elétrico.

Figura 7- QC acionamento dos motores da aeração (comando e força)

Fonte: Winckieel

2.3 COMUNICAÇÃO SOCKET

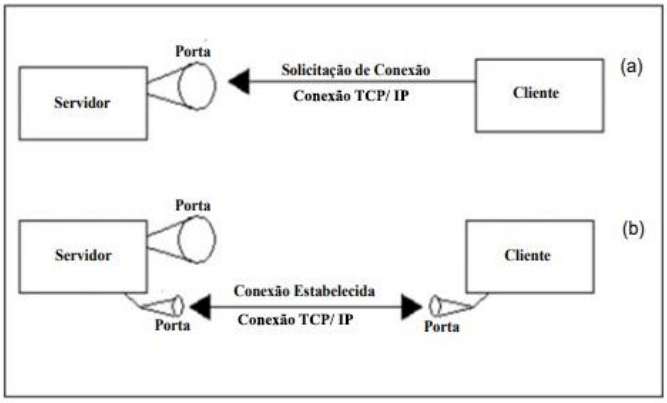
Para Correira (2008), o mecanismo de comunicação por socket é o mais comumente utilizado nas aplicações e permite que seja utilizado de forma orientada a conexão que trabalha com o uso do protocolo TCP / IP para fornecer serviços confiáveis sem perda de dados na web, a sequência de pacotes de dados também permite o uso de DataStreams.

Como mostra Stallings (2003), cada socket tem um número que consiste no IP do host mais um número de 16 bits local para este host, denominado porta. Para que a comunicação funcione, uma conexão deve ser estabelecida entre o socket do remetente e o socket do receptor.

Para conseguir fazer uma conexão cliente-servidor, o servidor terá que conter um socket com uma porta dedicada para receber conexões, cada conexão tem um mecanismo que permite ao receber várias conexões, criando vários canais ou outras conexões Individuais para que cada cliente faça as suas trocas de dados.

Normalmente um servidor primeiro “escuta” e “aceita” uma conexão e depois cria um novo processo para se comunicar com o cliente. Nesse meio tempo ele continuaria” escutando” pedidos de conexão no processo original. (COULOURIS, GEORGE; DOLLIMORE JEAN; KINDBERG TIM. 2007 p. 159). A Figura 8 mostra como a conexão entre o cliente e o servidor é estabelecida de forma muito simples.

Figura 8 - Comunicação via socket UDP (a) e TCP (b)



Fonte: Elaborado pelo autor (Adaptado de (Hopson et al., 1997))

Para a conexão ser estabelecida são realizados alguns passos do lado cliente e servidor.

Cliente:

Passo 1: Cria à conexão de *socket* cliente;

Passo 2: Adquirem fluxos de leitura e escrita para o *socket*;

Passo 3: Utilizam os fluxos de acordo com o protocolo do servidor;

Passo 4: Encerra os fluxos de comunicação;

Passo 5: Encerra o *socket.*

Servidor:

Passo 1: Cria a conexão de *socket* servidor e iniciam a escuta;

Passo 2: Chama o método *accept* para obter novas conexões;

Passo 3: Cria os fluxos de entrada e saída para o *socket* retornado;

Passo 4: Conduz à comunicação com base no protocolo determinado;

Passo 5: Encerra os fluxos e o *socket* do cliente;

Passo 6: Volta ao passo dois ou continuam até o passo sete;

Passo 7: Encerra o *socket* servidor.

2.4 PYTHON

Python é uma linguagem de programação de alto nível - ou *High Level Language*, inclui diversas estruturas (listas, tuplas, dicionários, data / hora, complexos e outras) e um grande número de módulos prontos e *frameworks* de terceiros que podem ser adicionados. Ele também

possui recursos em outras linguagens modernas, como geradores, introspecção, persistência, metaclasses e unidades de teste.

Python foi criado por Guido Van Rossum quinze anos atrás com a ajuda de dois colegas Jack Jansen e Sjoerd Mullender como *hobby*. O objetivo dos criadores de Python era criar uma linguagem orientada a objetos, altamente portável e menos complexa do que Java ou C++ (SONGINI, 2005). A filosofia de design da linguagem é enfatizar a importância do esforço do programador sobre o esforço computacional. Priorize a legibilidade do código em relação à velocidade ou expressividade. Atualmente, possui um modelo de desenvolvimento comunitário, que é aberto e gerenciado pela organização sem fins lucrativos Python Software Foundation.

Segundo a Redação Impacta (2020), após anos de ajustes, a linguagem Python começou a se tornar a primeira escolha da comunidade de desenvolvedores e se tornou uma das linguagens mais populares do mundo hoje. KAY (2005) enfatizou que "Python é uma linguagem de programação de código aberto orientada a objetos, frequentemente usada para desenvolvimento rápido de aplicativos. Sintaxe simples, ênfase na legibilidade, redução do custo de manutenção de programas e sua enorme biblioteca de funções incentiva a reutilização e escalabilidade Multiparadigma, além de orientada a objetos, a linguagem também suporta programação modular e funcional. Mesmo os tipos básicos no Python são objetos. A linguagem é interpretada através de *bytecode* pela máquina virtual Python, tornando o código portável. Isso torna possível compilar aplicativos em uma plataforma e executá-los em outras plataformas ou diretamente do código-fonte. Python é um *software* de código aberto (licenciado pela General Public License (GPL), mas com menos restrições, permitindo que Python seja incorporado a produtos proprietários). Python é amplamente utilizado como linguagem de *script* em vários *softwares*, permitindo automatizar tarefas e adicionar novos recursos, incluindo: BrOffice.org, PostgreSQL, Blender e GIMP. Você também pode integrar Python com outras linguagens (como a linguagem C). Em termos gerais, o Python apresenta muitas similaridades com outras linguagens dinâmicas, como Perl e Ruby (Borges, 2009).

2.5 LARAVEL

O Laravel é um *framework* PHP utilizado para desenvolvimento web. Possui uma arquitetura MVC (*Model-View-Controller*) e tem, como aspecto principal, o de auxiliar no desenvolvimento de aplicações seguras e de alto desempenho de forma ágil e simplificada, com

código limpo. Outra característica é o incentivo do uso de boas práticas de programação e a utilização de padrões específicos a ele determinados.

Assim como outros *frameworks*, o Laravel possui testes já definidos e a estrutura possui métodos auxiliares convenientes, que permitem testar expressamente suas aplicações. No diretório de testes podem ser encontrados testes de banco de dados, testes de navegador e testes que geram entradas de dados fictícios. Quando executados os testes, devem ser definidas as variáveis e realizadas as chamadas dos dados que serão testados na aplicação (LARAVEL, 2018).

Segundo Verma (2014), a arquitetura MVC utilizada pelo Laravel é um padrão que visa aumentar a modularidade de sistemas de software, sendo apresentada em três camadas:

- *Model*: gerencia os modelos de dados da aplicação, fazendo a interação com o banco de dados, analisando a lógica, os dados e as regras. É uma camada entre os dados e a aplicação, que pode armazenar diversos tipos de dados, de sistemas gerenciadores de bancos de dados, como o MySQL, ou arquivos *Extensible Markup Language* (XML);

- *View*: representa a camada de interação com o usuário por meio de interfaces. Diz respeito à representação da aplicação web e é responsável por mostrar os dados que a camada *controller* recebe da camada model. Pode ser implementado facilmente com o uso do pacote ".blade.php" ou com código PHP. O Laravel inicia sua execução com a extensão de arquivo (.blade.php ou .php) determinando quem deve prosseguir com a execução do modelo;

- *Controller*: recebe as solicitações dos usuários através da *view* para a exibição ou atualização de dados e faz requisições na camada model de acordo com a solicitação correspondente. É considerado um link entre a model e a *view* e dispõe de duas opções de desenvolvimento da lógica: *Router* e *Controller*. Os *Routers* são mais viáveis para páginas web estáticas. *Controllers* são definições de escrita para cada página web.

2.6 BANCO DE DADOS

Um banco de dados é uma coleção organizada de informações ou dados estruturados, normalmente armazenados eletronicamente em um sistema de computador. Um banco de dados é geralmente controlado por um sistema de gerenciamento de banco de dados. Juntos, os dados e o sistema de gerenciamento, juntamente com os aplicativos associados a eles, são chamados de sistema de banco de dados, geralmente abreviados para apenas banco de dados.

Os dados nos tipos mais comuns de bancos de dados em operação atualmente são modelados em linhas e colunas em uma série de tabelas para tornar o processamento e a consulta

de dados eficientes. Os dados podem ser facilmente acessados, gerenciados, modificados, atualizados, controlados e organizados. A maioria dos bancos de dados usa a linguagem de consulta estruturada para escrever e consultar dados (ORACLE, 2020).

2.6.1 MariaDB

MariaDB foi o nome dado ao sistema gerenciador de banco de dados (SGBD) pelos antigos desenvolvedores do MySQL, que foi comprado pela Oracle. Motivados pela vontade de manter o projeto com código aberto e gratuito, como era o MySQL antes de sua venda, o MariaDB foi criado. Sua sintaxe continuou igual à de seu antecessor, isso faz com que os sistemas tenham retro compatibilidade (BARTHOLOMEW, 2019).

Um sistema gerenciador de banco de dados é uma aplicação que fica responsável por manipular e manter dados. A vantagem de utilizar um SGBD é que não é necessário se preocupar em manipular dados diretamente em arquivos, o sistema fica isento dessa responsabilidade. Esses sistemas também fornecem uma *Command Line Interface* (CLI), em português Interface de Linha de Comando, onde é possível manipular os dados através de comandos. A linguagem utilizada pelo MariaDB para a manipulação dos dados é a SQL, que tem uma sintaxe bem simples e intuitiva (LAUDON, LAUDON, 2010).

2.7 SERVIDORES LINUX

Servidor pode ser considerado como um dispositivo que oferece recursos para a rede, ou um dispositivo compartilhado por muitos usuários. Os serviços oferecidos por um servidor são vários, como serviço de impressão, servidor web, servidor de arquivos, etc. Um único servidor pode executar vários serviços ao mesmo tempo, porém, para que isso seja possível, deve ser levado em consideração a quantidade de trabalho e a disponibilidade de *hardware* (MORIMOTO, 2008).

Os servidores Linux podem ser classificados: (i) conforme os serviços que possui configurado; (ii) de acordo com os serviços que rodam compartilhamento de conexão, impressoras, autenticação de usuários, compartilhamento de arquivos e; (iii) os que servem como servidor de *firewall*. Os servidores que hospedam aplicações para a grande rede e sites são classificados como servidores de Internet (MORIMOTO, 2008).

Para Ferreira (2008) o Linux é uma alternativa barata e funcional para quem não quer pagar alto por um sistema operacional comercial ou não tem um computador suficientemente rápido.

O Linux não é um *software* de domínio público, mas está licenciado como GNU (*General Public Licence*), e pode permanecer disponível livremente. As pessoas podem até cobrar pela cópia, mas isso não pode limitar a sua distribuição, pois é um código aberto, ou seja, qualquer usuário pode fazer alterações e melhorias no sistema. (FERREIRA 2008).

Segundo Reckers e Silva (2007), o sistema operacional Linux, apresenta a seguinte vantagem:

Vantagens: Trata-se de um sistema livre, com código-fonte aberto, assim sendo facilmente adaptado e personalizado para a empresa de acordo com o seu ramo de atividade e porte; sua distribuição é gratuita não sendo necessário desembolsar recursos para ter o direito de uso; tem uma grande adaptação com equipamento fora de linha ou antigos; possui um funcionamento estável sem perder desempenho; possuem diversos ambientes gráficos; além de ser compatível com todos os sistemas de arquivos.

