UNIVERSIDADE TECNOLÓGICA FEDERAL DO PARANÁ
COORDENAÇÃO DE ENGENHARIA DE COMPUTAÇÃO
CURSO DE ENGENHARIA DE COMPUTAÇÃO

ABEL DOS SANTOS MENEZES

SISTEMA DE MONITORAMENTO PARA SILOS DE CEREAIS

TRABALHO DE CONCLUSÃO DE CURSO

TOLEDO
2019

ABEL DOS SANTOS MENEZES

SISTEMA DE MONITORAMENTO PARA SILOS DE CEREAIS

Trabalho de Conclusão de Curso apresentado ao Curso de Engenharia de Computação da Universidade Tecnológica Federal do Paraná - UTFPR Campus Toledo, como requisito parcial para a obtenção do título de Bacharel em Engenharia de Computação.

Orientador: Prof. Dr. Tiago Piovesan Vendruscolo
Universidade Tecnológica Federal do Paraná

TOLEDO
2019

Ministério da Educação
Universidade Tecnológica Federal do Paraná
Campus Toledo
Coordenação do Curso de Engenharia de Computação

UTFPR
UNIVERSIDADE TECNOLÓGICA FEDERAL DO PARANÁ

TERMO DE APROVAÇÃO

Título do Trabalho de Conclusão de Curso Nº 03

Sistema de monitoramento para silos de cereais

por

Abel dos Santos Menezes

Esse Trabalho de Conclusão de Curso foi apresentado às **13h30 do dia 25 de novembro de 2019** como **requisito parcial** para a obtenção do título de **Bacharel em Engenharia de Computação**. Após deliberação da Banca Examinadora, composta pelos professores abaixo assinados, o trabalho foi considerado **APROVADO**.

Prof. Me. Bruno Meneghel Zilli
UDC

Prof. Dr. Elder Elisandro Schemberger
UTFPR-TD

Prof. Dr. Tiago Piovesan Vendruscolo
Orientador - UTFPR-TD

O termo de aprovação assinado encontra-se na coordenação do curso

Toledo, 25 de novembro de 2019

Dedico este trabalho para a minha família e todos os que me ajudaram nesta jornada.

AGRADECIMENTOS

A Deus por ter me dado saúde, a minha família e amigos por terem me dado força e suporte para superar as dificuldades.

A Universidade Tecnológica Federal do Paraná, pela oportunidade de fazer o curso, ao corpo docente pelas aulas e ensinamentos ministrados. Sou grato a coordenação do curso de Engenharia de Computação pelo acompanhamento oferecido, agradeço também ao Adriano do Laboratório de Circuitos Elétricos pela disponibilidade e auxílio.

Agradeço o meu orientador Prof. Dr. Tiago Piovesan Vendruscolo, sem ele este trabalho não teria sido realizado, agradeço em especial meus amigos Luís, Matheus e Eduardo que me ajudaram no desenvolvimento e nos testes realizados.

A educação é um elemento importante na luta pelos direitos humanos. É o meio para ajudar os nossos filhos e as pessoas a redescobrirem a sua identidade e, assim, aumentar o seu auto-respeito. Educação é o nosso passaporte para o futuro, pois o amanhã só pertence ao povo que prepara o hoje. (X, Malcon).

RESUMO

MENEZES, Abel dos Santos. Sistema de monitoramento para silos de cereais. 2019. 61 f. Trabalho de Conclusão de Curso – Curso de Engenharia de Computação, Universidade Tecnológica Federal do Paraná. Toledo, 2019.

Este trabalho de conclusão de curso tem como objetivo o desenvolvimento de um sistema de monitoramento em tempo real das condições internas de um silo de cereais. Esse sistema é composto por um sistema embarcado e uma interface web que permite o acompanhamento online dos parâmetros do silo de cereais, sendo que esse acompanhamento pode ser efetuado por qualquer aparelho que possua acesso à internet. Essa interface é configurada para emitir alertas sempre que as condições internas do silo atinjam níveis inseguros em relação a possibilidade da ocorrência de uma explosão. Além disso, esse trabalho também se baseou em normas técnicas a respeito da segurança do trabalhador, de forma que o sistema possa informar se as condições internas do silo estão seguras caso o operador precise realizar alguma inspeção interna. Para a validação segura do sistema, foi desenvolvido um segundo sistema microcontrolado especialista responsável pela simulação dos sinais provenientes de um silo de cereais.

Palavras-chave: Sistemas Embarcados. API. Armazenamento de grãos. Monitoramento de silos.

ABSTRACT

MENEZES, Abel dos Santos. Monitoring system for grain silos. 2019. 61 f. Trabalho de Conclusão de Curso – Curso de Engenharia de Computação, Universidade Tecnológica Federal do Paraná. Toledo, 2019.

This final paper has the ain of develop a real-time monitoring system for the internal conditions of a grain silo. This system is composed by an embedded system and a web interface that allows online monitoring of grain silo parameters, the monitoring can be performed by any device with internet access. This interface is configured to issue alerts whenever the internal silo conditions reach unsafe levels in relation to the possibility of an explosion. Besides that, this work was based on technical safety standards, so, the system can inform when the silo's internal conditions are safe to the operator perform any internal inspection. For the secure validation of the system, a second expert microcontrolled system was developed to simulate the signals from a grain silo.

Keywords: Embedded systems. API. Grain storage. Silos monitoring.

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

API	*Application Programming Interface* (Interface de Programação de Aplicações)
C	Concentração da nuvem de poeira (g/m³)
CORS	*Cross-Origin Resource Sharing* (Compartilhamento de Recursos de Origem Cruzada)
CRUD	*Create, Read, Update and Delete* (Criação, consulta, atualização e deleção)
DAC	*Digital-to-Analog Converter* (Conversor Digital Analógico)
DER	Diagrama Entidade Relacionamento
DHCP	*Dynamic Host Configuration Protocol* (Protocolo de Configuração Dinâmica de Host)
E	Energia de ignição (J)
Ge	Gravidade de Explosão
HTML	*Hypertext Markup Language* (Linguagem de Marcação de Hipertexto)
HTTP	*Hypertext Tranfer Protocol* (Protocolo de transferência de hipertexto)
IBGE	Instituto Brasileiro de Geografia e Estatística
Ie	Índice de Explosividade
IPEA	Instituto de Pesquisa Econômica Aplicada
JSON	*JavaScript Object Notation* (Notação de objeto em JavaScript)
LII	Limite Inferior de Inflamabilidade
LSI	Limite Superior de Inflamabilidade
MER	Modelo Entidade Relacionamento
MIDI	*Musical Instrument Digital Interface* (Interface Digital para Instrumentos Musicais)
NFPA	*National Fire Protection Association* (Associação Nacional de Proteção contra Incêndios)
O	Concentração de O_2 (%)
PCB	*Printed circuit board* (Placa de circuito impresso)
Pmp	Pressão máxima de explosão (kg/cm²)
REST	*Representational State Transfer* (Representação por estado de transferência)
Si	Sensibilidade de ignição
SOAP	*Simple Object Access Protocol* (Protocolo de acesso simples de objeto)
T1	Temperatura de ignição do leito (°C)
T2	Temperatura de ignição da nuvem de poeira (ºC)
URI	*Uniform Resource Identifier* (Identificador Uniforme de Recurso)
Vmp	Velocidade máxima de aumento da pressão (kg/cm²)
XML	*Extensible Markup Language* (Linguagem de Marcação Extensível)

LISTA DE ALGORITMOS

SUMÁRIO

1 INTRODUÇÃO

Nos últimos anos o Brasil aumentou suas atividades vinculadas ao setor agrícola. Segundo Gasques, Bacchi e Bastos (2018) através do Instituto de Pesquisa Econômica Aplicada (IPEA) a produção de grãos passou de 40,6 milhões para 187 milhões de toneladas no período de 1975 a 2016. Para armazenar esse volume útil são necessários silos e armazéns. De acordo com o Instituto Brasileiro de Geografia e Estatística (IBGE) no segundo semestre de 2018 87,85% da produção de grãos no Brasil foram armazenados em silos, portanto, tal instrumento se faz de suma importância para o setor (IBGE, 2019).

Uma preocupação existente tange a respeito da ocorrência de explosões relacionados a estes ambientes de armazenagem. Tais explosões podem ocorrer em detrimento da poeira, gases, temperatura, umidade e outras características presentes na armazenagem e secagem dos grãos, podendo assim prejudicar toda a estrutura do estabelecimento.

Outro ponto importante a ser considerado é a manutenção correta dos parâmetros internos do silo, tendo em vista que alta temperatura e alta umidade aceleram a degradação do grão, reduzindo seu valor econômico.

Portanto, esse trabalho de conclusão de curso teve como objetivo o monitoramento interno e em tempo real de um silo de grãos, que em conjunto com normas técnicas de segurança, constituiu um sistema de monitoramento permanente mais amplo, prevenindo acidentes e falhas relacionadas à armazenagem de grãos. Esse monitoramento foi realizado com a utilização de um sistema microcontrolado.

1.1 Formulação de delimitação do tema

Como silos são de suma importância para o setor agrícola, a prevenção dos problemas relacionados a explosões em silos de cereais se mostra algo importante, de igual modo o emprego de um sistema embarcado busca a facilitação de alguns processos, provendo um hardware especialista para aquele determinado tipo de problema.

Deste modo este trabalho buscou a elaboração de um sistema composto por sistema embarcado, aplicativo e API e capaz de classificar e distribuir as informações referentes ao ambiente interno de um silo de cereais.

1.2 Situação problema

Casos como a explosão do silo de cereal no porto de Paranaguá (ESTADO, 2012), demonstraram que tais explosões podem ser de grandes proporções, prejudicando além do patrimônio, a integridade física dos trabalhadores envolvidos.

A medição e verificação atualmente é realizada presencialmente com o auxílio de um dispositivo que calcula o nível de gases e a temperatura interna do silo, expondo assim o

trabalhador a um ambiente perigoso.

1.3 Hipótese

Projetar um sistema que monitore e classifique as condições internas dos silos de cereais, sendo elas a temperatura, umidade, pressão, concentração do nível de poeira, concentração do nível de gás e nível de oxigênio a fim de notificar níveis internos críticos bem como demais informações a respeito do ambiente.

1.4 Objetivos

1.4.1 Objetivo geral

Desenvolver uma solução por intermédio do emprego de um sistema microcontrolado e um servidor capaz de monitorar as características cruciais do ambiente interno do silo, podendo assim, enviar notificações por intermédio de uma interface gráfica para o usuário sempre que o silo atingir níveis críticos de segurança.

1.4.2 Objetivos específicos

- Prover uma literatura coesa e coerente sobre o tema.
- Monitorar a pressão interna, a temperatura, a concentração de oxigênio, a concentração da nuvem de poeira, a concentração do gás sulfídrico e a umidade.
- Determinar o ponto critico de explosão.
- Classificar o ambiente interno em relação a periculosidade que o mesmo oferece ao operador.
- Arquitetura de projeto hábil para ser implementada em um ambiente real.

1.5 Justificativa

Silos de cereais estão sujeitos a explosões por serem ambientes em que as condições necessárias para uma explosão podem se satisfazer, a verificação dos silos geralmente é realizada por meio de auditorias físicas, expondo assim o funcionário a uma atmosfera danosa, segundo Cheremisinoff (2014) o pó é um produto gerado pelo processo relacionado à manipulação de grãos, e a exposição a este ambiente pode significar um risco ao trabalhador se o pó for inalado.

Portanto, o desenvolvimento desse trabalho explorou formas de amenizar este problema automatizando o sistema de verificação, alertando o operador no caso do ambiente do silo oferecer riscos a sua saúde e integridade.

2 FUNDAMENTAÇÃO TEÓRICA

2.1 Silos

Segundo Palma (2005) silos são construções destinadas ao armazenamento de produtos granulares ou pulverulentos em geral. Palma (2005) também define que silos no setor agrícola são comumente construídos utilizando chapas metálicas corrugadas.

A importância da utilização de silos se dá principalmente pela necessidade da estocagem de produtos e materiais em espaços reduzidos, no setor agrícola onde os silos são utilizados por cooperativas e produtores a estocagem ganha uma importância do ponto de vista econômico sendo utilizada para o controle do escoamento e abastecimento (PALMA, 2005).

2.1.1 Silos planos

Carneiro (1948) define que uma das condições importantes para a construção de silos está relacionada a estabilidade, esta estabilidade está relativa a resistência da estrutura quanto a ação do vento e o peso do material armazenado.

Sadowski e Rotter (2011) explicam que a pressão exercida na base do silo cilíndrico vertical aumenta conforme a quantidade do produto ensilado aumenta, portanto num projeto de silo deve se levar em consideração tais fatores. Um dos modos de se minimizar o efeito e a adoção de um projeto de silo cilíndrico de base plana, um exemplo de silo plano pode ser visto na Figura 2.1.

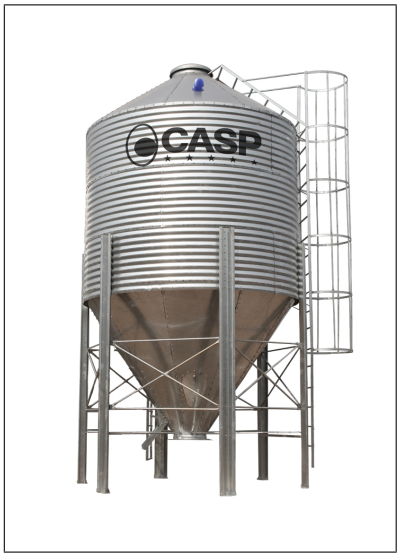
Figura 2.1 – Silo plano.

Fonte: Prado SPS (2018).

2.1.2 Silos elevados

Projetados para facilitar a descarga do grão armazenado eles são construídos apoiados em pilares de aço com bicas de descarga em 60 ou 45 graus, o acesso é feito por meio de escadas externas. Segundo a SILOGRÃO (2017) silos elevados são projetados para armazenar produtos granulares ou de fluxo livre. A estrutura de um silo elevado pode ser vista na Figura 2.2.

Figura 2.2 – Silo elevado.



Fonte: CASP (2018).

2.2 Explosões

Sistemas de armazenagem de grãos geralmente são compostos por equipamentos para secagem, moagem, escoamento, controle de peso e coletores de amostras, e um problema é a explosão ocasionada por excesso de pó, gases e temperatura.

Se tratando de explosões ocasionadas pela nuvem de poeira o meio onde o grão está armazenado é um fator determinante.

> "Ocorrem frequentemente em unidades processadoras em referência, onde as poeiras tenham propriedades combustíveis; é necessário, porém, que as mesmas estejam dispersas no ar e em concentrações adequadas."(Sá, 1997, p 1)

Explosões em silos são causadas quando os elementos fundamentais para queima (combustível, ignição e oxigênio) se encontram com uma reação em cadeia, elas ocorrem em relação à nuvem de pó, dimensão e concentração das partículas, oxigênio e fonte de ignição. Segundo Sá (1997) as explosões geralmente acontecem em série, configurando inicialmente uma pequena, mas suficiente eclosão capaz de movimentar o pó depositado criando assim uma situação propensa há novas ocorrências.

Segundo Cheremisinoff (2014) a menor amostra de poeira acumulada pode gerar explosões e incêndios.

Conforme abordado por Sá (1997) as explosões ocorrem dentro de uma faixa específica para cada material e esses valores foram estabelecidos previamente através de um experimento empírico realizado nos laboratórios da *National Fire Protection Association* (Associação Nacional de Proteção contra Incêndios) (NFPA) que consideraram as ocorrências de explosões nas minas de carvão da cidade de Pittsburgh. A Tabela 2.1 apresenta os níveis de classificação das explosões causadas pelo pó e na Tabela 2.2[1] se encontram alguns valores que foram extraídos pela NFPA.

Tabela 2.1 – Classificação explosões.

Tipo da Explosão	**Ie**
(P) Pequena	<0,1
(M) Moderada	0,1 – 1,0
(F) Forte	1,0 – 10
(MF) Muito forte	> 10

Fonte: (REATIVA, 2018) adaptado.

Tabela 2.2 – Poeiras agrícolas.

Tipo de pó	**Ie.**	**Si.**	**Ge.**	**Pmp**	**Vmp**	**T1**	**T2**	**E**	**C**	**O(%)**
Carvão de Pittsburg	**1**	**1**	**1**	**6.3**	**161**	**610**	**170**	**0.06**	**57**	**-**
Açúcar em pó	9.6	4	2.4	7.7	380	370	400	0.03	45	21
Alho desidratado	0.2	0.2	1.2	4	367	360	-	0.24	10	21
Arroz	0.3	10.6	0.5	3.3	49.2	510	450	0.1	85	21
Casca de arroz	2,7	1,6	1,7	7,7	282	450	220	0,05	55	17
Semente de arroz(casca)	1,4	1,1	1,3	4,3	9,3	490	-	0,08	45	12
Proteína de soja	4	1.2	3.3	6.9	457	540	-	0.06	40	12
Farinha de soja	0.7	0.6	1.1	6.6	56	550	340	0.1	60	15
Trigo bruto	2.6	1	2.5	5	155	500	220	0.06	0.5	12
Farinha de trigo	4,1	1,5	2,7	7	197	440	440	0,06	50	15
Pó cereal trigo	9,2	2,6	3,3	9,2	495	430	230	0,035	35	13
Palha de trigo	5	1,6	3,1	8,2	422	470	220	0,035	75	13
Polvilho de trigo	17.7	5.2	3.4	7	457	430	-	0.025	45	12
Milho	6.9	2.3	3	8	422	400	250	0.04	55	13
Casca de milho cru	12.1	3.1	3.9	8.7	387	410	390	0.04	40	17
Polvilho de milho	23.2	4.3	5.4	10.2	668	390	350	0.03	40	11
Semente de milho	5.5	2.5	2.2	8.9	260	450	240	0.045	45	12

Fonte: (REATIVA, 2018) adaptado.

[1] Os valores relacionados a pressão são a Pressão máxima de explosão (Pmp) e a Velocidade máxima de aumento da pressão (Vmp) que estão expressos em kg/cm², a Temperatura de ignição do leito (T1) e a Temperatura de ignição da nuvem de poeira (T2) estão expressas em graus célsius, a Energia de ignição (E) está expressa em joules, a Concentração da nuvem de poeira (C) está expressa em g/m³ e a Concentração de O_2 (O) em porcentagem.

Os parâmetros destacados na Tabela 2.2 apresentados também na lista de abreviaturas demonstram características do pó em questão, a Sensibilidade de ignição (Si) é a relação da temperatura de ignição com a energia necessária, a Gravidade de Explosão (Ge) é a relação entre a pressão máxima pela velocidade máxima de crescimento da pressão e o Índice de Explosividade (Ie) é a relação direta entre Si e Ge.

2.2.1 Dimensão da partícula

Segundo Sá (1997) quanto menor a dimensão da partícula em questão maior a facilidade para a nuvem que ela forma entrar em combustão, sua dimensão também intervém na velocidade do crescimento da pressão.

Outro fator é a capacidade elétrica das nuvens de pó, que podem culminar em pequenas descargas ocasionando assim uma ignição. Essa capacidade aumenta conforme a dimensão da partícula diminui, porém conforme explica Sá (1997) para se incendiar uma nuvem de pó é necessário alta energia, portanto, é pouco provável que explosões tenham sido ocasionadas por tal fenômeno.

2.2.2 Concentração

Para Sá (1997) a concentração é a medida em unidade de massa por unidade de volume, ou seja, é a quantidade de determinado material dentro de um espaço disponível dada pela quantidade de pó no espaço vago do silo.

Conforme registra a Tabela 2.2, cada tipo de pó possui um valor especifico de concentração mínimo. Cheremisinoff (2014) explica que geralmente a concentração das nuvens de poeira estão entre 50 e 100 g/m³ de limite inferior podendo atingir de 2 até 3 kg/m³ de limiar superior e que fora dessa faixa a maioria dos pós perdem suas características explosivas.

No experimento realizado pela NFPA os materiais ensaiados são relacionados aos valores do carvão de Pittsburgh com uma concentração de 0,5 kg/m³.

2.2.3 Umidade

Quanto maior a umidade do pó maior a temperatura necessária para a ignição, a umidade relativa contida no ar possui pouca influência nesse aspecto (CHEREMISINOFF, 2014).

Segundo Sá (1997) os testes referentes ao efeito da umidade realizados para criação da NFPA resultaram em valores que vão de 15 a 50%.

Outro fator importante sobre a umidade, é que ela acelera o processo de degradação dos grãos conforme seu nível aumenta, no entanto, esse trabalho não irá abordar esse processo detalhadamente.

2.2.4 Efeito de gases

Segundo Cheremisinoff (2014) aborda, conforme a concentração de O_2 aumenta maior a facilidade com que a ignição ocorrerá, e ambientes turbulentos propiciam explosões mais violentas.

A existência de gases inflamáveis na nuvem de pó colabora para explosões mais violentas, mesmo em baixas concentrações, aumentando assim a velocidade de expansão da pressão em relação a condições normais.

Conforme registrado na Tabela 2.2, os valores de concentração de oxigênio variam de acordo com cada material, os valores variam de 12 até 21%.

De acordo com Mainier e Viola (2005), o Gás sulfídrico ou ainda Sulfeto de hidrogênio é um gás incolor, mais denso que o ar e extremamente tóxico e o contato prolongado pode causar danos ao ser humano.

Mainier e Viola (2005) definem que no setor industrial este gás está relacionado aos processos de remoção química, lavagens de gases ácidos, tratamento de efluentes, fermentações, decapagens entre outras.

O Gás sulfídrico é um gás inflamável e segundo Scardino (2017) possui os limites de inflamabilidade no valor de 4,30 % para o (LII) e 15,50 % volume total do espaço para o (LSI).

Conforme demostra a Figura 2.3, os limites definem a faixa em que o gás pode inflamar, sendo menor que LII ou maior que LSI o gás não oferece perigo de inflamabilidade, porém ainda pode oferecer perigos a saúde. Conforme explicam Mainier e Viola (2005) os limites são calculados em relação ao a porcentagem do volume total do espaço.

Figura 2.3 – Inflamabilidade.

Fonte: Elaboração própria.

2.2.5 Fonte de ignição

Conforme registra a Tabela 2.2, em média são necessárias temperaturas entre 300 e 600ºC para que ocorra a explosão. Segundo Sá (1997) as fontes comuns podem ser chamas, luzes, produtos defumadores, arcos elétricos, faíscas, filamentos incandescentes entre outros, a energia de ignição elétrica pode variar conforme os valores apresentados na Tabela 2.3.

Tabela 2.3 – Energia miníma de ignição.

Material	**Energia Miníma**
Pó de carvão	>1000mJ
Farinha	300-1000mJ
Açúcar	10-30mJ

Fonte: Elaboração própria.

Cheremisinoff (2014) relata que fontes de ignição podem estar relacionadas a superfície quente como fornalhas, brasas e faíscas, auto aquecimento provocado por reações químicas, impacto ou fricção provocados por equipamentos, equipamento elétrico e descargas eletroestáticas.

2.2.6 Efeitos do confinamento

Para Cheremisinoff (2014) o confinamento é necessário para que o processo de explosão da nuvem de poeira possa ocorrer, geralmente o confinamento está relacionado ao equipamento de armazenagem podendo também estar relacionado ao ambiente no caso do pó estar sendo deflagrado.

Sá (1997) registra que durante a explosão gases são formados liberando assim calor que por sua vez aumenta a temperatura do local, aumentando assim a pressão interna, uma contramedida é o emprego de sistemas de alívio para reduzir a pressão criada pelos gases a fim de diminuir os possíveis danos.

2.3 Espaço confinado

Segundo NR33 (2012) espaços confinados é a nomeação dada para qualquer área ou ambiente que possua meios limitados e a ventilação é insuficiente para remover contaminantes, ou seja, são espaços não projetados para a ocupação humana contínua onde pode existir a deficiência ou enriquecimento de oxigênio.

A Figura 2.4 demonstra um modelo de sinalização adotada espaços confinados conforme as especificações definidas na NR33 (2012).

Figura 2.4 – Placa Espaço confinado NR33.

Fonte: barrafire.com.br (2019).

2.3.1 Atmosfera de risco

Segundo ABNT (2001), atmosferas de risco são espaços que podem apresentar perigos relacionados a incapacitação, restrição da habilidade para auto-resgate, doenças ou ainda morte. As atmosferas conforme registra ABNT (2001) são classificadas por:

- **Gás/Vapor ou névoa inflamável:** é considerado como situação de risco quando é atingido 10% de seu limite inferior de inflamabilidade.
- **Poeira combustível:** é considerado situação de risco quando é atingido uma concentração superior ao limite inferior de explosividade.

A ABNT (2001) define como causas de risco em relação a concentração de oxigênio níveis abaixo de 19,5% sendo considerado nível de oxigênio baixo ou baixa oxigenização, ou acima de 23% sendo considerado nível de oxigênio alto.

2.4 *Web Service*

Segundo Tanenbaum (2011), a base da Web é a transferência de páginas do servidor para o cliente, essas páginas são em geral arquivos *Hypertext Markup Language* (Linguagem de Marcação de Hipertexto) (HTML) armazenados em algum ponto no servidor. O HTML com o passar do tempo foi agregando novas funcionalidades podendo apresentar páginas estáticas que são páginas que exibem sempre o mesmo conteúdo e páginas dinâmicas que variam de conteúdo mediante a necessidade (TANENBAUM, 2011).

Tanenbaum (2011) ainda explica páginas dinâmicas que consistem em aplicações que são executadas no cliente podem ser tão bem sucedidas como aplicações tradicionais e ainda dispensam a necessidade da instalação de programas separados. Uma das formas de implementar páginas dinâmicas consiste na criação de *Web Services*.

Segundo Fawcett (2012), *Web Services* são utilizados mediante a necessidade de prover uma integração entre comunicação e aplicações independentes da plataforma utilizada, essa integração ocorre utilizando o protocolos de rede da camada de transporte do modelo TCP/IP[2] e usualmente a linguagem *Extensible Markup Language* (Linguagem de Marcação Extensível) (XML). As arquiteturas utilizadas são ou o *Simple Object Access Protocol* (Protocolo de acesso simples de objeto) (SOAP) ou o *Representational State Transfer* (Representação por estado de transferência) (REST).

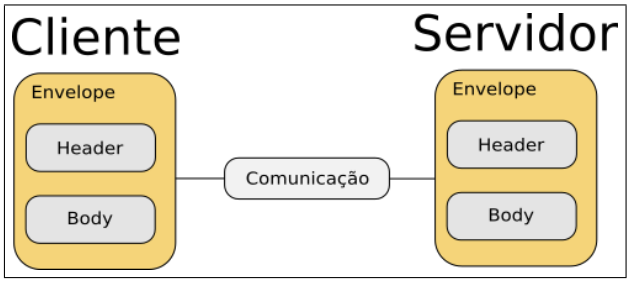
2.4.1 SOAP

Segundo Fawcett (2012) o SOAP é um protocolo de acesso que realiza chamadas remotas estruturadas para troca de informações, o protocolo é dividido em três partes:

- **Envelope:** Define a estrutura e codificação do conteúdo da mensagem.
- **Header:** Contém informações adicionais.
- **Body:** Contém a mensagem.

A fluxo de mensagens entre o cliente e o servidor SOAP ocorre conforme ilustra a Figura 2.5, a requisição SOAP é gerada no cliente, consumida no servidor e retornada para o cliente.

Figura 2.5 – Protocolo SOAP.