Selim (2006) relata que as vantagens para migração são muitas. As empresas podem beneficiar-se mais do *software* livre do que usuários *desktop*. As principais vantagens encontradas são:

Não pagar licença: o *software*, bem como suas atualizações, não gera custos;

Segurança: sistemas com código-fonte aberto têm uma tendência a possuir menos brechas de segurança;

Privacidade: o código-fonte aberto possibilita a empresa saber exatamente o que o *software* está fazendo;

Adaptação: os *softwares* licenciados pela GPL podem ser modificados livremente, o que permite às empresas adaptarem o software às suas necessidades;

Controle: o próprio sistema possui, desde que foi criado, nível de controle para os usuários que permite tornar o sistema mais “estável”, ou seguro, diante de erros cometidos por usuários.

3 METODOLOGIA

3.1 CLASSIFICAÇÃO DA PESQUISA

A natureza de estudo se classifica como pesquisa aplicada, pois desenvolveu uma solução para o problema da empresa Winckieel de Panambi. Seu desenvolvimento envolveu conceitos e padrões existentes.

Pode-se considerar a abordagem qualitativa, pois através da experiência e *feedbacks* dos clientes foi possível implementar melhorias que o antigo sistema não suporta.

3.2 UNIVERSO AMOSTRAL

Como a empresa Winckieel presta suporte para o Brasil inteiro, e também para o exterior, principalmente na América Latina em países como Paraguai e Argentina, o sistema poderá ser utilizado no mundo inteiro. O projeto foi desenvolvido com um sistema multi-languagem em português, inglês e espanhol, dessa forma ele foi desenvolvido como uma nova aplicação, no lugar do antigo sistema termocenter server, e é hospedado localmente em um servidor web Apache, com banco de dados MariaDB.

3.3 SUJEITO DA PESQUISA

O sujeito da pesquisa são os armazenadores de grãos, setor da engenharia e comercial da empresa Winckieel, clientes que usam o sistema termocenter server que forneceram explicações sobre o funcionamento do sistema, sugestões de implementação de novos recursos no sistema e melhorias.

3.4 PLANO DE COLETA DE DADOS

O estudo para desenvolvimento do projeto foi baseado na utilização do sistema termocenter server da empresa Winckieel. Setor da engenharia e comercial, como já citado na seção 3.3, também colaboraram para o estudo do projeto.

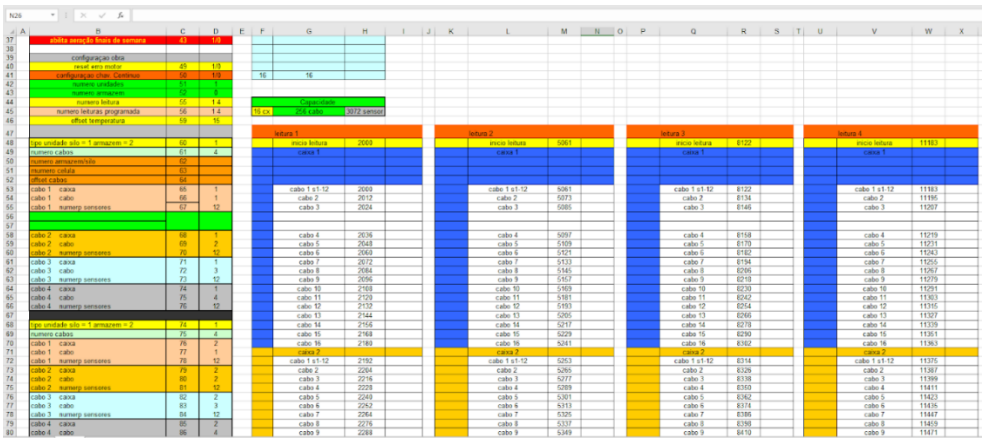
3.5 PLANO DE ANÁLISE E INTERPRETAÇÃO DOS DADOS

A análise foi através de testes unitários na empresa Winckieel, buscando a correção de problemas de código na integridade e confiabilidade dos dados coletados e exibidos no sistema. Além disso, através dos *feedbacks* (termo utilizado em inglês para um parecer sobre alguma coisa) da unidade armazenadora teste.

3.6 PLANO DE SISTEMATIZAÇÃO DE ESTUDO

O presente projeto desenvolvido parte do estudo do antigo sistema termocenter server disponibilizado pela empresa Winckieel e do mapa memorial, extraindo padrões de informações, identificando a estrutura dos dados coletados e as variáveis para a elaboração da infraestrutura do sistema, esse estudo foi ainda mais importante pelo fato do sistema ser configurável, neste caso, ser de porte pequeno ou de porte grande, com muitos silos e motores de aeração.

Figura 9 – Mapa memorial



Fonte: Winckieel

Os resultados do sistema foram baseados em apresentar os mesmos dados do antigo sistema termocenter server. Porém o sistema tem novos recursos e correções de falhas do antigo sistema.

Por se tratar de um sistema web, foi utilizado conceitos aprendidos em aula para desenvolvimento toda sua estrutura geral, de material de estudos do Bootstrap para o desenvolvimento da interface gráfica, teste de algoritmos e integridade de dados do banco.

3.7 RECURSOS

Para desenvolvimento do novo sistema termocenter, foi analisado e utilizado o antigo sistema termocenter server, conforme visto na Figura 10, presente na empresa Winckieel em Panambi. O sistema foi utilizado como base para o novo sistema, onde foi retirado quais recursos mínimos necessários o novo sistema tem, a estrutura de dados e organização dos mesmos.

Figura 10 - Tela sistema termocenter

Fonte: Winckieel

Também foi adquirido um servidor Linux para a instalação do servidor web, banco de dados e serviços que irão coletar os dados do painel termocenter.

Para utilização do sistema, será necessário um *software* de navegadores de internet, de preferência Google Chrome, Chromium, Firefox ou Edge. Seu desenvolvimento se deu na ferramenta Visual Studio Code. A linguagem de programação para web utilizada foi o PHP junto ao *framework* Laravel, seu layout é formato HTML, junto com *framework* Bootstrap de CSS para a estilização das páginas. Os serviços foram desenvolvidos em Python, no qual, fazem a comunicação socket ao painel, coletam os dados e salvam no banco de dados as informações.

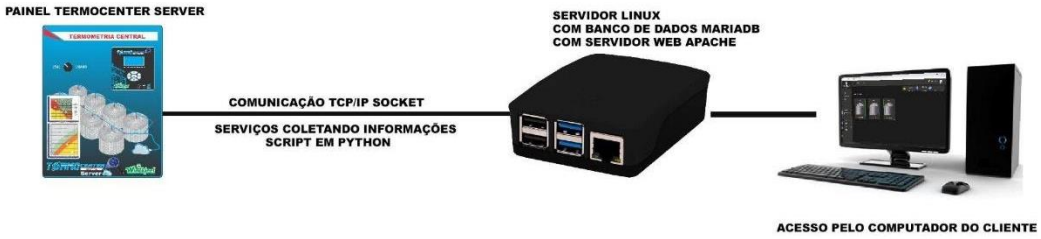
4 RESULTADOS E DISCUSSÕES

Este capítulo apresenta a análise realizada para sistema web, sistema de coletas das informações, alguns diagramas que exemplificam a utilização do sistema proposto, e outros que representam uma visão de classes e objetos.

São apresentados também o procedimento de desenvolvimento dos aplicativos, as telas, funcionalidades e alguns códigos, que representam algumas funcionalidades principais.

Para melhor entendimento, a Figura 11 apresenta um diagrama geral de como o sistema funciona. Os serviços são responsáveis pela comunicação e coleta de dados do painel termocenter, estes dados são coletados via socket, tratados e gravados no banco de dados MariaDB. Por meio do sistema web desenvolvido o usuário tem acesso a essas informações, visualizando e exportando relatórios, sabendo em tempo real qual a situação da sua aeração, podendo através dos recursos do sistema alterar parâmetros de aeração com base nos dados coletados e visualizados na tela.

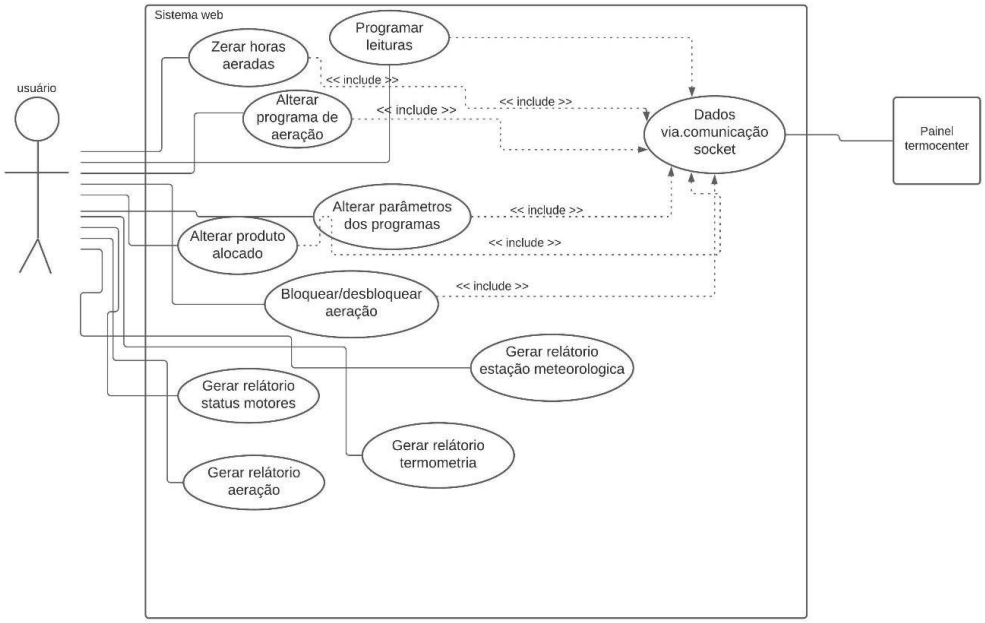
Figura 11 - Arquitetura de comunicação painel - servidor - cliente



Fonte: Elaborado pelo autor

As funcionalidades do sistema web pode ser entendidas com o diagrama de caso de uso, vista na Figura 12, onde o usuário através do sistema é capaz de alterar o programa de aeração do painel, alterar seus status, onde a aeração pode estar bloqueada ou desbloqueada, alterar o produto alocado dentro do silo/armazém e alterar os parâmetros dos programas, como, a temperatura mínima e máxima, umidade mínima e máxima, horário de acionado e término da aeração.