Fonte: Elaboração própria.

Tanenbaum (2011) define que SOAP é uma forma de implementar *Web Services* que realizam chamadas de procedimentos remotos entre programas de forma independente da linguagem e sistema operacional, com o cliente construindo uma requisição XML enviada para o servidor através do protocolo *Hypertext Tranfer Protocol* (Protocolo de transferência de hipertexto) (HTTP).

2.4.2 REST

REST é uma arquitetura descrita por Roy Fielding criada para fornecer uma forma de se trabalhar com aplicações para a WWW[3]. Segundo Fawcett (2012) o REST fornece interoperabilidade através da interface HTTP entre aplicações estabelecidas através de chamadas remotas

[2]Modelo em camadas que define os protocolos e a infraestrutura da *internet*.
[3]World Wide Web - Rede Mundial de Computadores.

conforme um conjunto de restrições, as restrições ou ainda recursos são definidas de acordo com a necessidade.

Uma aplicação REST conta com:

- **Recursos:** Elementos de informação;
- **Métodos:** Funções que manipulam os recursos.

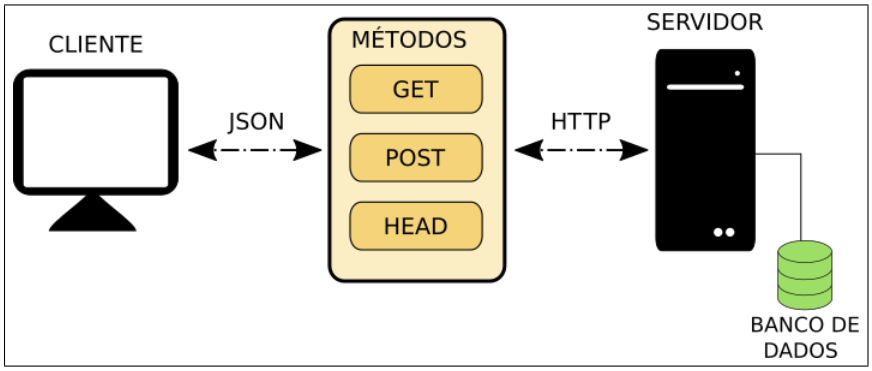
2.5 API

Segundo Masse (2011), uma *Application Programming Interface* (Interface de Programação de Aplicações) (API) é construída mediante a necessidade de integrar sistemas provendo uma interface que facilita o intercambio entre diversas aplicações, uma API diverge de um *Web Service* por não requirir que o cliente conheça o procedimento será chamado no servidor provendo assim a API uma maior flexibilidade.

Masse (2011) explica que na maioria dos casos uma API é construída utilizando a arquitetura REST através do emprego dos métodos HTTP, *Uniform Resource Identifier* (Identificador Uniforme de Recurso) (URI)'s e métodos *Create, Read, Update and Delete* (Criação, consulta, atualização e deleção) (CRUD).

A Figura 2.6 conta com um exemplo de uma API construída utilizando a arquitetura *REST*, o cliente realiza a chamada do método enquanto o servidor possui a implementação dos métodos e os recursos que serão manipulados, o servidor por sua vez depois de processar o requerimento retorna o *JavaScript Object Notation* (Notação de objeto em JavaScript) (JSON) (MASSE, 2011).

Figura 2.6 – API REST.



Fonte: Elaboração própria.

2.6 Trabalhos Correlatos

No artigo apresentado por Ferrasa, Biaggioni e Dias (2010) os autores apresentaram uma solução de monitoramento para silos de cereais através da utilização de sensores de temperatura e umidade via rádio.

Os principais pontos desenvolvidos pelos autores são:

- Monitoramento da temperatura através de sensores.
- Monitoramento da umidade através de sensores.
- Radiofrequência para o envio dos sinais.
- Interface gráfica para visualização escrita em JAVA.
- Protótipo funcional e aplicável.

Outro trabalho desenvolvido a respeito do monitoramento do ambiente de um silo foi desenvolvido por Citolin (2012), a monografia teve por objetivo o desenvolvimento de um sistema de termometria utilizando termistores NTC voltados para o monitoramento da temperatura a fim de solucionar o problema da deterioração dos grãos.

Os principais pontos apresentados pelo autor são:

- Monitoramento da temperatura utilizando termistores.
- Protocolo Modbus para o envio dos sinais do microcontrolador para o computador.
- Interface gráfica para visualização.

Esses trabalhos apresentaram bons resultados, no entanto, são limitados ao monitoramento de no máximo dois parâmetros: temperatura e umidade, além de não ter a possibilidade de disponibilizar os dados online. Tendo essas limitações como base, o presente trabalho objetiva preencher essas lacunas.

3 MATERIAIS E MÉTODOS

Neste capítulo é apresentado o levantamento de requisitos e a descrição detalhada acerca dos elementos relacionados ao desenvolvimento da solução.

3.1 Levantamento de Requisitos

De acordo com os tópicos abordados nos objetivos específicos e fundamentados no Capítulo 2, as condições necessárias para monitorar o silo simulado estão relacionadas à concentração da nuvem de poeira, umidade, concentração de oxigênio, fonte de ignição, temperatura, pressão e gás sulfídrico. O sinal proveniente dos sensores foi simulado, tendo em vista que para um teste real haveria a necessidade de um ambiente que pudessem ter todos os parâmetros em questão controlados[1].

A energia da fonte de ignição está ligada diretamente a temperatura interna ou a potência consumida pelos sensores estando assim ligada a corrente consumida, como o projeto utilizou a simulação dos demais sensores esta característica em relação a corrente foi sempre considerada em seu valor máximo desta forma a solução avalia a energia referente a potência desprezando a influência da temperatura que no caso dos silos representa uma parcela relativamente pequena.

Para a classificação se fez necessário um microcontrolador capaz de embarcar um sistema que consiga receber e interpretar os sinais dos sensores simulados e uma API que recebe, classifica e disponibiliza as informações geradas pelo microcontrolador.

Para disponibilidade e visualização dos dados se fez necessário o desenvolvimento de um aplicativo que garante o acesso ao sistema.

Logo o projeto requereu:

- Um *software* para simulação dos sensores.
- Dois microcontroladores de uso geral.
- Uma API responsável pela centralização da solução.
- Um banco de dados para armazenar as informações.
- Uma aplicativo para o acesso remoto do sistema.

Dados os requisitos optou-se pela escolha do programa MyOpenLab, um Arduíno UNO, uma interface de rede para Arduínos (*shield*), um Arduíno MEGA e o *framework* para desenvolvimento multiplataforma Ionic. As justificativas bem como a descrição do ferramental foram apresentados no decorrer do capítulo.

O Arduíno UNO responsável por simular em conjunto com o MyOpenLab e transmitir os sinais para o Arduíno MEGA que por sua vez interpreta e disponibiliza as informações em rede, através do *shield* de rede. A API construída utilizando Python, Flask, SQLAlchemy e a arquitetura REST, e o aplicativo do usuário com Ionic.

[1] Em ambientes reais, todos os sensores utilizados deverão ser de uso específico para ambientes confinados com atmosfera explosiva.

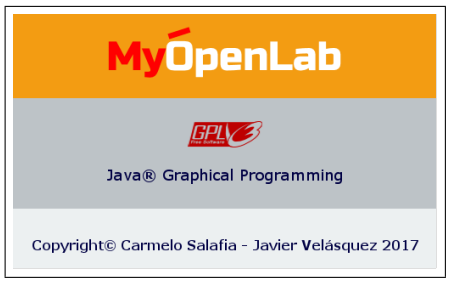
3.2 Materiais

A presente sessão teve por objetivo definir e descrever os materiais que foram necessários para a implementação do projeto.

3.2.1 MyOpenLab

Salafia e Velásquez (2017) definem que o MyOpenLab é um *software open-source* de desenvolvimento através de interfaces gráficas voltado para a criação de aplicações para plataformas de baixo custo, o *software* foi desenvolvido em Java e conta com protocolos de comunicação voltados a aplicações industriais. A Figura 3.1 demonstra a tela inicial do programa.

Figura 3.1 – MyOpenLab.



Fonte: myopenlab.com (2018).

O MyOpenLab conta com elementos voltados para a comunicação serial chamados de interfaces de entrada, contando com uma interface para execução de comandos do sistema operacional, uma interface para Raspberry PI e uma interface para Arduínos que funciona através do protocolo Firmata (SALAFIA; VELáSQUEZ, 2017).

O *software* conta com um painel de circuito onde os componentes e a lógica da programação é realizada, e um painel frontal que conta com os elementos gráficos exibidos durante a execução do programa.

3.2.2 Protocolo Firmata

Hoefs (2018) define o protocolo Firmata como um protocolo para comunicação entre o microcontrolador e um *software*. O protocolo foi desenvolvido baseado no formato definido como *Musical Instrument Digital Interface* (Interface Digital para Instrumentos Musicais) (MIDI) de mensagens com 8 bits para comandos e 7 bits para dados. O protocolo conta com uma biblioteca oficial implementada para Arduínos.

O MyOpenLab se utiliza do protocolo para realizar a comunicação com o Arduíno, conferindo ao programa a possibilidade controlar o microcontrolador a fim de realizar os experimentos.

O MyOpenLab foi escolhido para o desenvolvimento do projeto por se tratar de um programa estável, *open-source*, de fácil uso e com uma gama de ferramentas que possibilitaram a simulação dos sensores necessários para aferir as condições do silo.

3.2.3 Arduíno UNO

Segundo Arduíno (2019) o Arduíno UNO representado na Figura 3.2 é uma placa microcontroladora baseada no processador ATMEGA328P, possui 14 pinos digitais, 6 pinos analógicos.

Dos 14 pinos digitais 6 pinos podem ser utilizados para geração de sinais de PWM e possuem a frequência de aproximadamente 490 Hz nos pinos 3, 9, 10, 11 e 980 Hz nos pinos 5 e 6 (ARDUÍNO, 2019).

Figura 3.2 – Arduíno UNO R3.

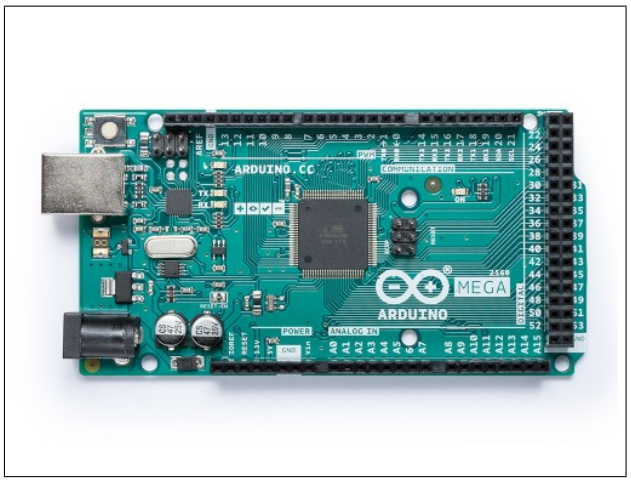
Fonte: (ARDUÍNO, 2019).

O UNO foi escolhido por ser uma plataforma conhecida com uma gama bibliotecas que facilitaram o desenvolvimento do projeto além de ser uma placa que o MyOpenLab oferece suporte.

3.2.4 Arduíno MEGA

Segundo Arduíno (2018) o Arduíno MEGA 2560 demonstrado na Figura 3.3 é uma placa microcontroladora baseada no processador ATMEGA2560, ela conta com 54 pinos digitais, 16 pinos analógicos e 4 portas para comunicação UART.

Figura 3.3 – Arduíno MEGA 2560.



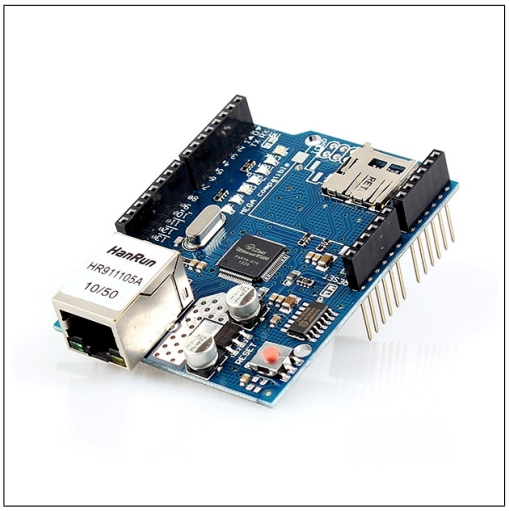
Fonte: (ARDUÍNO, 2018)

De maneira semelhante ao UNO o MEGA foi escolhido para o projeto por ser uma placa que o MyOpenLab oferece suporte além de ser uma placa com capacidade de memoria e processamento superior em relação ao UNO.

3.2.5 *Shield* de rede

O *shield* de rede representado na Figura 3.4 foi construído com base no controlador de internet W5100 criado pela WIZnet e, segundo WIZnet (2008), o W5100 oferece um interfaceamento para o modelo TCP/IP contando com os protocolos TCP, UDP, IPv4, ICMP, ARP e PPPoE, o chip oferece um endereço MAC próprio e uma memória interna de 16KB para a transmissão de dados.

Figura 3.4 – Shield Ethernet.



Fonte: FilipeFlop (2018).

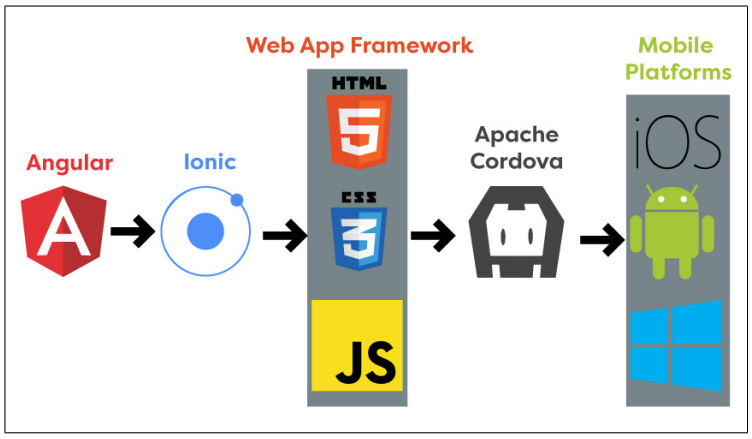
No projeto o *shield* se fez necessário dado que o MEGA não possui uma interface própria para comunicação em rede.

3.2.6 Ionic

Segundo a documentação o Ionic é um *framework open-source* de desenvolvimento multiplataforma que utiliza tecnologias e linguagens voltadas para o desenvolvimento Web, o Ionic possui um foco para a experiência do usuário criando interfaces minimalistas. Possui uma integração oficial com Angular, além de HTML, CSS e JavaScript.

Conforme apresenta a Figura 3.5, a aplicação em Ionic é desenvolvida utilizando o Angular, o Ionic então gera um *Web App* que é uma página web e o Cordova traduz essa aplicação para criar versões nativas para os dispositivos finais.

Figura 3.5 – Estrutura Ionic.



Fonte: code.tutsplus.com (2018).

3.2.7 Angular

Para Seshadri e Green (2014) o Angular é um *framework* desenvolvido por engenheiros do Google voltado para aplicações *Web*, se trata de um *framework* de grande modularidade e reusabilidade. O Angular possui uma estrutura semelhante ao modelo MVC[2] que separa as camadas de código conforme a sua usabilidade conferindo assim também uma boa manutenibilidade.

Seshadri e Green (2014) registram que o Angular trabalha com *data-binding* que permite ao Angular a possibilidade de trabalhar de forma bidirecional com os elementos HTML e os elementos de tela por meio do controlador, trabalhando com controle de dependências, declaração de chamadas e API'S.

3.2.8 Apache Cordova

Lopes (2016) define como um *framework open-source* voltado para aplicações híbridas, ele se alimenta da aplicação *web* para construir aplicações nativas voltadas para tecnologias

[2]MVC é o acrônimo de Model-View-Controller e é um padrão de projeto de software.

mobile.

Segundo Lopes (2016) o Cordova cria uma interface chamada de *WebView*[3] que será responsável por realizar as chamadas nativas equivalentes os comandos e estruturas *web* da aplicação original.

3.2.9 Ambiente de testes

Para o desenvolvimento do projeto foi necessário a utilização de dispositivos variados. Para realizar os testes, a API foi hospedada em um servidor local localizado numa rede local própria, deste modo todos os dispositivos necessários para a execução dos testes foram conectados a esta rede, a Tabela 3.1 descreve os dispositivos utilizados.

Tabela 3.1 – Descrição do Ambiente de testes.

	Celular	**Desktop**	**Servidor**
Processador	Snapdragon 625 2GHz 64bit	Intel Core i7-4510U 3.1GHz 64bit	Intel Core i7-4510U 3.1GHz 64bit
RAM	4 GB	8 GB	8 GB
Sistema	Android 9 API 28	Manjaro 18.1.2	Manjaro 18.1.2

Fonte: Elaboração própria.

3.3 Métodos

Com os materiais definidos os métodos utilizados no desenvolvimento do projeto podem ser abordados.

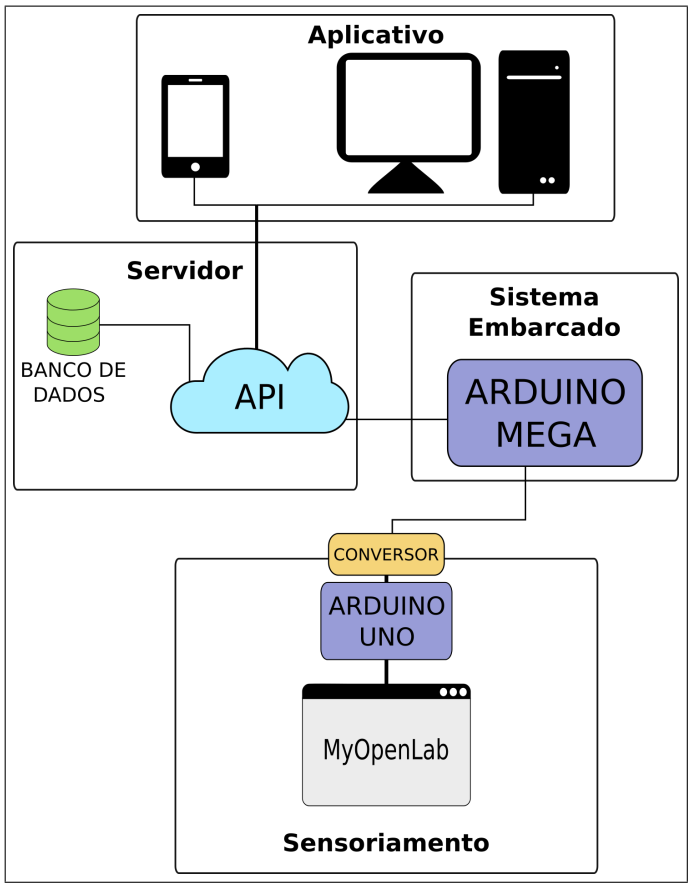
A Figura 3.6 demonstra o diagrama que representa a implementação da solução proposta no presente trabalho, a divisão se deu da seguinte forma:

- **Sensoriamento:** Nesta etapa foi realizada a simulação dos sensores referentes as características internas do silo.
- **Sistema Embarcado:** Nesta etapa foi realizada a aquisição dos sinais através da utilização do conversor, o sistema embarcado mapeia, classifica e disponibiliza as informações para a API.
- **Servidor:** O servidor composto pela **API** e pelo **Banco de Dados** é o elemento central do projeto.
- **Aplicativo:** Nesta etapa foi realizado o desenvolvimento da interface gráfica responsável por prover uma inteface em que o utilizador do sistema consiga cadastrar, visualizar, editar e excluir as informações referentes ao sistema.

Os detalhes de cada uma das partes do projeto serão apresentados no decorrer do capítulo.

[3] WebView é um componente que permite que apps exibam conteúdo da Web.

Figura 3.6 – Arquitetura do projeto.



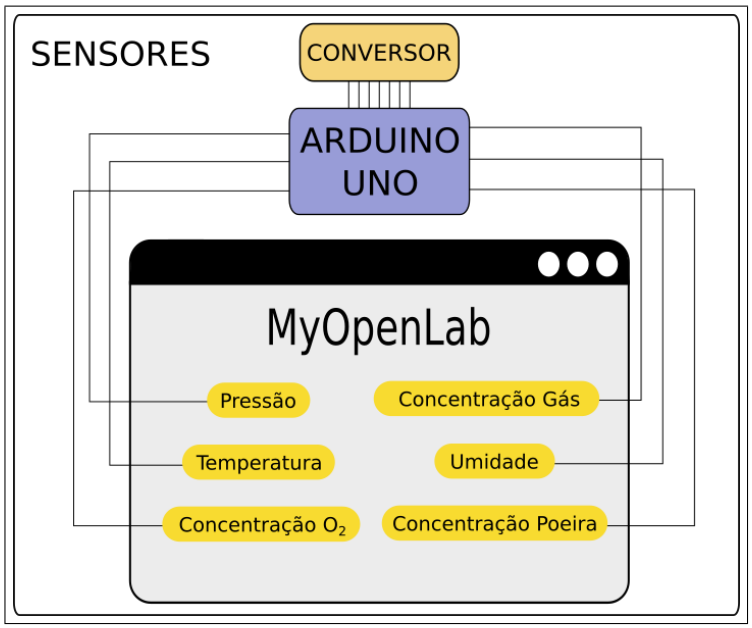
Fonte: Elaboração própria.

3.4 Sensoriamento

Embora o projeto tenha adotado uma metodologia para simulação dos sensores, um dos objetivos do projeto foi prover uma arquitetura hábil para uma implementação real, deste modo o sensoriamento foi construído para prover um funcionamento semelhante aos sensores reais, dessa forma, com o emprego dos sensores reais as adaptações são minímas.

Deste modo o UNO foi empregado no projeto para se comunicar com o MyOpenLab. Conforme demonstra a Figura 3.7 o MyOpenLab foi utilizado para realizar a simulação de 6 sensores referentes ao ambiente do silo. Para simular os sensores, foi necessário a geração de sinais de PWM. Como o Arduino UNO não possui conversores digitais-analógicos, optou-se pela utilização de um módulo de conversão, dessa forma, de acordo com a variação do duty cicle do PWM, é gerada uma tensão analógica equivalente. Para filtrar o sinal PWM e gerar o sinal analógico apropriado, foram implementados filtros passa-baixa no módulo de conversão.

Figura 3.7 – Sensoriamento.



Fonte: Elaboração própria.

A descrição detalhada do sensores está disposta na Tabela 3.2.

Tabela 3.2 – Descrição dos sensores.

Sensor	**Unidade**	**Funcionamento**
Pressão	kg/cm²	Determina a pressão interna do silo
Temperatura	ºC	Determina a temperatura no leito do grão
Concentração O2	Adimensional (Porcentagem)	Determina a concentração do oxigênio no silo
Concentração de Gás	Adimensional (Porcentagem)	Determina a concentração do gás sulfídrico no silo
Umidade	Adimensional (Porcentagem)	Determina a umidade relativa do silo
Concentração de poeira	g/m³	Determina a concentração da nuvem de poeira

Fonte: Elaboração própria.

Na Tabela 3.3 estão dispostas as faixas de operação dos sensores, esses valores foram escolhidos baseados nos valores expostos na Tabela 2.2 da sessão de Explosões.

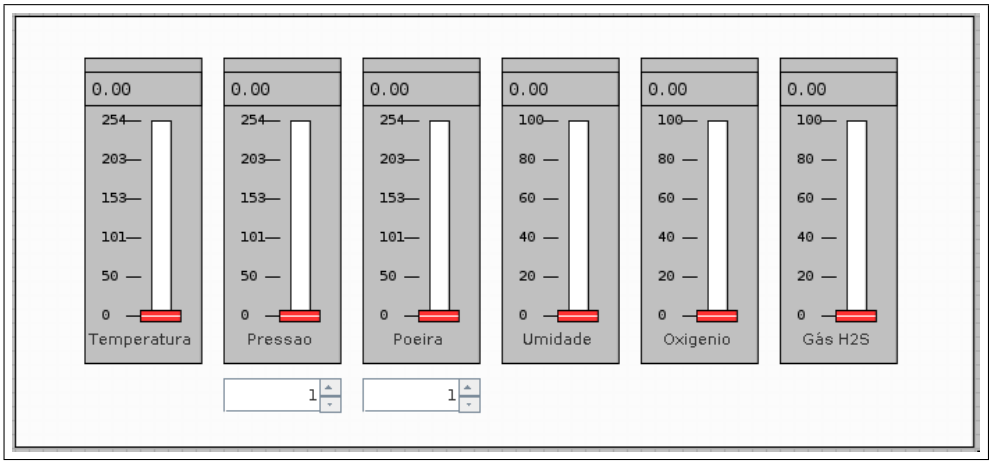
Tabela 3.3 – Funcionamento dos sensores.

Sensor	Faixa de operação
Pressão	0-20
Temperatura	0-70
Concentração O2	0-100
Concentração de Gás	0-100
Umidade	0-100
Concentração de poeira	0-100

Fonte: Elaboração própria.

No MyOpenLab inicialmente foi criado o painel frontal que contêm os controles de cada um dos 6 sensores como registra a Figura 3.8.

Figura 3.8 – MyOpenLab - Painel frontal.



Fonte: Elaboração própria.

Os controles deslizantes controlam o *Duty Cycle* das portas digitais do UNO e seus valores variam de 0 a 255. Os sensores de umidade, oxigênio e gás no sistema possuem uma limitação e ela se dá pelo fato de que essas características variam de 0 a 100% de intensidade, desse modo seus controles podem estar de 0 a 100, no caso dos demais sensores 100% do *Duty Cycle* foi utilizado.

Para o sensor de temperatura o regime está de 0 a 70 ºC, logo a resolução será de aproximadamente 0,2745 ºC para cada passo dado no controle deslizante, conforme demonstra a Equação (1).

$$Res = \frac{0-70}{0-255} = 0{,}2745 \tag{1}$$

O componente localizado abaixo dos controles possui a funcionalidade de dividir a escala de operação do sensor o que confere ao sistema mais sensibilidade diminuindo a resolução do mesmo.

Para o sensor de pressão o regime está de 0 a 20 kg/cm², logo a resolução será de aproximadamente 0,08 kg/cm² para cada passo dado no controle deslizante, conforme demonstra a Equação (2).