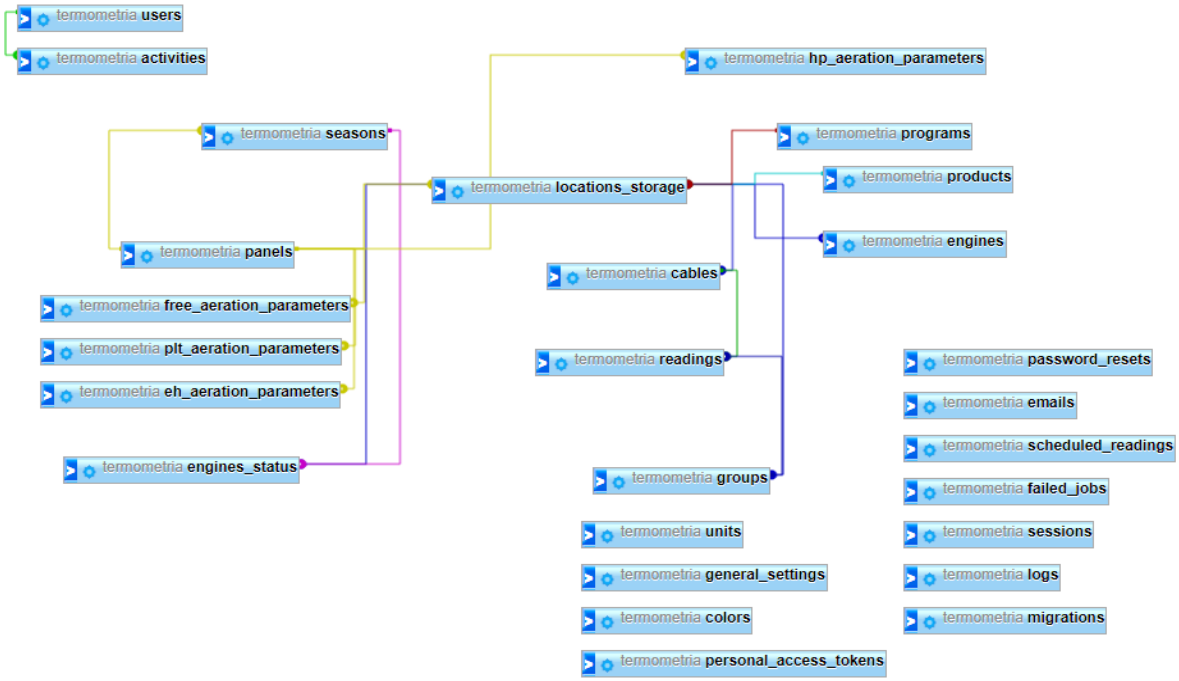
Figura 12 - Diagrama classe de uso



Fonte: Elaborado pelo autor

O banco de dados relacional MariaDB foi configurado através do recurso de migração do framework Laravel, na Figura 13, pode ser visto a sua modelagem, com os relacionamentos e tabelas existentes no banco. A tabela “locations_storage” é a tabela pivô, local onde são gravadas as informações de silos e armazéns. Através dela é possível saber qual programa de aeração aquele silo está executando, total de horas aeradas, nível, capacidade de armazenagem. Sendo uma tabela pivô, há várias relações de tabelas com ela, o que chamamos de *foreign key* (chave estrageiro). Pode ser visto isso na tabela “engines”, tabela criada para salvar as informações de cada motor, através da coluna “place_storage” é feito o relacionamento com a tabela “locations_storage”, o mesmo acontece com a tabela “cables”, responsável por armazenar informações sobre o cabo, como, a posição do cabo (quando este cabo pertence a um armazém), arco, quantidade de sensores.

Figura 13 - Diagrama de Entidade Relacional



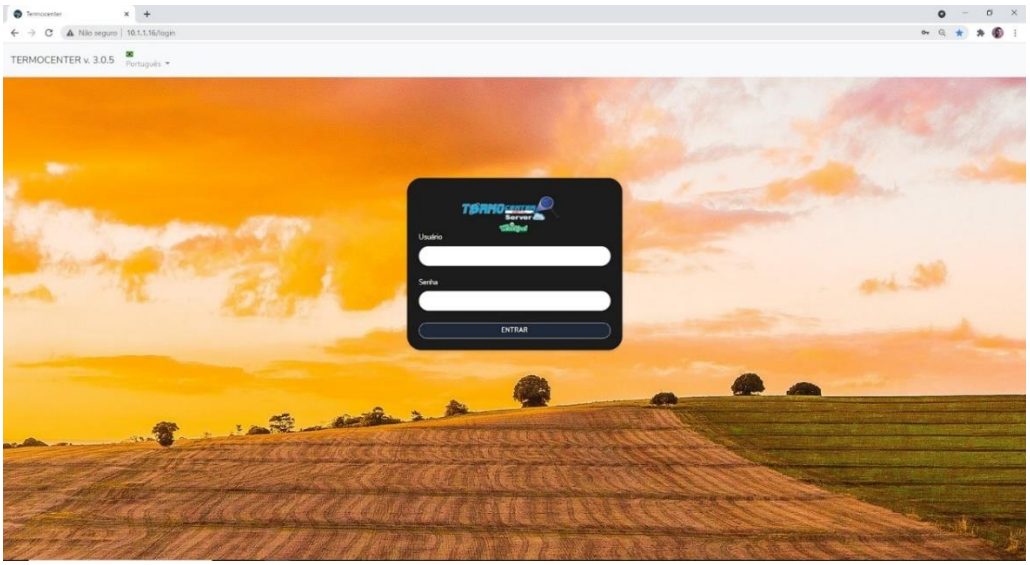
Fonte: Elaborado pelo autor

4.1 DESENVOLVIMENTO DO SISTEMA WEB

Neste tópico serão apresentadas algumas telas do sistema web, produto do sistema objeto principal deste trabalho.

Como primeira tela, como visto na Figura 14, apresenta-se uma interface de autenticação, a tela de login, nela pode-se informar os valores de usuário e senha e pressionar o botão “Entrar” para possibilitar o acesso ao sistema, também na barra superior é possível alterar a língua do sistema para Espanhol, Inglês e Português e visualizar a versão do sistema. As tecnologias utilizadas para o desenvolvimento da autenticação foram o Laravel Jetstream com Bootstrap.

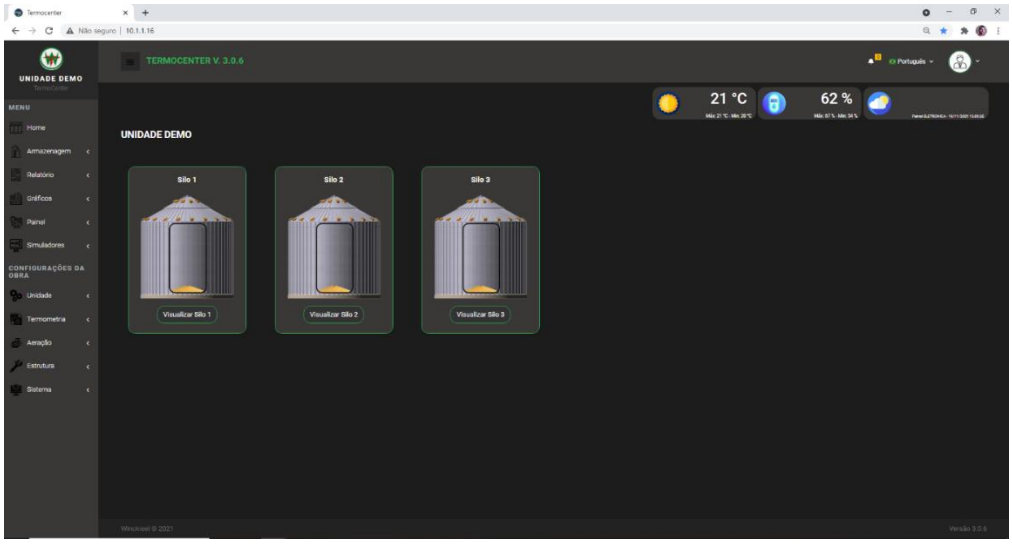
Figura 14 – Tela de login do sistema



Fonte: Elaborado pelo autor

Depois de efetuado acesso ao sistema com sucesso, o sistema busca todos os silos/armazéns gravados no banco de dados, o último registro da estação informando a temperatura e umidade. Depois de buscar essas informações o sistema renderiza uma tela conforme vista na Figura 15. Ao lado esquerdo da tela fica o menu de navegação para outras telas e recursos do sistema. Na parte superior da tela ficam os logs do sistema e a opção de logout. No meio da tela é apresentado os silos/armazéns com a figura do nível (nível (em %) de ocupação do grão dentro do silo).

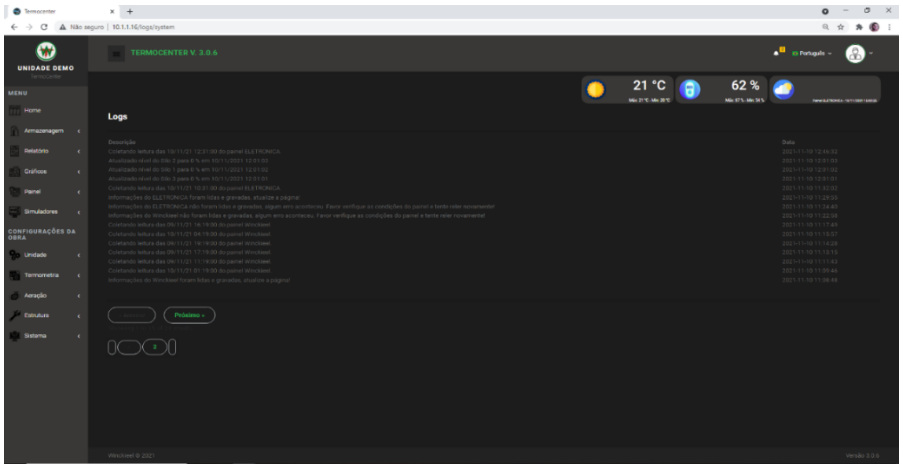
Figura 15 - Tela Home do sistema



Fonte: Elaborado pelo autor

Na tela dos logs o sistema retorna todos os eventos que aconteceram, ordenado pela data, mostrando na primeira página os logs mais recentes. Os *logs* apresentados na tela são eventos de coletadas de leituras, atualização do nível dos silos/armazéns, releitura da unidade, status de erro de operação, ações de exclusão de relatórios feitas por usuários.

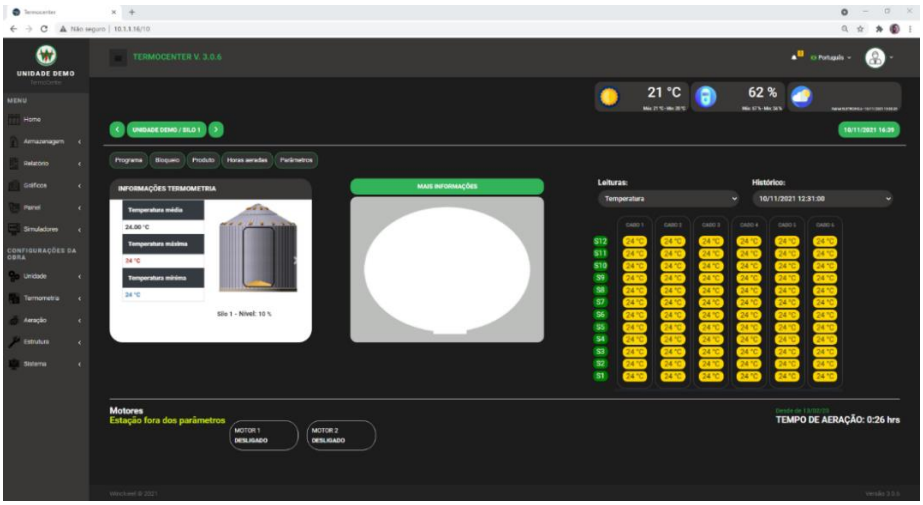
Figura 16 - Tela de logs do sistema



Fonte: Elaborado pelo autor

Na tela do silo o sistema busca todas as informações referente ao silo escolhido, conforme visto na Figura 17, é exibido as leituras coletadas do silo, status dos motores (ligado, desligado, falha de acionamento), tempo total de aeração em horas, isso se refere ao tempo em horas que os motores ficaram ligados, programa de aeração escolhido, status da aeração (bloqueado ou desbloqueado), produto alocado.