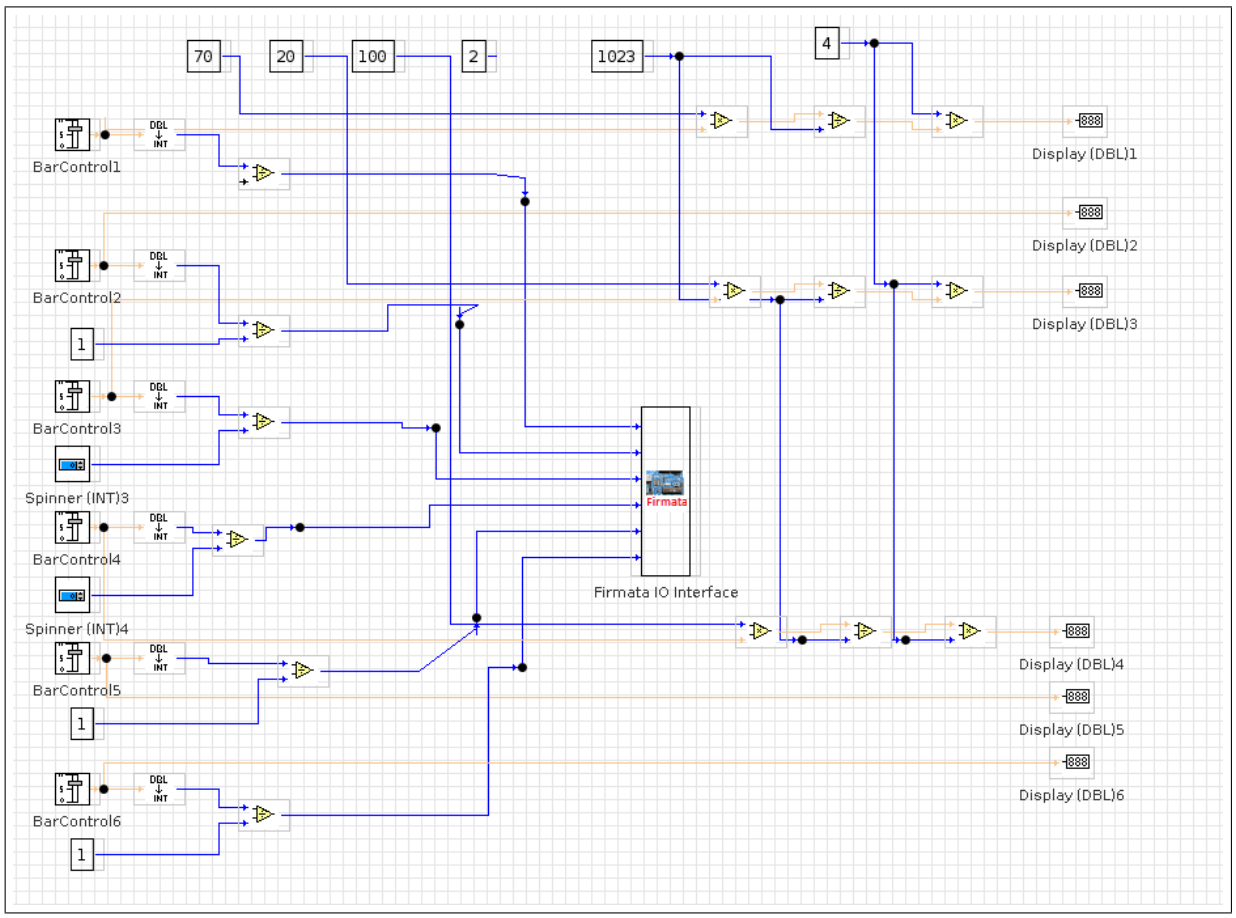
$$Res = \frac{0-20}{0-255} = 0{,}0784 \tag{2}$$

Para o sensor de pressão o regime está de 0 a 100 g/m³, logo a resolução será de aproximadamente 0,4 g/m³ para cada passo dado no controle deslizante, conforme demonstra a Equação (3).

$$Res = \frac{0-100}{0-255} = 0{,}3922 \tag{3}$$

Por fim a lógica foi implantada para o painel de circuito do MyOpenLab conforme demostra a Figura 3.9.

Figura 3.9 – MyOpenLab - Circuito.



Fonte: Elaboração própria.

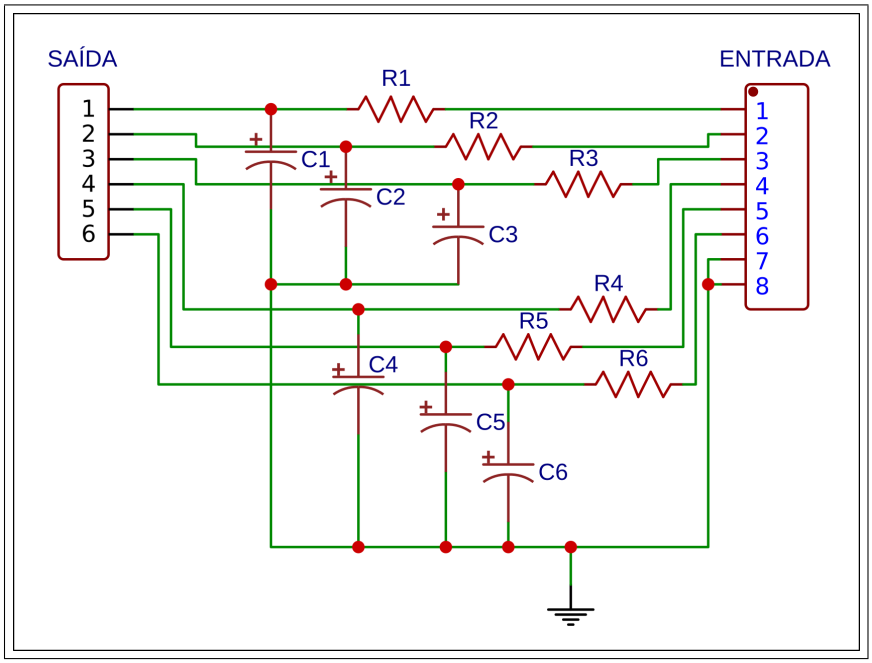
3.5 Sistema Embarcado

No projeto o MEGA foi responsável por embarcar a solução que envia pela rede os sinais gerados pelo UNO em conjunto com o MyOpenLab conforme demonstra a Figura 3.6.

Para interpretar os sinais produzidos na etapa de sensoriamento se fez necessária a utilização de filtros passa-baixa que filtram o sinal *PWM* para um nível de tensão equivalente funcionando como um *Digital-to-Analog Converter* (Conversor Digital Analógico) (DAC).

O DAC foi construído em cada um dos canais utilizando um resistor R em série com um capacitor C, onde a tensão de saída V_{out} está em paralelo com o capacitor C conforme demonstra a Figura 3.10.

Figura 3.10 – Conversor Digital Analógico.



Fonte: Elaboração própria.

Logo a equação referente ao cálculo da resistência e do capacitor é expressa na Equação (4).

$$f_c = \frac{1}{2\pi RC} \tag{4}$$

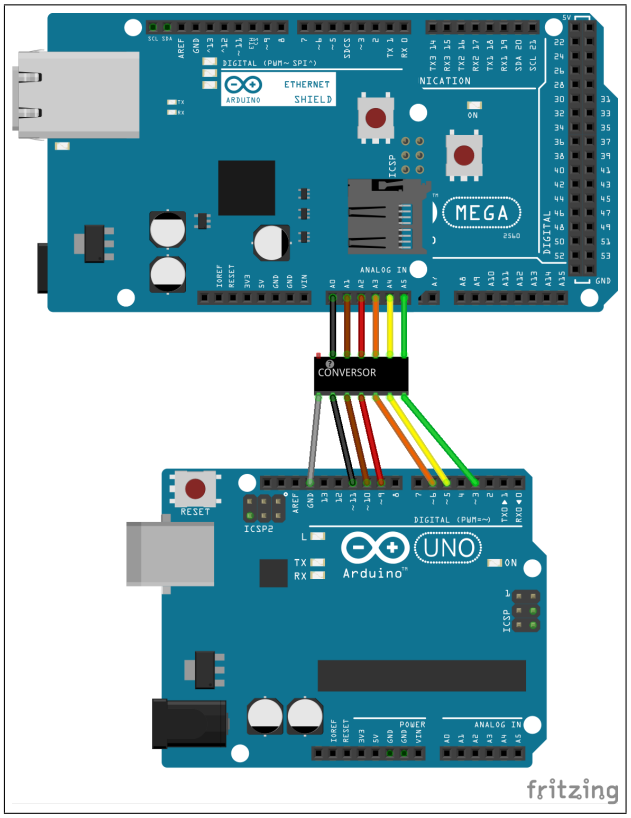
Por se tratar de um filtro passa baixa é recomendado que a frequência de corte escolhida seja menor que a frequência de funcionamento do sinal de *PWM*, logo foi considerado a frequência de corte $f_c = 200H_z$ e a capacitância $C = 1\mu F$ encontrando a resistência $R = 795{,}77\Omega$, por consequência foi considerado o valor de $R = 1k\Omega$ por se tratar de um valor comercial.

Com os valores de resistência e capacitância o cálculo da constante de tempo τ pode ser efetuado, está constante de tempo evidência o tempo dado em segundos que o capacitor demora até atingir aproximadamente 63% da tensão CC aplicada. O cálculo está expressado na Equação (5).

$$\tau = RC = 1m\ s \tag{5}$$

Com o conversor os sinais podem ser interpretados pelo MEGA, o esquemático de ligações disposto na Figura 3.11 exibe as conexões realizadas através do uso de segmentos condutores conectados e o DAC construído.

Figura 3.11 – Esquemático ligações.



Fonte: Elaboração própria.

Os conectores verde, amarelo, laranja, vermelho, marrom, e preto possuem respectivamente os sinais de temperatura, umidade, pressão, concentração de poeira, nível de oxigênio e nível de gás, o conector cinza representa o aterramento do circuito.

O Embarcado possui o seguinte funcionamento:

- **1º:** O sistema inicializa e se registra na rede através do *Dynamic Host Configuration Protocol* (Protocolo de Configuração Dinâmica de Host) (DHCP).
- **2º:** Com o MEGA registrado na rede o sistema passa a mapear os valores adquiridos nas entradas analógicas.
- **3º:** As informações mapeadas são disponibilizadas para a API, para manter integridade relacional do banco de dados o embarcado conta com um $codSerie$ que é um hash seguro (SHA512-256) de um código de série que identifica o silo no qual o MEGA atua.

Inicialmente a função de *setup* presente no Algorítimo 1 é executada e nela o sistema embarcado realiza o registro na rede, no caso de falha o MEGA não executa as demais funções e somente executa o laço *while(true)* o que requer que o microcontrolador seja reiniciado, dessa

forma foi garantido que o sistema somente executa as funções referentes ao mapeamento e o envio dos dados se o registro na rede for atestado pelo DHCP.

Algoritmo 1: Setup Arduíno.

```
1  void setup() {
2    // Inicia a Serial
3    Serial.begin(57600);
4    Serial.println("
       ============================================================");
5    Serial.println("WebClient Arduino");
6    Serial.println("Iniciando DHCP:");
7    if (Ethernet.begin(mac) == 0) {
8      Serial.println("Nao foi possivel registrar DHCP");
9      while (true) {
10       delay(1);
11     }
12   }
13
14   Serial.print("IP registrado: ");
15   Serial.println(Ethernet.localIP());
16   Serial.println("
       ============================================================\n");
17   // Delay para o Shield iniciar:
18   delay(1000);
19 }
```

O Algorítimo 2 demostra a implementação da função de loop do sistema, esta função é executada continuamente pelo MEGA. O Algorítimo conta com a definição de dois intervalos de execução o intervalo MILLS_AMBIENTE, referente a atualização do das características internas do silo e intervalo MILLS_WEBSERVICE, referente ao de envio das características mapeadas para a API consumir.

O intervalo de execução do MILLS_AMBIENTE é de 30 milissegundos já o intervalo de execução MILLS_WEBSERVICE é de 10 segundos, esta estratégia foi tomada para facilitar a implementação de novas funções no sistema embarcado como por exemplo a utilização de um cartão micro-SD aliado ao *slot* micro SD presente no *shield* de rede para armazenar o histórico de leituras realizadas pelo MEGA.

Algoritmo 2: Loop Arduíno.

```
#define MILLS_AMBIENTE 30
#define MILLS_WEBSERVICE 10000

void loop() {
  unsigned long currentMillisWebClient = millis();
  if (currentMillisWebClient - antWebClientMillis > MILLS_WEBSERVICE)
  {
    antWebClientMillis = currentMillisWebClient;
    webClient();
  }

  unsigned long currentMillisAmbiente = millis();
  if (currentMillisAmbiente - antAmbienteMillis > MILLS_AMBIENTE)
  {
    antAmbienteMillis = currentMillisAmbiente;
    setAmbienteSimulado();
  }
}
```

Com isso as condições do silo puderam ser mapeadas e conforme demonstra o Algorítimo 3 inicialmente é realizada a leitura das portas analógicas referentes aos sensores, com o valor da leitura analógica é realizada a conversão dos valores analógicos para a representação real do sensores respeitando os intervalos em que cada uma das características opera.

Algoritmo 3: Mapeando características.

```
void setAmbienteSimulado() {
  // Atualiza os valores dos sensores
  atualTemp = analogRead(A5);
  atualUmi = analogRead(A4);
  atualPre = analogRead(A3);
  atualPoeira = analogRead(A2);
  atualOxi = analogRead(A1);
  atualGas = analogRead(A0);

  // Mapeia o ambiente respeitando as faixas de operacao
  atualTemp = map(atualTemp, 0, 1023, 0, 70);
  atualUmi = map(atualUmi, 0, 409, 0, 100);
  atualPre = atualPre * 20 / 1023;
  atualPoeira = atualPoeira * 100 / 1023;
  atualOxi = map(atualOxi, 0, 409, 0, 100);
  atualGas = map(atualGas, 0, 409, 0, 100);
  atualGas = (atualGas / 100) * 4.3;
}
```

As funções responsáveis por se conectar no servidor e enviar as informações mapeadas estão demostradas no Algorítimo 4 e conforme registra o algorítimo, o MEGA estabelece a conexão com o servidor, mapeia as informações sobre a leitura atual e o CODSERIE responsável por identificar o MEGA, constrói e envia o JSON para o servidor consumir.

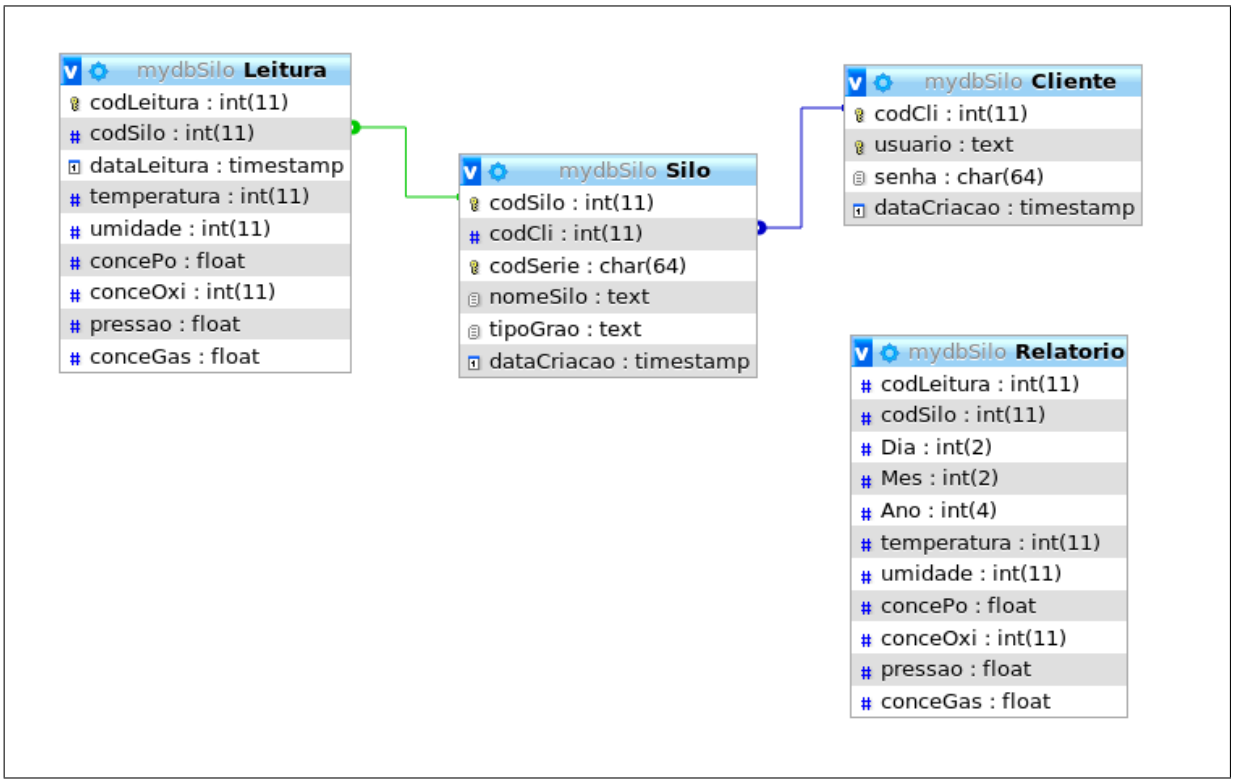
Algoritmo 4: Conexão com o servidor.

```
#define CODSERIE
"3e23e8160039594a33894f6564e1b1348bbd7a0088d42c4acb73eeaed59c009d"

void webClient() {
  // Implementa o WebClient verificando se consegue
  // se conectar no servidor
  client.stop();
  if (client.connect(servidor, 5001)) {
    printJsonAmbiente();
    // Espera um tempo
    delay(1);
  }
}

void printJsonAmbiente() {
  // Envia os dados para a API consumir
  StaticJsonDocument<250> doc;
  doc["pressao"] = atualPre;
  doc["temperatura"] = atualTemp;
  doc["conceOxi"] = atualOxi;
  doc["conceGas"] = atualGas;
  doc["umidade"] = atualUmi;
  doc["concePo"] = atualPoeira;
  doc["codSerie"] = CODSERIE;
  client.println(F("POST /api/ambiente HTTP/1.1"));
  client.println(F("Host: 192.168.16.4:5001"));
  client.println(F("Content-Type: application/json"));
  client.println(F("User-Agent: arduino-ethernet"));
  client.println(F("Accept: */*"));
  client.println(F("Accept-Encoding: gzip, deflate"));
  client.println(F("Connection: keep-alive"));
  client.print(F("Content-Length: "));
  client.println(measureJsonPretty(doc));
  client.println();
  delay(1);
  serializeJsonPretty(doc, client);
}
```

3.6 Banco de Dados

O banco foi construído utilizando o MySQL e possui quatro tabelas sendo três tabelas físicas e uma tabela virtual. O diagrama entidade relacionamento (DER) disposto na Figura 3.12 demonstra a construção do banco de dados utilizado no projeto.

Figura 3.12 – DER.



Fonte: Elaboração própria.

A Tabela 3.4 demonstra a construção da tabela Cliente, esta tabela foi pensada para registrar os clientes que acessam o sistema e é baseado nesta tabela que o controle de acesso é realizado.

Tabela 3.4 – Descrição tabela Cliente.

Coluna	Tipo	Nulo	Padrão	Comentários
codCli (Primária)	int(11)	Não		Chave primária de identificação da tabela
usuario	text	Não		Usuário do Cliente
senha	char(64)	Não		Senha do Cliente
dataCriacao	timestamp	Não	current_timestamp()	Data de criação do Cliente

Fonte: Elaboração própria.

A Tabela 3.5 demonstra a construção da tabela Leitura e esta tabela foi pensada para registrar os as características lidas do silo simulado.

Tabela 3.5 – Descrição tabela Leitura.

Coluna	Tipo	Nulo	Padrão	Comentários
codLeitura (Primária)	int(11)	Não		Chave primária de identificação da tabela
codSilo	int(11)	Não		Chave estrangeira do Silo
dataLeitura	timestamp	Não	current_timestamp()	Data em que a leitura foi salva no banco
temperatura	int(11)	Não		Temperatura no instante da leitura
umidade	int(11)	Não		Umidade no instante da leitura
concePo	float	Não		Concentração da poeira no instante da leitura
conceOxi	int(11)	Não		Nível da concentração de oxigênio no instante da leitura
pressao	float	Não		Pressão no instante da leitura
conceGas	float	Não		Concentração de Gás H2S no instante da leitura

Fonte: Elaboração própria.

A Tabela 3.6 demonstra a construção da tabela Silo e esta tabela foi pensada para registrar os silos presentes no sistema.

Tabela 3.6 – Descrição tabela Silo.

Coluna	Tipo	Nulo	Padrão	Comentários
codSilo (Primária)	int(11)	Não		Chave primária de identificação da tabela
codCli	int(11)	Não		Chave estrangeira do Cliente
codSerie	char(64)	Não		Identificação do arduíno
nomeSilo	text	Não		Nome do Silo
tipoGrao	text	Não		Tipo do grão ensilado
dataCriacao	timestamp	Não	current_timestamp()	Data da criação do Silo

Fonte: Elaboração própria.

Por fim a Tabela 3.7 demonstra a definição da tabela virtual Relatorio, está *View* é construída com base na tabela Leitura separando o atributo dataLeitura em Dia, Mes e Ano e esta estratégia foi adotada para facilitar o processamento dos dados afim de gerar os relatórios.

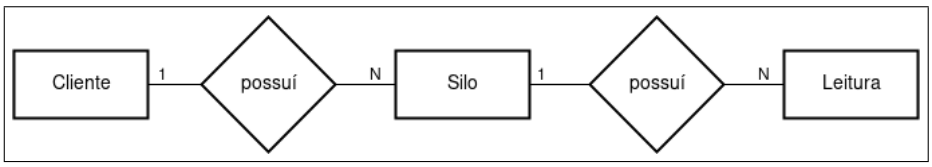
Tabela 3.7 – Descrição tabela Relatorio.

Coluna	Tipo	Nulo	Padrão	Comentários
codLeitura	int(11)	Não	-	Chave primária de identificação da tabela
codSilo	int(11)	Não		Chave estrangeira do Silo
Dia	int(2)	Sim	Dia da Leitura	
Mes	int(2)	Sim	Mês da Leitura	
Ano	int(4)	Sim	Ano da Leitura	
temperatura	int(11)	Não		Temperatura no instante da leitura
umidade	int(11)	Não		Umidade no instante da leitura
concePo	float	Não		Concentração da poeira no instante da leitura
conceOxi	int(11)	Não		Nível da concentração de oxigênio no instante da leitura
pressao	float	Não		Pressão no instante da leitura
conceGas	float	Não		Concentração de Gás H2S no instante da leitura

Fonte: Elaboração própria.

O modelo entidade relacionamento (MER) disposto na Figura 3.13 demonstra os relacionamentos entre as tabelas físicas do banco. Um Cliente possui muitos Silos enquanto um Silo pertence a um Cliente[4] e um Silo possui muitas Leituras enquanto uma Leitura pertence exclusivamente a um Silo.

Figura 3.13 – MER.



Fonte: Elaboração própria.

Para auxiliar no tratamento das informações do banco de dados foi desenvolvido dois gatilhos. A descrição dos gatilhos se deu por:

- **ExcluiLeituras:** Gatilho ativado antes de excluir um silo e tem por função excluir todas as leituras relacionadas ao silo.
- **ExcluiSilo:** Gatilho ativado antes de excluir um cliente e tem por função excluir o silo relacionado ao cliente.

Também foram criados *procedures* para auxiliar nas buscas necessárias para a utilização do banco de dados, a Tabela 3.8 descreve cada um dos *procedures* utilizados.

[4]Este relacionamento foi pensado visando a expansão do projeto tendo em vista que o aplicativo cria o silo no instante de criação do cliente e não possui uma alternativa para a criação do silo separadamente.

Tabela 3.8 – *Procedures* utilizados.

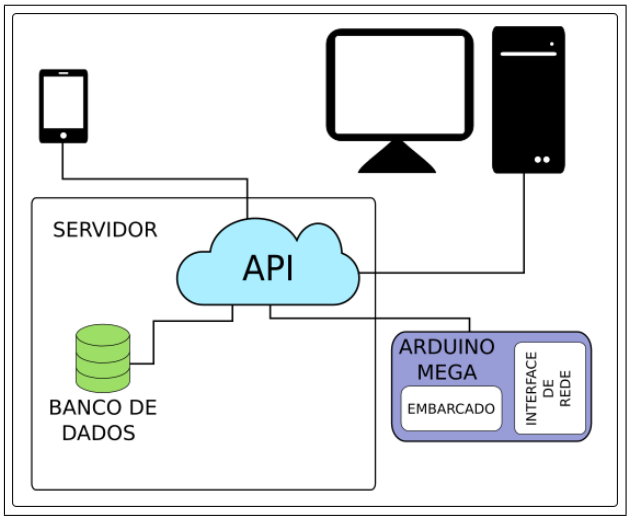
Procedure	**Parâmetros**	**Retorno**	**Descriçao**
BuscaCodCli	usuario, senha	codCli	Busca o código do cliente
BuscaLeitura	codSilo	Leitura	Busca a última leitura do silo
BuscaSilo	codCli	Silo	Busca o silo do cliente
Login	usuario, senha	codSilo, codCli, codSerie, nomeSilo, tipoGrao	Busca as informações de cliente e silo
RelatorioDia	codSilo	Relatorio	Busca na *view* Relatorio e estrutura em função da média diária
RelatorioMes	codSilo	Relatorio	Busca na *view* Relatorio e estrutura em função da média mensal
ValidaLeitura	codSerie	codSilo	Busca o Silo que possui o codSerie

Fonte: Elaboração própria.

3.7 API

A API foi desenvolvida utilizando Python em conjunto do Flask que é um *framework* voltado para o desenvolvimento de soluções web e do SQLAlchemy que é uma biblioteca que facilita a integração do banco de dados com o Python. A API foi criada com o intuito de armazenar as informações referentes ao funcionamento tanto do silo como da interface de acesso do cliente criando a possibilidade do acesso e monitoramento independente da rede que esteja sendo utilizada requerendo apenas que o servidor que esteja hospedando a API seja acessível pela rede conforme demonstra a Figura 3.14.

Figura 3.14 – Arquitetura *Web*.



Fonte: Elaboração própria.

A API possui 8 classes construídas utilizando a arquitetura REST conforme demonstra a Tabela 3.9.

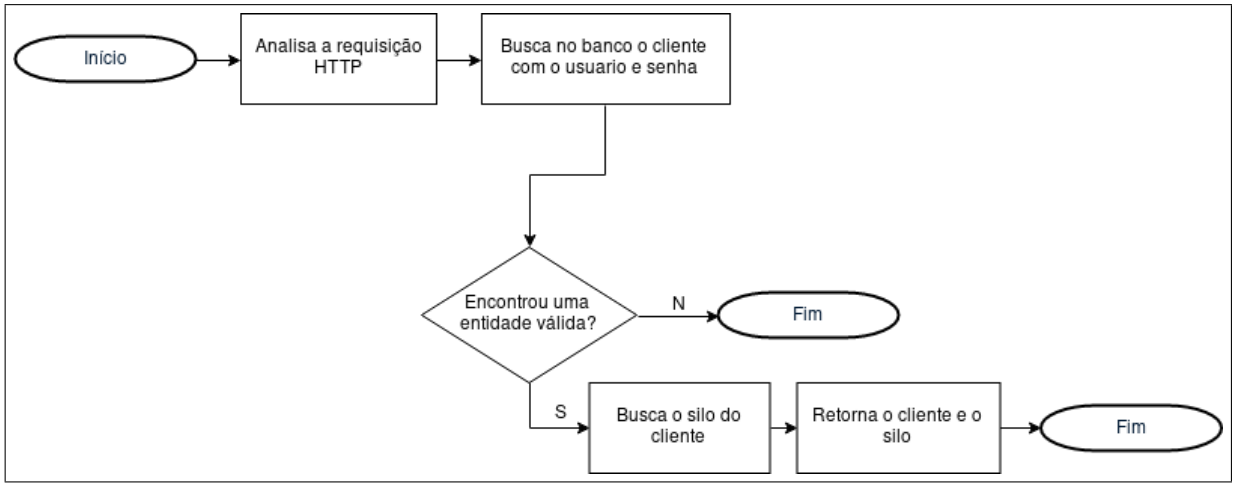
Tabela 3.9 – Classes API.