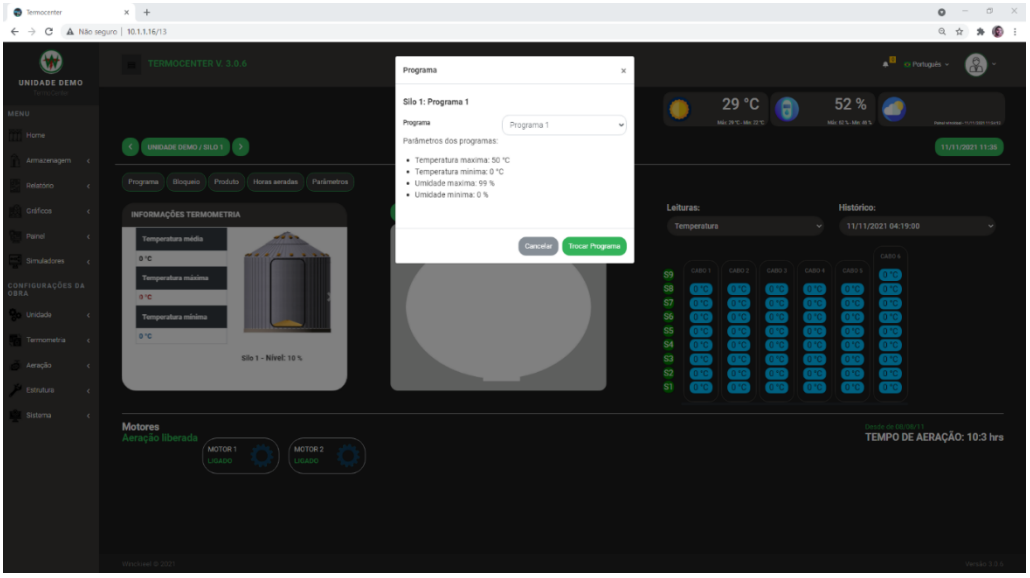
Figura 17 - Tela do silo/armazém do sistema



Fonte: Elaborado pelo autor

Na tela do silo selecionado é possível alterar o programa de aração que está sendo executado naquele silo. Conforme visto na Figura 18, ao acessar o programa um *model* será exibido mostrando o programa atual e sua parametrização.

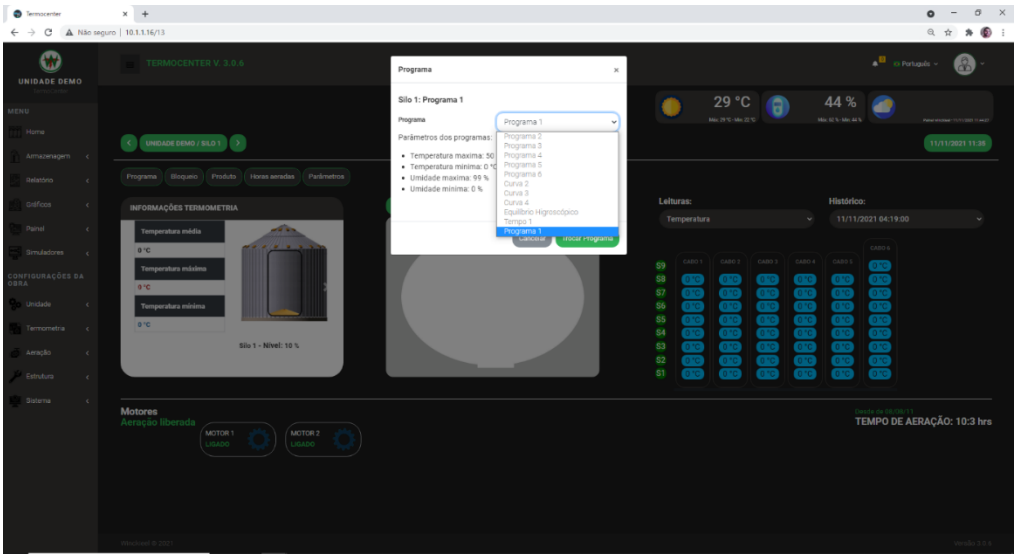
Figura 18 - Alteração do programa de aeração



Fonte: Elaborado pelo autor

Para alterar esse programa, o usuário acessa a lista de programas conforme Figura 19, ao selecionar o programa o sistema busca os parâmetros utilizados por aquele programa e envia ao painel.

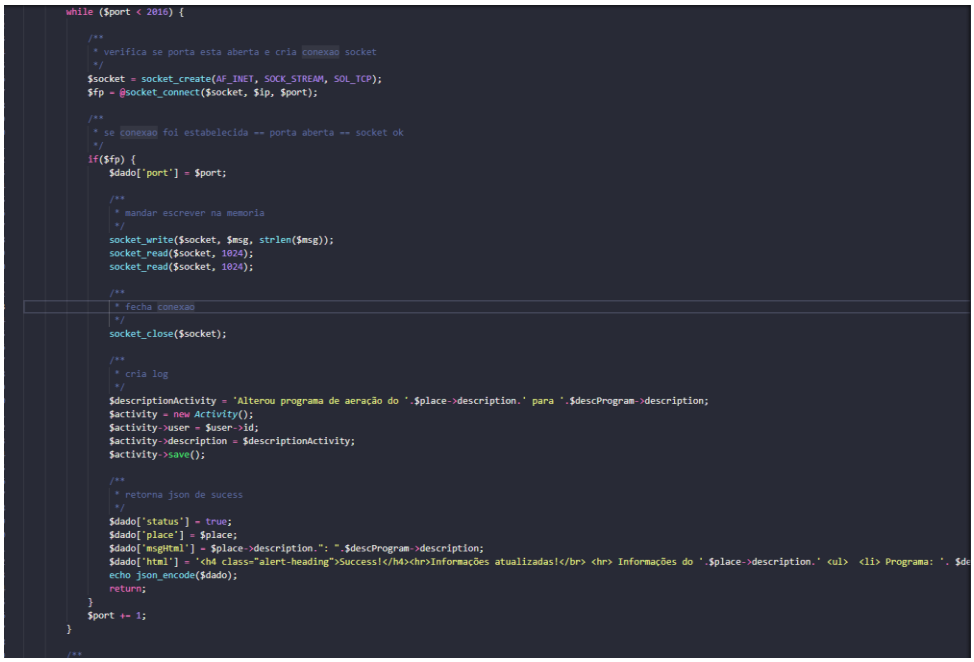
Figura 19 - Lista de programas de aeração



Fonte: Elaborado pelo autor

Conforme visto na Figura 20, essa parte do código é responsável pela atualização do programa do silo no painel termocenter. O código desenvolvido em PHP realiza a conexão socket com o painel e envia a informação para o mesmo. Após o envio, o sistema cria um *log* de atividade com o usuário que realizou a ação e a descrição da atividade. Em seguida retorna um JSON de sucesso para a tela do usuário.

Figura 20 - Código PHP responsável pela atualização do programa de aeração



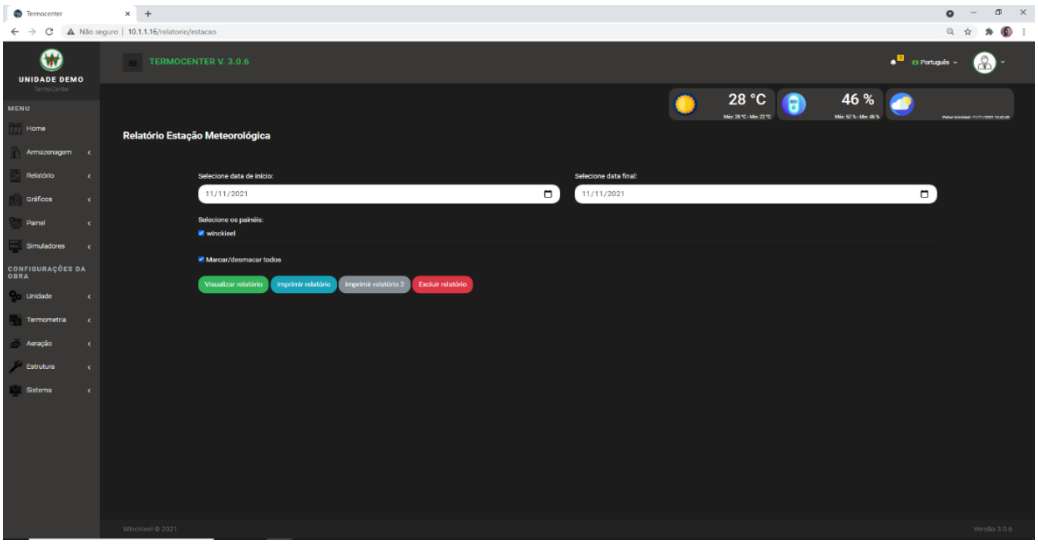
```
while ($port < 2016) {

    /**
     * verifica se porta esta aberta e cria conexao socket
     */
    $socket = socket_create(AF_INET, SOCK_STREAM, SOL_TCP);
    $fp = @socket_connect($socket, $ip, $port);

    /**
     * se conexao foi estabelecida == porta aberta == socket ok
     */
    if($fp) {
        $dado['port'] = $port;

        /**
         * mandar escrever na memoria
         */
        socket_write($socket, $msg, strlen($msg));
        socket_read($socket, 1024);
        socket_read($socket, 1024);

        /**
         * fecha conexao
         */
        socket_close($socket);

        /**
         * cria log
         */
        $descriptionActivity = 'Alterou programa de aeração do '.$place->description.' para '.$descProgram->description;
        $activity = new Activity();
        $activity->user = $user->id;
        $activity->description = $descriptionActivity;
        $activity->save();

        /**
         * retorna json de sucess
         */
        $dado['status'] = true;
        $dado['place'] = $place;
        $dado['msgHtml'] = $place->description.": ".$descProgram->description;
        $dado['html'] = '<h4 class="alert-heading">Success!</h4><hr>Informações atualizadas!</br> <hr> Informações do '.$place->description.' <ul>  <li> Programa: '. $de
        echo json_encode($dado);
        return;
    }
    $port += 1;
}

/**
```

Fonte: Elaborado pelo autor

Alteração da aeração para bloqueada ou desbloqueada, alteração do produto e as horas aeradas no silo seguem os mesmos princípios da alteração do programa de aeração. O usuário acessa o recurso que deseja, conforme visto na Figura 19. Em seguida é exibido um modelo com as opções, então o usuário toma a ação, após isso o sistema cria a conexão socket com o painel e atualiza os valores. Em apêndices estão as imagens do model e parte dos códigos que realizam essa ação.

Através da tela do relatório da estação meteorológica desenvolvida pela engine Blade do Laravel com Bootstrap, é possível visualizar o histórico de dados da estação em determinado intervalo de datas definidas pelo usuário, conforme Figura 21. O usuário seleciona das datas, selecionado o painel termocenter e qual opção de exibição deseja, podendo ser apenas a visualização em tela das informações, relatório em PDF ou exclusão das informações.

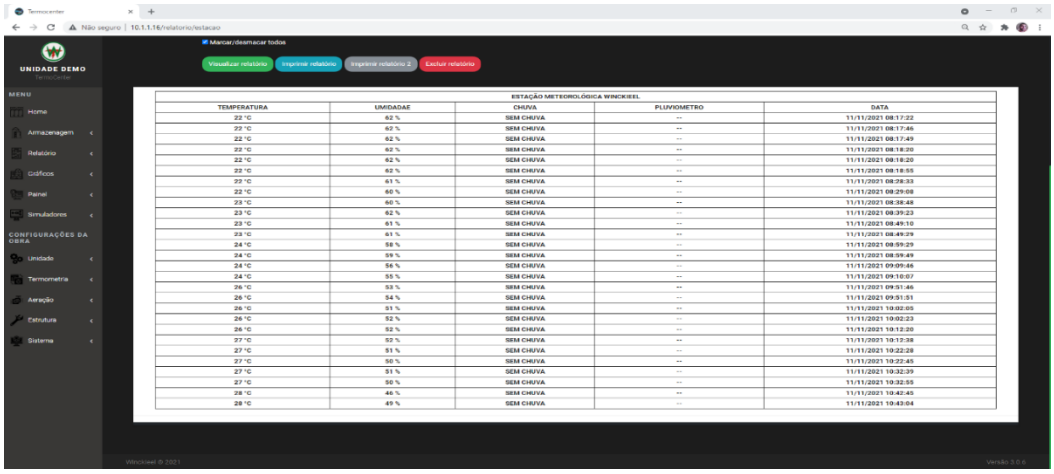
Figura 21 - Tela de relatório da estação meteorológica



Fonte: Elaborado pelo autor

A visualização do relatório, conforme visto na Figura 22, é exibida na tela uma tabela com as informações buscadas no banco de dados. São ordenadas de forma crescente pela data, mostrando a temperatura ambiente, umidade ambiente e status da chuva (chovendo ou não chovendo) e o pluviômetro quando existir, pois, existem estações que não marcam o pluviômetro em suas medições e com isso, este valor não é mostrado na tela.

Figura 22 - Tela de visualização dos dados da estação meteorológica



Fonte: Elaborado pelo autor

Conforme visto na Figura 23, esta é a função no *controller*, responsável por fazer a renderização da tela com as informações da Figura 22. No conceito MVC, ao acessar a rota /relatorio/estacao é acionado a função da Figura 21, no qual acessa o *model* das unidades, locais

de armazenagem (silos e armazéns) e os painéis, depois disso, é renderizado a *view* com as informações buscadas do model, que é a parte responsável por se conectar no banco de dados e retornar as informações solicitados no *controller*.

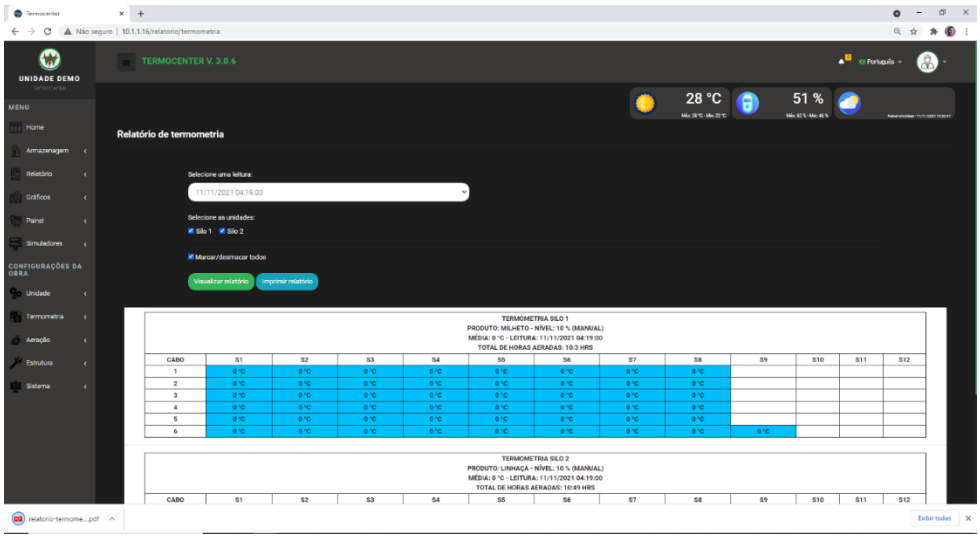
Figura 23 - Código PHP responsável exibição da tela de relatório da estação meteorológica

```php
/**
 * rota responsavel pela view relatorio da estação
 */
public function reportEstacao()
{
    /**
     * busca uma unidade
     */
    $unity = Unity::first();
    $units = [];

    /**
     * busca todos os silos/armazens
     */
    $locations_storage = PlaceStorage::orderBy('description', 'asc')->get();

    /**
     * busca todos os panels
     */
    $panels = Panel::all();

    /**
     * retona a view reportEstacao.blade.php
     */
    return view('pages.system.unity.reportEstacao', [
        'page' => 'viewReportSeason',
        'unity' => $unity,
        'units' => $units,
        'locations_storage' => $locations_storage,
        'panels' => $panels
    ]);
}
```

Fonte: Elaborado pelo autor

Os demais relatórios seguem o mesmo conceito do relatório da estação. Para os relatórios da termometria, conforme visto na Figura 24, o usuário seleciona a leitura que deseja visualizar ou exportar relatório em PDF. Em seguida seleciona as unidades que deseja ver e qual opção deseja visualizar, o relatório em tela ou relatório em PDF. O sistema retorna várias tabelas, sendo cada uma a leitura do silo, são exibidas informações sobre o silo no cabeçalho e os cabos dos silos com os valores da temperatura de cada sensor.

Figura 24 - Tela relatório da termometria

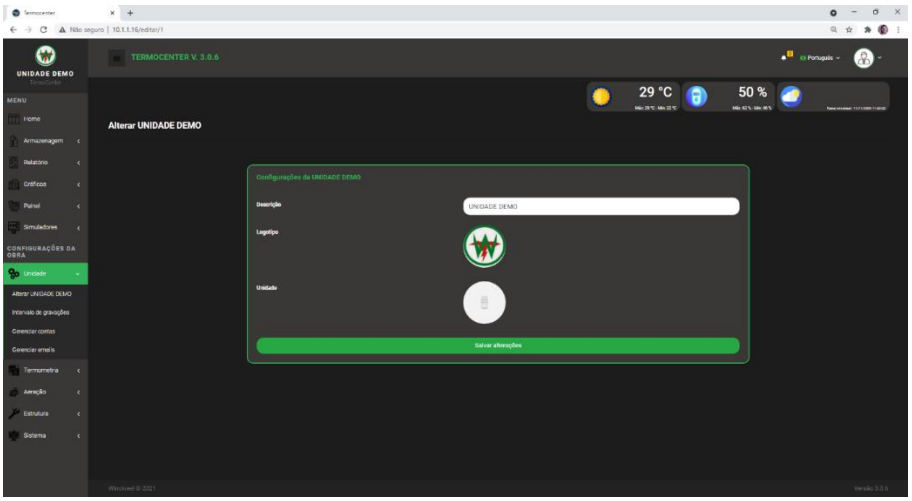


Fonte: Elaborado pelo autor

No relatório dos motores são exibidas as informações por intervalo de datas. São selecionadas as datas de início e término, os silos que deseja ver os status dos motores, a opção de visualização, sendo elas a visualização em tela ou relatório em PDF. O sistema retorna então uma tabela por silo, com a coluna de cada motor com o status (ligado, desligado ou falha de acionamento) com os dados da estação meteorológica e o horário dos status.

Através do sistema, conforme visto na Figura 25 é possível alterar as informações da logo da unidade, a imagem da unidade e o nome da unidade.

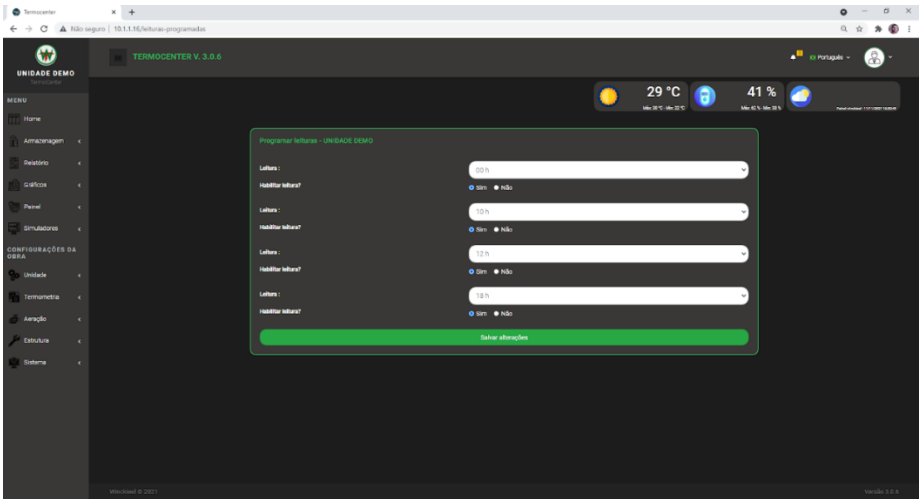
Figura 25 - Tela de alteração dos dados da unidade



Fonte: Elaborado pelo autor

Os painéis termocenter são capazes de realizar até 6 leituras programadas por dia, sendo pré-definidas 2 por dia às 4 horas da manhã, e 16 horas da tarde. Através do sistema é possível programar até 4 outras leituras. Conforme visto na Figura 26, o usuário seleciona o horário de cada leitura, marca se a leitura será feita ou não e salva as alterações.

Figura 26 - Tela de programação das leituras



Fonte: Elaborado pelo autor

Essas alterações são gravadas no banco de dados e enviadas ao painel via socket, na Figura 27, é possível visualizar parte do código responsável por criar a conexão socket e enviar essa informação ao painel termocenter. O código abaixo é feito em PHP, no qual realiza um *foreach* nas leituras habilitadas, conforme visto na Figura 26. A cada leitura o código cria conexão socket e envia a *string* ao painel, essa *string* precisa antes ser formatada com um padrão que o painel entenda e interprete as informações.