Classe	**Caminho**	**Métodos**	**Descrição**
Login	/api/login	Get	Controla o acesso ao sistema
Ambiente	/api/ambiente	Get, Post	Trabalha com os dados do ambiente interno do silo simulado
Cliente	/api/cliente/	Post	Cria o cliente e o silo do cliente
SenhaCliente	/api/cliente/senha/	Post	Troca a senha do cliente
DeletaCliente	/api/cliente/deleta/	Post	Exclui o cliente e seus dados
NomeSilo	/api/silo/nomesilo/	Post	Troca o nome do silo
TipoGrao	/api/silo/tipograo/	Post	Troca o tipo do grão ensilado
Relatorio	/api/relatorio	Get	Gera o relatório do silo

Fonte: Elaboração própria.

A classe Login controla o acesso ao sistema verificando se o usuário e a senha existem no banco, se existir retorna a entidade do Cliente e a entidade do Silo que o cliente possui conforme demostra o fluxograma exposto na Figura 3.15.

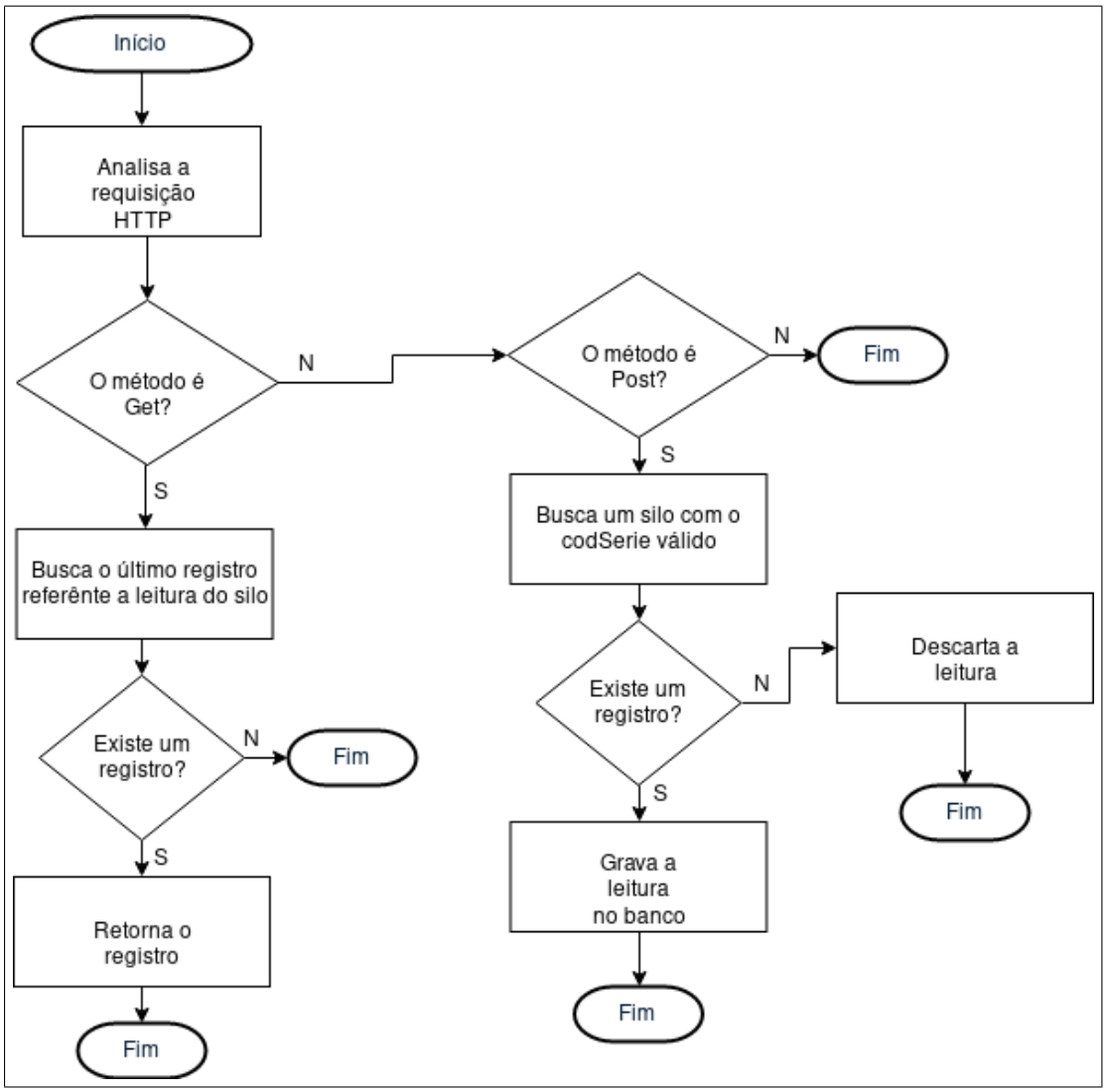
Figura 3.15 – Fluxograma da classe Login.



Fonte: Elaboração própria.

A classe Ambiente responsável por popular a tabela Leitura e por retornar a leitura mais recente do silo funciona conforme demostra o fluxograma da Figura 3.16.

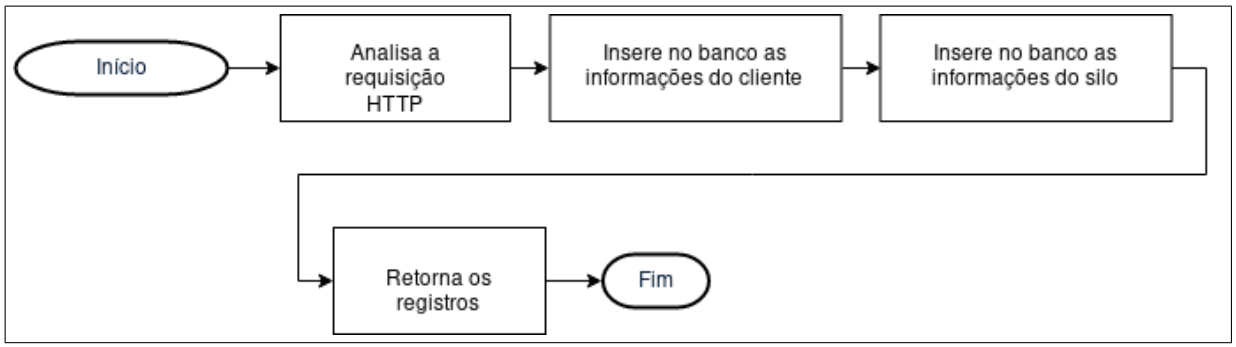
Figura 3.16 – Fluxograma da classe Ambiente.



Fonte: Elaboração própria.

A classe Cliente realiza o cadastro do cliente e do silo do cliente, inicialmente uma requisição para criação do cliente é gerada, o banco verifica se não existe um cliente com o usuario do cliente que se deseja cadastrar bem como um silo com o codSerie, se não existir o cliente é cadastrado. Para cadastrar o silo é realizada uma busca referente ao cliente que acabou de ser cadastrado que retorna o codCli do mesmo e com isso o silo pode ser cadastrado. Feito isto a classe retorna para a aplicação as entidades estruturadas, este fluxo pode ser observado na Figura 3.17.

Figura 3.17 – Fluxograma da classe Cliente.



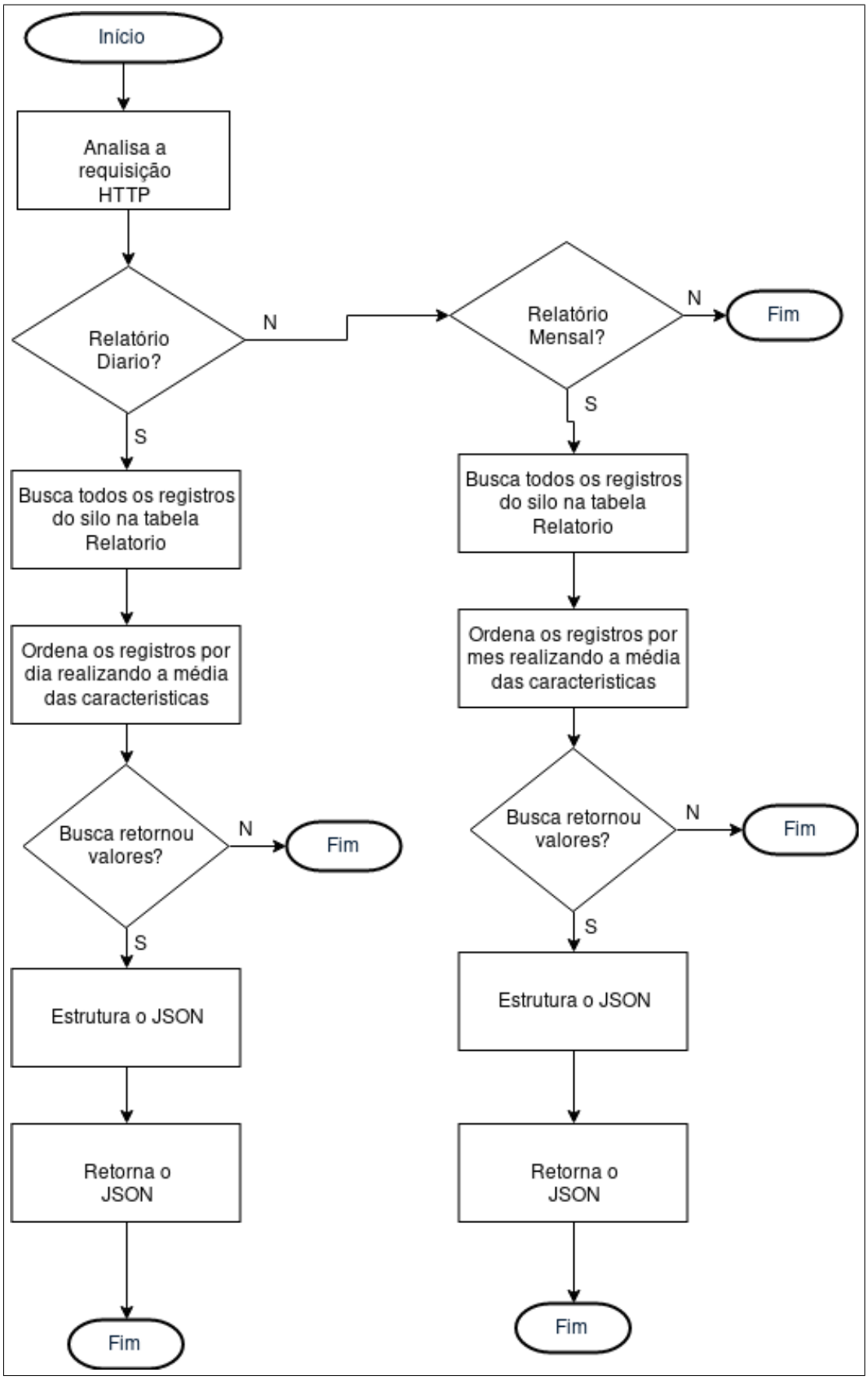
Fonte: Elaboração própria.

As classes SenhaCliente, NomeSilo e TipoGrao são classes voltadas para a alteração de características relacionadas ao Silo (NomeSilo e TipoGrao) ou ao Cliente (SenhaCliente).

Por fim a classe Relatorio gera os relatórios a respeito das condições internas do silo, existem duas opções de relatório o relatório diário e o relatório mensal, no relatório diário as informações são organizadas por dia através da média de cada uma das características mapeadas e para o relatório mensal o mesmo ocorre porém levando em consideração o mês, em ambos os casos a quantidade de leituras realizadas é informada. Conforme a Figura 3.18 demonstra o fluxo se dá da seguinte forma:

- **1º:** Ocorre a requisição.
- **2º:** A requisição é processada a fim de especificar qual o tipo de relatório requisitado.
- **3º:** O banco é consultado e retorna o conjunto de dados equivalente a especificação.
- **4º:** Os dados processados são retornados para o host.

Figura 3.18 – Fluxograma da classe Relatorio.



Fonte: Elaboração própria.

3.8 Aplicativo

O aplicativo foi desenvolvido com o Ionic e possui a função de ser um painel central gráfico onde o cliente pode cadastrar as informações a respeito do silo, acompanhar em tempo real as condições internas do silo e visualizar os relatórios.

Para o desenvolvimento da interface gráfica foi utilizado o projeto de exemplo disponibilizado pelo Ionic, o *Conference APP*. Foram adaptadas e criadas 7 *pages*, e conforme demonstra a Tabela 3.10 cada uma das *pages* possui uma funcionalidade para o sistema.

Tabela 3.10 – Descrição das *pages* do aplicativo.

Page	**Descrição**
about	Possui detalhes sobre as características e o funcionamento do aplicativo
account	Gerência as características do silo e cliente
login	Controle de acesso do cliente
report	Requere e exibe os relatórios sobre o ambiente interno do silo
schedule	Página principal do sistema que exibe o monitoramento em tempo real
singup	Criação do cliente e do silo
tutorial	Página com um tutorial de utilização do sistema

Fonte: Elaboração própria.

Foram utilizados três *providers* e essas classes possuem a funcionalidade de controlar cada uma das partes da interface conforme registra a Tabela 3.11. Cada um dos *providers* serão descritos em maior detalhe no decorrer do capítulo.