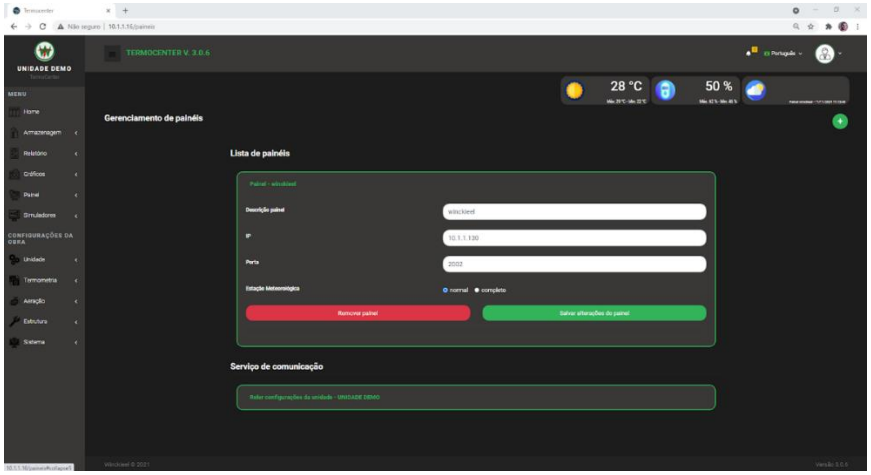
Figura 27 - Código PHP responsável pela atualização das leituras via socket

```php
$memorias = [14600, 14601, 14602, 14603];
// dd($memorias);
$indice = 0;
// dd($leituras);
$verifica = 0;
foreach ($leituras as $leitura) {
    $porta = 2004;
    while ($porta <= 2006) {

        $socket = socket_create(AF_INET, SOCK_STREAM, SOL_TCP);
        $fp = @socket_connect($socket, $ip, $porta);

        if($fp) {

            while (strlen($leitura->hour) < 3){
                $leitura->hour = "0$leitura->hour";
            }
            $msg = "wg".$memorias[$indice]."d".$leitura->hour;
            // echo $msg;

            /**
             * mandar escrever na memoria a hora da leitura
             */
            socket_write($socket, $msg, strlen($msg));
            socket_read($socket, 1024);
            socket_read($socket, 1024);

            /**
             * fecha conexao
             */
            socket_close($socket);
            $verifica += 1;

            break;
        }
        $porta += 1;
    }
    $indice += 1;
}
```

Fonte: Elaborado pelo autor

O sistema desenvolvido é capaz de lidar com mais de um painel, conforme visto na Figura 28, através do sistema web o usuário é capaz de adicionar, editar ou remover um painel.

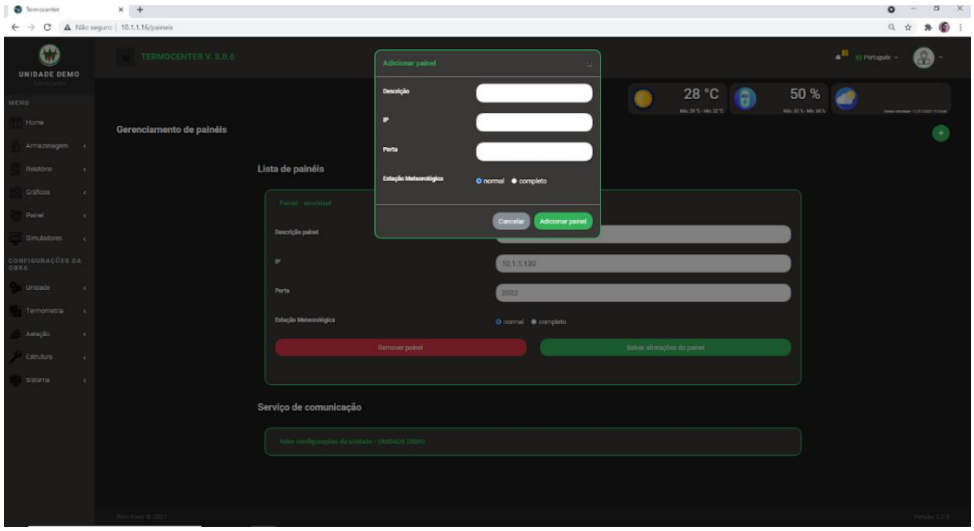
Figura 28 - Tela de gerenciamento de painéis



Fonte: Elaborado pelo autor

Para adicionar um painel ao sistema é necessário informar uma descrição do painel, o IP do painel, a porta de conexão e qual a estação meteorológica tem no painel. Conforme visto na Figura 29, algumas estações meteorológicas medem o valor do pluviômetro, sendo essas estações completas, estações sem o pluviômetro são estações normais.

Figura 29 - Tela para adicionar novo painel



Fonte: Elaborado pelo autor

O código da Figura 30 é responsável pela gravação do novo painel, após o usuário adicionar o painel, esta função é acionada. A primeira parte do código realiza a validação do IP do painel, caso não seja um IP válido e código e retorna um JSON com a mensagem de erro. Após a validação, caso seja um IP válido, é criado um objeto Panel e atribuído os valores informados na Figura 29. Após salvar os dados, é retornado um JSON com a mensagem de sucesso.

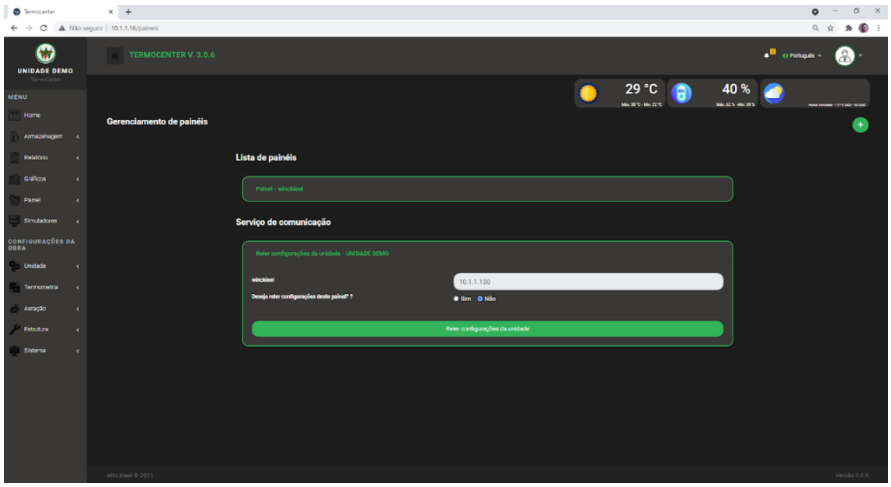
Figura 30 - Código PHP responsável por adicionar novo painel

```php
public function newPanel(Request $request)
{
    if (!filter_var($request->ip, FILTER_VALIDATE_IP)) {

        $return['status'] = 400;
        $return['msg'] = '<strong>IP inválido - '.$request->ip.'!</strong>Favor informe um IP válido! ';
        echo json_encode($return);
        return;
    }

    $panel = new Panel();
    $panel->description = $request->description;
    $panel->ip = $request->ip;
    $panel->port = $request->port;
    $panel->season = $request->season;
    $panel->save();

    $return['status'] = 200;
    $return['msg'] = '<strong>Informações atualizadas do '.$panel->id.'(id.'.$panel->id.') !</strong>Quando possível atualize a página! ';
    echo json_encode($return);
    return;
}
```

Fonte: Elaborado pelo autor

Após adicionar ou editar o painel, é necessário realizar a releitura das configurações do painel. Conforme visto na Figura 31, os painéis adicionados ao sistema são exibidos com seu IP e a opções de reler as suas configurações ou não. Por fim, o usuário manda reler as configurações.

Figura 31 - Tela de reler configurações do painel



Fonte: Elaborado pelo autor

Conforme Figura 32, o sistema executa comando para parar o serviço termocenter e inicia o serviço de *reset*. Este serviço de *reset* é responsável por buscar as configurações dos painéis e salvar no banco de dados. Essas configurações são a quantidade de silos, motores, estação e leituras do painel.

Figura 32 - Código PHP responsável por executar comando de inicialização do serviço *reset*

```php
/**
 * funcao respponsavel por parar o servico de comunicaçao e inincar o serviço de reler config
 */
public function arqPython()
{
    try {
        /**
         * para serviços
         */
        shell_exec('sudo systemctl stop termocenter.service');
        shell_exec('sudo systemctl start termocenterReset.service');

        /**
         * comando do windows
         */
        // $command = popen('start python C:\Users\Bruno\Desktop\termocenter\reset.py', 'w');

        /**
         * busca todos os paneis que estao habilitados para reset
         */
        $panels = Panel::where('reset', 'enabled')->get();
        foreach($panels as $panel){

            /**
             * deleta as informações daquele panel
             * por estarem com cascade
             * silos seramo removidos
             * motores
             * staus
             * estação
             * este dados seram removidoa
             */
            DB::table('locations_storage')->where('panel', $panel->id)->delete();
            // DB::table('eh_aeration_parameters')->where('panel', $panel->id)->delete();
            // DB::table('eh_aeration_parameters')->where('panel', $panel->id)->delete();
            // DB::table('free_aeration_parameters')->where('panel', $panel->id)->delete();
        }

        // $command = escapeshellcmd('C:\Users\Bruno\Desktop\termocenter\main.py');
        // $output = shell_exec($command);

        return 1;
    } catch (Exception $e) {
        dd('Execão capturada: ', $e->getMessage(), "\n");

        return 0;
    }
}
```

Fonte: Elaborado pelo autor

4.2 INTERFACE RESPONSIVA

As telas vistas anteriormente na seção 4.1 foram todas desenvolvidas com responsividade. Foi utilizado o *framework* de CSS Bootstrap, bastante conhecido e utilizado pelo mercado para seu desenvolvimento. Conforme visto na Figura 33, os conteúdos mudam de lugar e tamanho para ter uma boa experiência em usar o sistema pelo smartphone sem perder nenhum recurso ou funcionalidade.

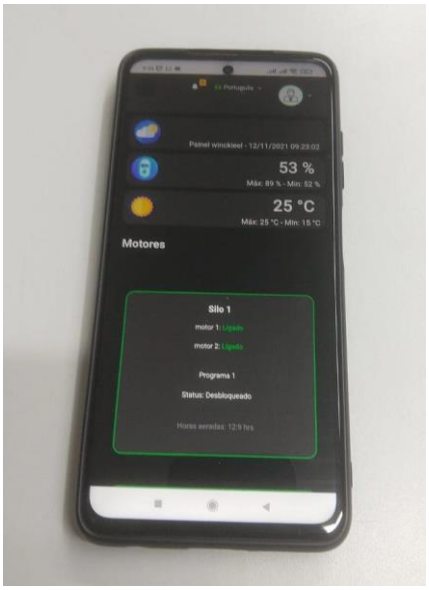
Figura 33 – Tela do silo/armazém do *smartphone*

Fonte: Elaborado pelo autor

Figura 34 - Tela *Home* no *smartphone*

Fonte: Elaborado pelo autor

Figura 35 - Tela dos status dos motores no *smartphone*



Fonte: Elaborado do autor

Foi utilizado também conhecimentos em CSS, através do recurso de @media screen do css foi feito alterações das propriedades das classes. Na Figura 36 é possível visualizar parte do arquivo *style* com o código CSS para tratar as diferentes telas. Através do comando @media screen o código consegue pegar a largura da tela em pixels e atribuir os códigos às classes.

Conforme Figura 36, telas com largura máxima de 1200 pixels recebem alteração nas classes ".col-sm-2" com os atributos flex: 0 0 24.666667% e max-width: 24.666667%. Em telas com largura máxima de 755 pixels recebem alteração nas classes ".col-sm-2" com atributos flex: 0 0 49.666667% e max-width: 24.666667%.

Figura 36 - Código CSS responsável pelo @media screen

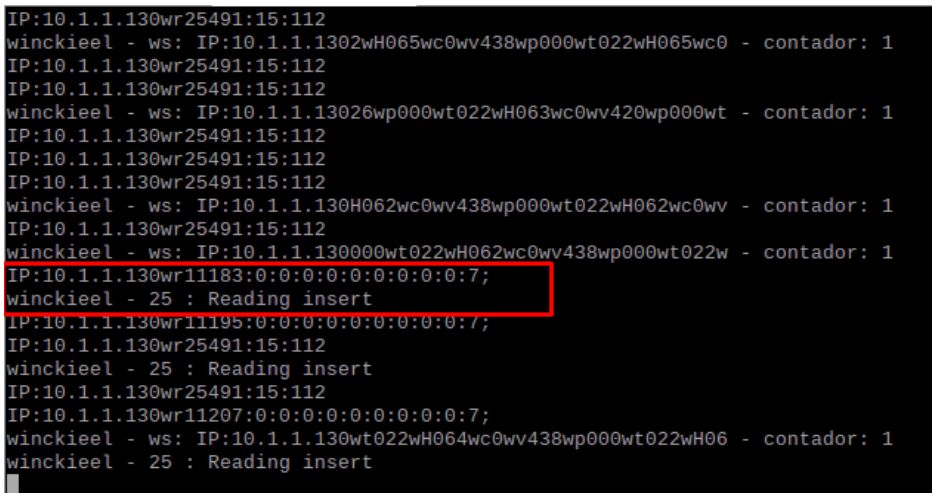
Fonte: Elaborado pelo autor

4.3 SERVIÇOS DE COLETA DE DADOS

Os serviços de coleta de dados são os responsáveis por ficarem coletando as informações do painel termocenter. A partir das leituras dos programas pelo usuário no sistema web, o painel termocenter realiza no horário programado a leitura dos sensores de temperatura dentro dos silos e armazéns, processa esses dados e os salva em sua memória. Um *script* em Python cria uma conexão socket ao painel e fica verificando se nessas posições de memórias existe algum

dado novo, quando houver, isso, quando o painel terminar de fazer a leitura, o *script* recebe essas informações, interpretada o dado, indexando de qual silo ou armazém é a informação recebida e salva no banco de dados. Na Figura 37 é possível visualizar o dado recebido do painel, onde cada valor entre : é referente a temperatura dos sensores instalados nos cabos dentro dos silos.

Figura 37 – Dados recebidos via socket dá termometria



Fonte: Elaborado pelo autor

Na Figura 38 apresenta parte do código responsável por salvar essa informação tratada dentro do banco de dados. Cria-se um objeto leitura, passa o id do cabo, grupo da leitura e os valores coletados mais a média calculada.

Figura 38 - Código Python responsável por adicionar dados da termometria no banco de dados

```
        '''
        placeReading = Reading()
        placeReading.conn = self.conn
        placeReading.cursor = self.cursor
        placeReading.cable = cable[0]
        placeReading.group = group.id
        placeReading.datas = text
        placeReading.average = average
        with self.mutex:
            placeReading.save()
        print('{} - {} : Reading insert'.format(self.panel['panel'], group.id))
```

Fonte: Elaborado pelo autor

A coleta das informações da estação meteorológica e status dos motores tem algumas regras diferentes. Existe um intervalo de gravação no banco de dados entre um dado da estação e um novo da estação conforme visto na Figura 39, passado os minutos configurados o *script* em Python coleta o dado via socket, trata e salva no banco de dados. Para uma informação da

estação meteorológica ser salva antes do intervalo configurado, deve acontecer uma mudança climática de chuva, quando o script interpreta a mudança climática de chuva para não chuva, ou vice-versa, é salva a informação coletada. Na Figura 39 é possível verificar o dado coletado pelo *script* Python. O valor após "wH" se refere a umidade ambiente, o valor após "wt" se refere à temperatura ambiente. Após "wc" o valor recebido se refere ao status da chuva, sendo 0 para não chovendo, e 1 para chovendo. O valor interpretado é: "Temperatura ambiente 21 °C, Umidade ambiente 58 % e não está chovendo no momento".

Figura 39 - Dados recebidos via socket da estação meteorológica

Fonte: Elaborado pelo autor

Na Figura 40 é possível visualizar o código Python responsável por enviar mensagem ao socket para receber os dados da estação.

Figura 40 - Código Python responsável por receber os dados estação meteorológica

```
        '''
            verifica esta comunicando
        '''
msg = "ws"
with self.mutex:
    retorno = thermometer.msg(msg)
print(f"{self.panel['panel']} - ws: {retorno} - contador: {tempo_excedido_estacao}")
```

Fonte: Elaborado pelo autor

A regra do intervalo de tempo configurado também vale para os status dos motores, porém se houver alguma mudança de status do motor de ligado para desligado, ou desligado para ligado, o *script* interpretando isso, salva imediatamente. Na Figura 41 é possível visualizar o dado recebido via socket de um status de motor. O valor coletado (valor 8 no exemplo da Figura 41) é referente a um CL com 7 saídas, esse valor é transformado para binário, o script interpreta as primeiras 7 posições, onde 1 é ligado e 0 é desligado. O *script* finaliza verificando qual motor fica no respectivo CPL e, após indexar, salva o status do motor com 0 ou 1.

Figura 41 - Dados recebido via socket dos status dos motores

Fonte: Elaborado pelo autor

Na Figura 42 é possível visualizar parte do código em Python que faz a conexão socket, recebe o status dos motores, trata e salva no banco de dados. O código desenvolvido, conforme visto na Figura 42, primeiramente realiza a montagem da *string* (a mensagem *text*) que irá ser enviado ao painel termocenter. Em seguida ele envia a mensagem ao painel e recebe a resposta, no qual armazena em dado. Como visto na figura 41, o retorno foi "IP10.1.1.96wr25490:8:104", então começa a ser feito o tratamento do dado, através do comando split do Python ele divide o dado retornando uma lista de 2 posições, ficando na posição 0 o valor de "IP10.1.1.96" e na posição 1 o valor de "25490:8:104". Sequentemente é utilizado novamente o comando split para retorno na posição 1, buscando dividir o valor entre os ":". Retornando assim uma lista com 3 posições, sendo na posição 0 o valor de 25490, referente a posição da memória no painel, na posição 8 o valor decimal dos status dos motores e na posição 2 o valor check utilizado pelo painel para certificar se o valor enviado está correto. O valor do status dos motores é transformado para binário com 7 posições. Por último, é realizado um for no binário de 7 posições onde o script busca a posição correta do motor, ao encontrar ele sai do for e insere o status na lista de valores de entrada, podendo ser 0 de desligado, ou 1 de ligado.

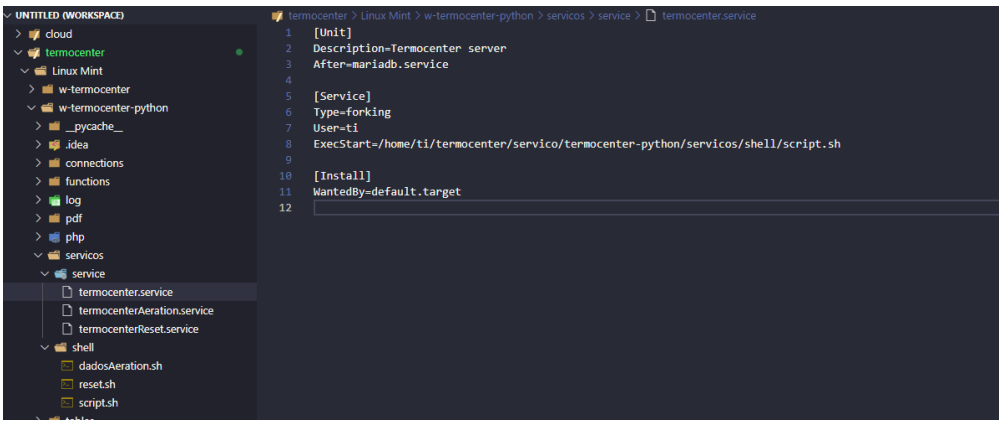
Figura 42 - Código Python responsável por receber os status dos motores

```python
'''
    busca valor socket da entrada desse motor
'''
input_memoria = input + (clp - 1)
text = f'wp{input_memoria}'
contFall = 0
while contFall < self.fall:
    with self.mutex:
        dado = thermometer.msg(text)
    if dado is not False:
        break
    contFall += 1
retorno = dado.split('wr')
sub_dados = retorno[1].split(':')

valor = int(sub_dados[1])

'''
    apos retorno dos status
    passa para binario
'''
binary = f'{valor:07b}'
cont = len(binary)
'''
    como ele retorna uma lista de 7 dados
    faço um for para pegar a posicao (exit) do motor atual
'''
for c in binary:
    if cont == exit:
        status = int(c)
        break
    cont -= 1
'''
    add o status a lista de entradas
'''
listEntradas.append(status)
```

Fonte: Elaborado pelo autor

Abaixo na Figura 43 estão os serviços Linux responsáveis por chamar os *scripts* Python. O arquivo. service chama um *script shell* (Figura 44) que executa os *scripts* Python em segundo plano.

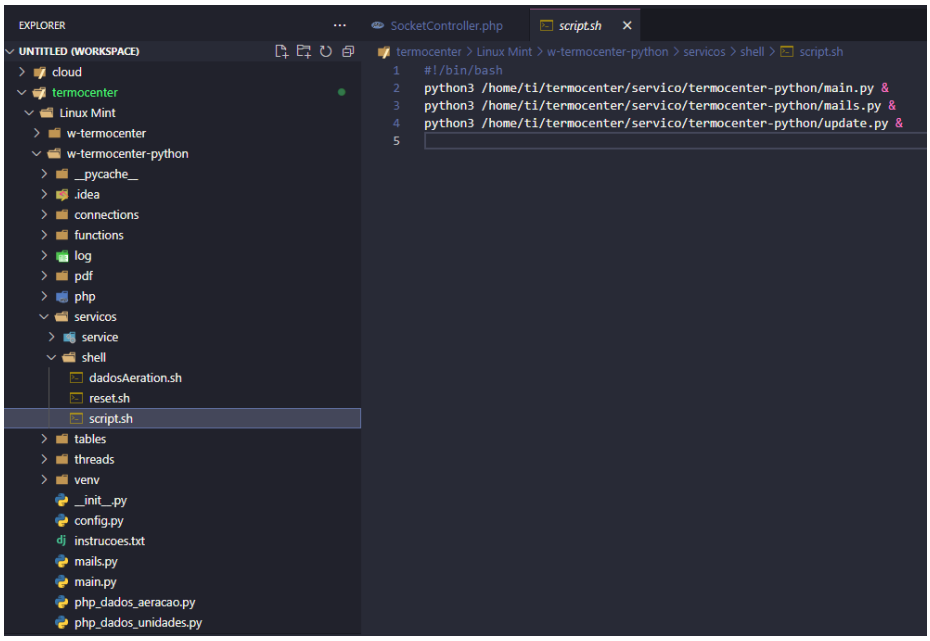
Figura 43 - Arquivo termocenter.service



Fonte: Elaborado pelo autor

Em Linux ao executar o *script* com & no final, irá executar o comando em segundo plano. Podendo assim executar vários códigos Python sem travar a execução de outro.

Figura 44 – Arquivo shell de inicialização dos scripts Python



Fonte: Elaborado pelo autor

Através dos serviços criados então é feito toda a coleta dos dados do painel termocenter e o armazenamento dos mesmos no banco de dados. Toda essa lógica fica invisível para o usuário, os serviços são iniciados automaticamente ao ligar o servidor sem que seja necessária alguma ação do usuário.

O trabalho aqui proposto conseguiu atingir os objetivos. O sistema consegue coletar os dados do painel termocenter e alimentar o banco de dados. Através do sistema web o usuário consegue visualizar essas informações armazenadas, exportar relatórios de termometria, da estação meteorológica, status dos motores, e assim, tomar decisões sobre seus grãos armazenados a partir desses dados. O acesso pode ser feito através de computadores e também por meio de smartphones, possibilitada pela interface responsiva, no qual era um dos grandes problemas do antigo sistema termocenter, que dificultava a usabilidade e experiência do usuário ao utilizar o sistema web através de dispositivos móveis.

5 CONSIDERAÇÕES FINAIS

O objetivo deste trabalho foi o desenvolvimento de um novo sistema de gerenciamento de unidades armazenadoras de grãos, possibilitando o histórico da termometria, status dos motores e parametrização da aeração e o acesso remoto via computador e smartphones. A nova versão do sistema foi desenvolvida em uma nova plataforma (Linux) o que reduz custo de equipamentos além de ser um novo sistema com conceitos de responsividade para smartphones.

O primeiro passo do trabalho foi identificar, através do uso do antigo sistema quais os recursos necessários do sistema, seu funcionamento, parte lógica e comunicação com o painel. Através desse estudo foi possível identificar quais ferramentas iriam ser necessárias para o desenvolvimento do trabalho. Por meio do estudo do mapa memorial foi possível também estruturar a lógica e padrões dos dados para criar a modelagem do banco de dados do sistema.

Paralelamente, foi realizado um estudo sobre o *framework* Laravel para construção de aplicações MVC *(Model, View e Controller)* para web. Através do *framework* Laravel foi possível fazer a modelagem do banco de dados e sua configuração de forma rápida e fácil permitindo assim, ter uma base de dados para o armazenamento de informações.

Com os recursos do *framework* Laravel foi possível construir todo o sistema web responsivo fornecendo suporte para geração de relatórios, análise gráfica e conexão socket para alteração dos parâmetros do painel termocenter.

A última parte do trabalho foi construir serviços no servidor para coleta de dados da termometria, estação meteorológica e status dos motores do painel de forma automática. Aproveitando o estudo do mapa memorial, foi desenvolvido *scripts* em Python que realiza a coleta desses dados via socket e gravava no banco de dados. Com o uso do serviço foi possível construir *scripts* em Python para automatização de relatórios e *logs* ao usuário.

A conclusão final é que através desse novo sistema para controle e gerenciamento de unidades armazenadoras de grãos foi possível reduzir custos reais com compra de equipamentos Windows, e fornecer uma nova receita para a empresa. Salienta-se ainda que foi corrigido alguns problemas encontrados no antigo sistema no qual não coleta as informações por erro de código, e não havia coerência em algumas informações apresentadas na tela.

REFERÊNCIAS

AZEVEDO, Loianny Faria et al. **A capacidade estática de armazenamento de grãos no brasil**. Disponível em: <http://abepro.org.br/biblioteca/enegep2008_TN_STP_069_492_11589.pdf>. Acesso em 03 de jun. 2021.

BARTHOLOMEW, Daniel. **MariaDB vs. MySQL**. Disponível em: <http://rozero.webcindario.com/disciplinas/fbmg/abd3/MariaDB_vs_MySQL.pdf>. Acesso em 30 de jun 2021.

BORGES, L. E. **Python para Desenvolvedores,** 2ª Edição, Edição do autor, Rio de Janeiro, 2010.

CLOUD, H. A.; MOREY, R.V. **Management of stored grain with aeration**. St. Paul: University of Minnesota, 1979. 8p.

CORREIRA, F. **Java Socket**. Disponível em: <http://wmagician.wordpress.com/2008/03/05/java-sockets/ >. Acesso em 03 de jun. 2021.

Empresa Brasileira de Pesquisa Agropecuária – EMBRAPA. Disponível em: <https://www.embrapa.br/soja/cultivos/soja1>. Acesso em 03 de jun. 2021

FERRARI, Fabrício Augusto, **Curso prático de Linux,** São Paulo: Digerati book, 2007.

FERREIRA, Rubem, **Linux Guia do Administrador do Sistema**. 1. ed. São Paulo: Novatec,2005.

FIGUEIREDO, Kleber. **Gestão estratégica de armazenagem.** Rio de Janeiro: CEL-COPEAD, 2004. FLEURY, P. F., WANKE, P.;

FIGUEIREDO, K. **Logística empresarial: a perspectiva brasileira.** São Paulo: Atlas, 2000.

GRAEFF, Romeu. **Demistifi cação do controle de qualidade da massa de grãos. Grãos Brasil: Da Semente ao consumo.** Maringá: a. I, n. II, p.25-26, mar.2002.

Hopson, K., Ingram, C, E, Stephen, E., (1997); **Desenvolvendo Applets com Java.** Editora Campus, pag. 335-351.

KAY, Russell. Python. **Computerworld**, 09 maio 2005. Disponível em: <http://www.computerworld.com.au/index.php/id;826423396;fp;2;fpid;523913170> Acesso em 03 de jun de 2021.

MORIMOTO, C. E. **"Redes e Servidores Linux: guia prática"**. 2ª ed. – Porto Alegre: Sul Editores. 2006.

ORACLE. **O Que É um Banco de Dados?**[S.l.: s.n.], 2020.Disponível em: <https://www.oracle.com/br/database/what-is-database/>. Acessa do em: 19 de jun de 2021.

RECKERS, Filipe B.; SILVA, Juliano. Linux x Windows nas Empresas. Disponível em: . Acesso em 03 de jun de 2021.

Redação Impacta. **10 Linguagens de Programação em Alta no Mercado para 2021,** 30 nov. 2020. Disponível em: https://www.impacta.com.br/blog/linguagens-de-programacao-em-alta-2021/. Acesso em 03 de jun de 2021.

REZENDE, Arnaldo Cavalcanti. **Armazenagem de grãos. Grãos Brasil: Da semente ao consumo.** Maringa, PR., n.35, ano VIII, p. 23-25, marlabr 2009.

SELIM, José M. Linux e Software Livre nas Empresas. Disponível em: <http://www.linuxhard.org/publicacoes.php?acessar=publicacao&id_texto=876>. Acesso em 12 de novembro de 2021.

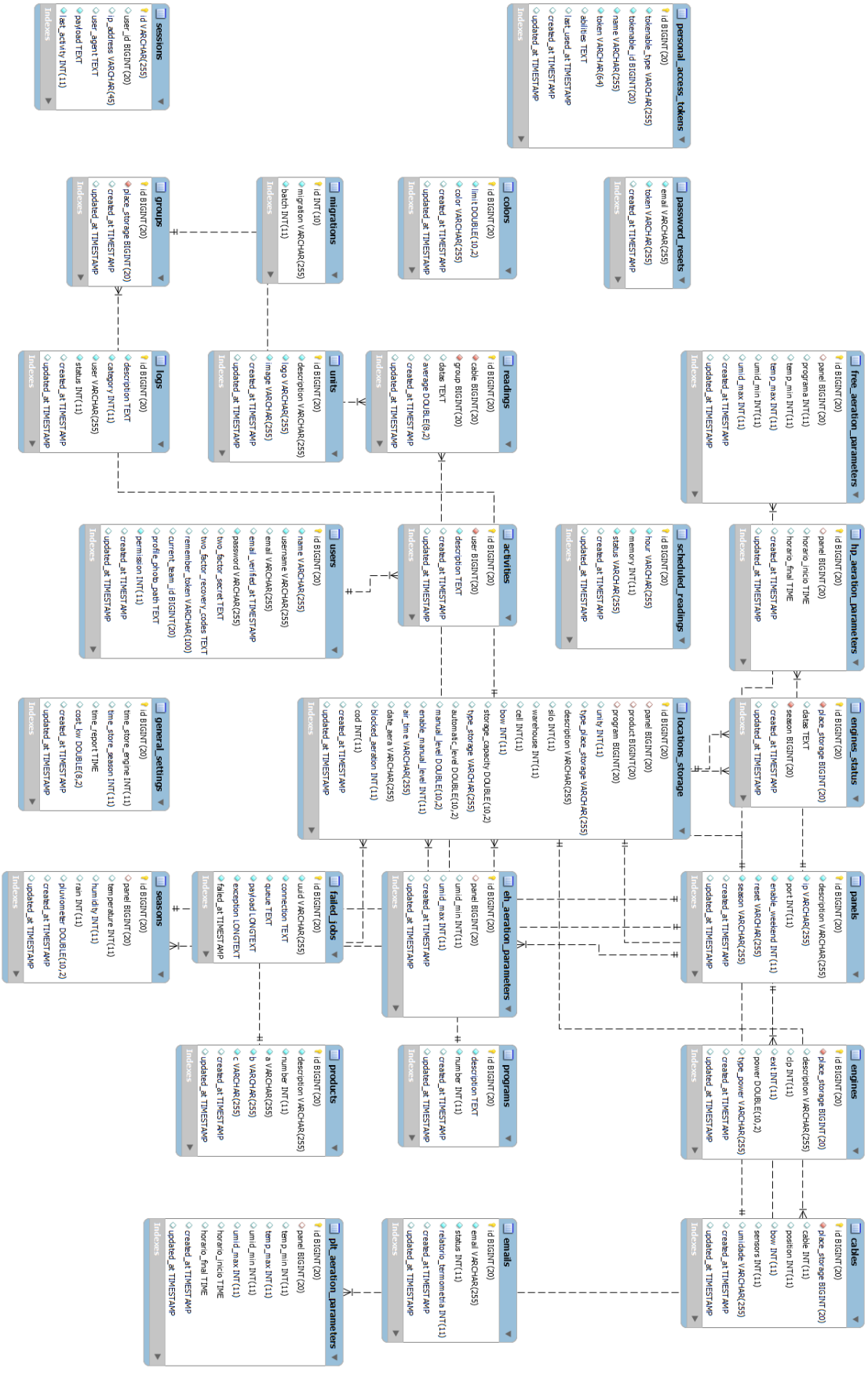
STALLINGS, W. **Redes e sistemas de comunicação de dados: teoria e aplicações corporativas.** 5. ed. Rio de Janeiro: Campus, 2005. xvi, 449 p.

SONGINI, Marc L. **Put in Plain Language: The high portable, object-oriented Python language moves into enterprise application development. Computerworld.** 12 set. 2005. Disponível em: <http://www.computerworld.com/softwaretopics/software/story/0,10801,104484,00.html>. Acesso em 03 de jun de 2021.

VERMA, A. (2014). **MVC Architecture: A Comparative Study Between Ruby on Rails and Laravel. Indian Journal of Computer Science and Engineering (IJCSE).** Disponível em: <http://www.ijcse.com/docs/INDJCSE14-05-05-053.pdf>. Acesso em 30 de jun de 2021..

WEBER, Erico Aquino. **Excelência em beneficiamento e armazenamento de grãos.** [S.l.]: Editora Salles, 2005.

APÊNDICE A – DIAGRAMA ENTIDADE RELACIONAL



ANEXO A – CARTA DE AUTORIZAÇÃO

CARTA DE AUTORIZAÇÃO

Eu, Loni L. Winck, GERENTE WINCKIEEL tenho ciência e autorizo a realização da pesquisa intitulada **Sistema De Gerenciamento Da Termometria E Automação Da Aeração Para Unidades Armazenadoras De Grãos**, sob responsabilidade do pesquisador **Bruno Pautz de Oliveira**, na empresa **WINCKIEEL INDUSTRIA DE EQUIPAMENTOS ELETRICOS AGROINDUSTRIAIS LTDA**. Para isto, serão disponibilizados ao pesquisador documentos para análise, painel termocenter, sistema termocenter.

Panambi, RS. 11 de novembro de 2021.

LONI LUDWIG WINCK