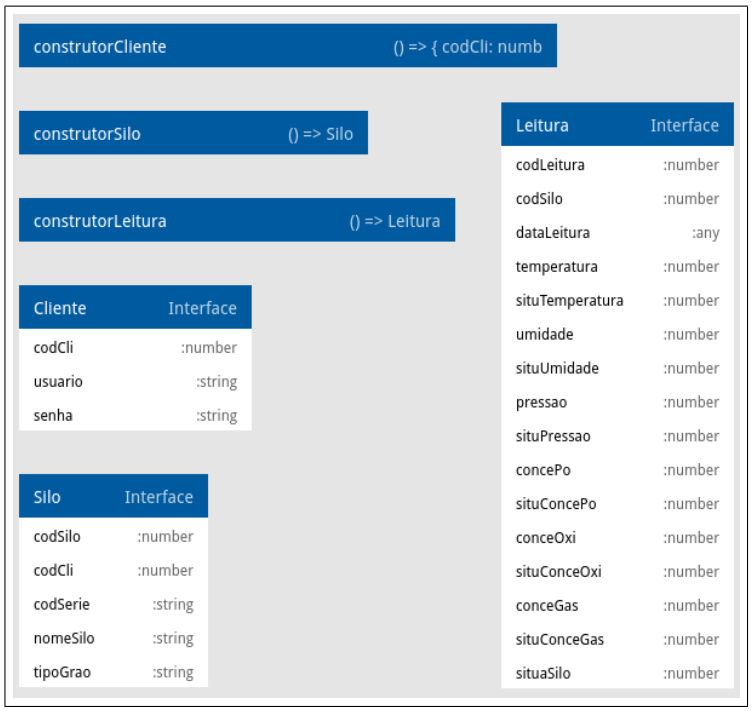
Tabela 3.11 – Descrição dos *providers*.

Provider	**Descrição**
check-tutorial	Verifica no momento de iniciação do aplicativo se o tutorial já foi realizado no dispositivo e se já foi redireciona para a página principal
db-data	Controla os métodos de acesso a API
user-data	Controla os métodos de uso geral da interface

Fonte: Elaboração própria.

E por fim o projeto possui uma classe *user-options* que implementa três interfaces e três funções de iniciação das mesmas conforme demonstra a Figura 3.19.

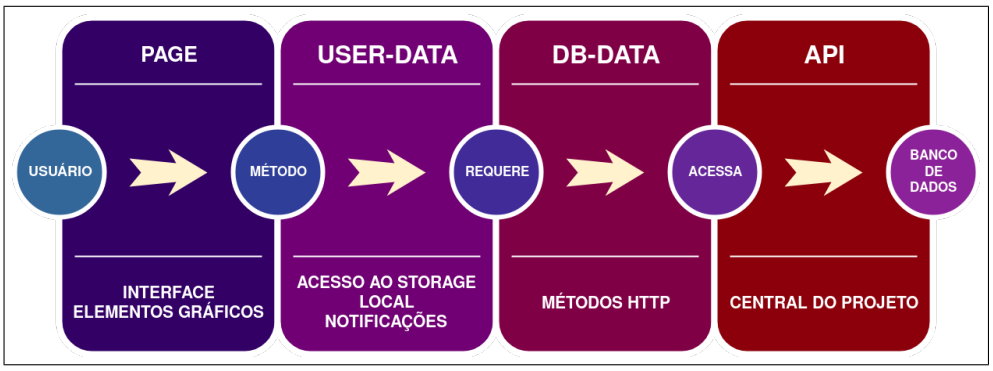
Figura 3.19 – Descrição *user-options*.



Fonte: Elaboração própria.

O fluxo comum de ações do aplicativo se dá com o usuário realizando alguma ação na *page*, esta ação desencadeia um método que utiliza de métodos implementados no *provider user-data*, alguns desses métodos requerem o acesso a API e este acesso é efetivamente realizado no *provider db-data* que implementa os métodos HTTP para realizar efetivamente o acesso a API e por fim a API realiza o que deve ser feito conforme demonstra a Figura 3.20.

Figura 3.20 – Fluxo do projeto visto pelo APP.



Fonte: Elaboração própria.

3.8.1 *user-data*

Este *provider* tem por função implementar métodos responsáveis por manipular os dados adquiridos e gerados durante a execução do sistema, manipulando eventos e o *storage*

local.

A Tabela 3.12 descreve o funcionamento básico do *provider*.

Tabela 3.12 – Descrição dos métodos do *user-data*.

Método	**Parâmetro**	**Descrição**	**Retorno**
zeraLeitura	-	Zera a interface Leitura	void
suporte	dados: any, opcao: string	Controla as ações da relacionadas a *page account* baseado no dado e na opção passados	void
login	usuario: string, senha: string	Controla o acesso do *login* através da chamada do retorno do *db-data*	void
signup	dados: any	Chama o método de *singup* do *db-data* passando os dados tratados para inserção no banco	void
logout	-	Desloga o cliente e limpa o *storage* local	void
setCliente	cliente: Cliente	Grava o cliente no *storage* local	void
getCliente	-	Retorna o cliente registrado no *storage* local	Cliente
setSilo	silo: Silo	Grava o silo no *storage* local	void
getSilo	-	Retorna o silo registrado no *storage* local	Silo
setLeitura	leitura: Leitura	Grava a leitura no *storage* local	void
getLeitura	-	Retorna a leitura registrada no *storage* local	Leitura
setTimer	data: any	Grava o timer no *storage* local	void
getTimer	-	Retorna o timer registrado no *storage* local	any
setChart	chart: any	Grava os dados do gráfico no *storage* local	void
getChart	-	Retorna os dados do gráfico registrados no *storage* local	any
isLoggedIn	-	Retorna o estado do *login*(*true* ou *false*)	boolean
checkHasSeenTutorial	-	Retorna o estado a respeito da execução do tutorial	void
getAmbi	atual: Silo	Atualiza a leitura atual	void
getRelatorio	codSilo: number, opcao: string	Busca os dados para realizar o relatório	void
setSituacao	atual: Silo, resposta: any	Atualiza a situação atual do silo baseado na leitura atual do silo	void

Fonte: Elaboração própria.

O método de *signup* recebe dados a respeito do Cliente e do Silo e realiza tratamentos relacionados a verificação da existência de um cliente ou um silo que possua o usuario ou o codSerie igual ao que se desejava cadastrar, o método também aplica uma função de hash SHA512-256 nos atributos senha e codSerie para prover maior segurança para o usuário.

O método setSitucao conforme abordado na Tabela 3.12 tem por funcionalidade classificar cada uma das características do silo.

Como abordado anteriormente para classificar o silo foi considerado que:

- O sistema considera que está sempre operando no pior caso em relação a energia de ignição, logo elimina também a influência da temperatura já que esta temperatura para o silo sempre estará operando uma ordem de grandeza abaixo da temperatura necessária para combustão.

Conforme abordado na Fundamentação Teórica para que ocorra uma condição favorável para a explosão da poeira deve ocorrer uma combinação em que cada uma das características fundamentais é satisfeita. Portanto para classificar a situação do silo foi realizado o cálculo da porcentagem em relação cada uma das variáveis normalizadas que representam o ambiente interno do silo, respeitando o tipo do grão armazenado.

A normalização ocorre da seguinte forma:

- **1º:** O tipo do grão do silo é mapeado.
- **2º:** Baseado no tipo do grão as características mínimas são mapeadas de acordo com a Tabela de Poeiras agrícolas.
- **3º:** Com as características mínimas mapeadas é realizado um cálculo de porcentagem envolvendo a leitura atual em relação a característica mínima para cada uma das características que representam o ambiente interno do silo.
- **4º:** Com isso as variáveis normalizadas para cada uma das características internas do silo são definidas.

Deste modo, a Equação (6) demostra o cálculo da situação do silo utilizando as variáveis normalizadas. Uma notificação é gerada caso o valor da situação do silo atinga 80% e possui um intervalo de 1 minuto para geração de uma nova notificação para o caso da situação se manter.

$$Silo_{(\%)} = \frac{Oxi_{(\%)} + Po_{(\%)} + Gas_{(\%)} + Pre_{(\%)} + Umi_{(\%)}}{5} \tag{6}$$

3.8.2 *db-data*

Este *provider* tem por função implementar métodos responsáveis por acessar a API, disponibilizando os dados e verificando os erros relacionados a rede.

A Tabela 3.13 descreve o funcionamento básico do *provider*.

Tabela 3.13 – Descrição dos métodos de *db-data*.

Método	**Parâmetro**	**Descrição**	**Retorno**
handleError	error: HttpErrorResponse	Mapeia o erro que ocorreu durante a execução de um método HTTP	void
postCliente	dados: any	Método voltado para o método Post no resource *cliente/*	void
postSenhaCliente	dados: any	Método voltado para o método Post no resource *cliente/senha/*	void
postNomeSilo	dados: any	Método voltado para o método Post no resource *silo/nomesilo/*	void
postTipoGrao	dados: any	Método voltado para o método Post no resource *silo/tipograo/*	void
postDeletaCliente	dados: any	Método voltado para o método Post no resource *cliente/deleta/*	void
getLogin	usuario: string, senha: string	Método voltado para o método Get no resource *login* passando usuario e senha como parâmetros da *query*	void
getAmbi	atual: Silo	Método voltado para o método Get no resource *ambiente* passando codSilo como parâmetro da *query*	void
getRelatorio	codSilo: number, opcao: string	Método voltado para o método Get no resource *relatorio* passando codSilo e opcao como parâmetros da *query*	void

Fonte: Elaboração própria.

3.8.3 *about*

Está página possui como única funcionalidade disponibilizar informações a respeito das características que requeridas na *page* de *singup* além de apresentar um panorama geral do projeto, portanto não se fez necessário a utilização de métodos ou funções e apenas foi utilizado marcações em HTML.

3.8.4 *account*

Esta *page* gerência a conta do cliente e o silo do cliente e possui funções para alteração e exclusão conforme demostra a Tabela 3.14.

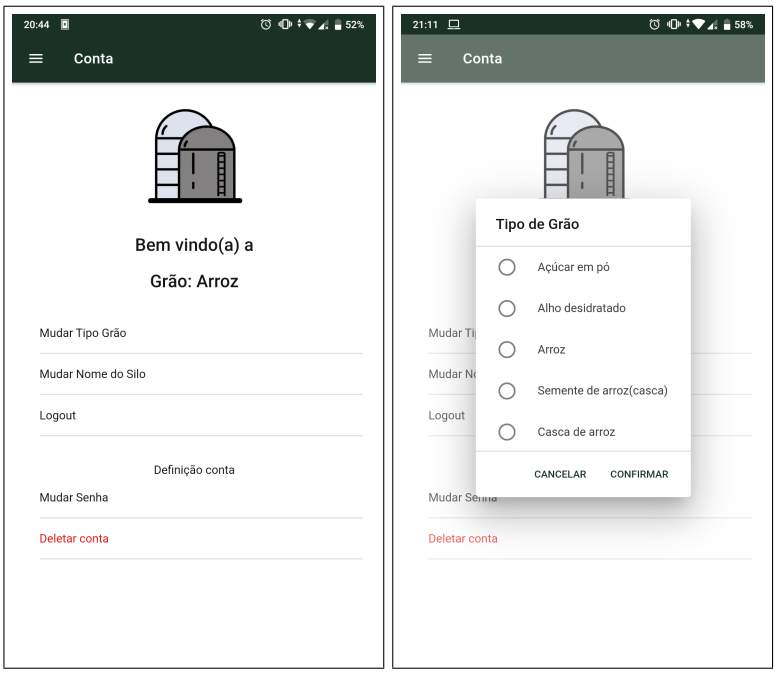
Tabela 3.14 – Descrição dos métodos de *account*.

Método	**Parâmetro**	**Descrição**	**Retorno**
ngAfterViewInit	-	Atualiza os dados da interface através do método *getCliente*	void
changeNomeSilo	-	Exibe uma caixa para inserção de um novo nome do Silo e atualiza na API	void
changeSenha	-	Exibe uma caixa para inserção de uma nova senha do Cliente e atualiza na API	void
changeTipoGrao	-	Exibe todas as opções de grão e atualiza na API	void
getCliente	-	Atualiza os dados do Cliente e do Silo	void
logout	-	Desloga do sistema	void
support	-	Chama a função assíncrona de exclusão do Cliente	void
deletarConta	-	Função assíncrona de exclusão do Cliente	void

Fonte: Elaboração própria.

A demonstração do visual da *page* pode ser vista na Figura 3.21, a figura da esquerda demostra o painel principal e nele é exibido o usuário com o tipo do grão ensilado e 5 botões que são voltados para alterações nos dados conforme seus textos indicam.

Figura 3.21 – Visão do *account*.



Fonte: Elaboração própria.

3.8.5 *login*

Esta *page* controla o acesso do usuário ao sistema, na Tabela 3.15 está descrito o funcionamento da mesma.

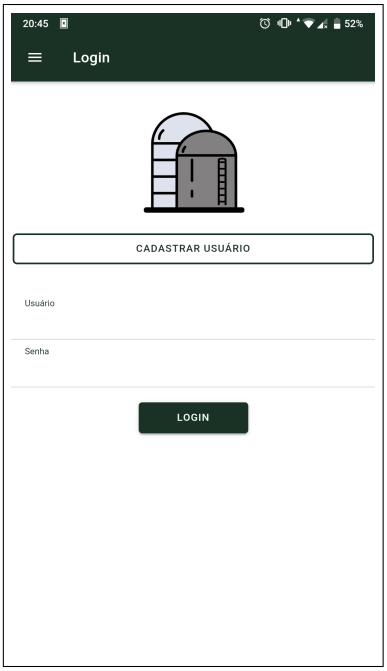
Tabela 3.15 – Descrição dos métodos de *login*.

Método	**Parâmetro**	**Descrição**	**Retorno**
onLogin	form: NgForm	Verifica o preenchimento do formulário e chama o método de login do *user-data*	void
onSignup	-	Redireciona para a página de criação do cliente e do silo	void

Fonte: Elaboração própria.

A interface gráfica disposta na Figura 3.22 demonstra os campos usuário e senha voltados para o login e o botão para criação do usuário que evoca o método *onSingup*.

Figura 3.22 – Visão do *login*.



Fonte: Elaboração própria.

3.8.6 *report*

Esta *page* exibe os relatórios referentes ao ambiente interno do silo simulado, na Tabela 3.16 se encontra a descrição dos métodos do *report*.

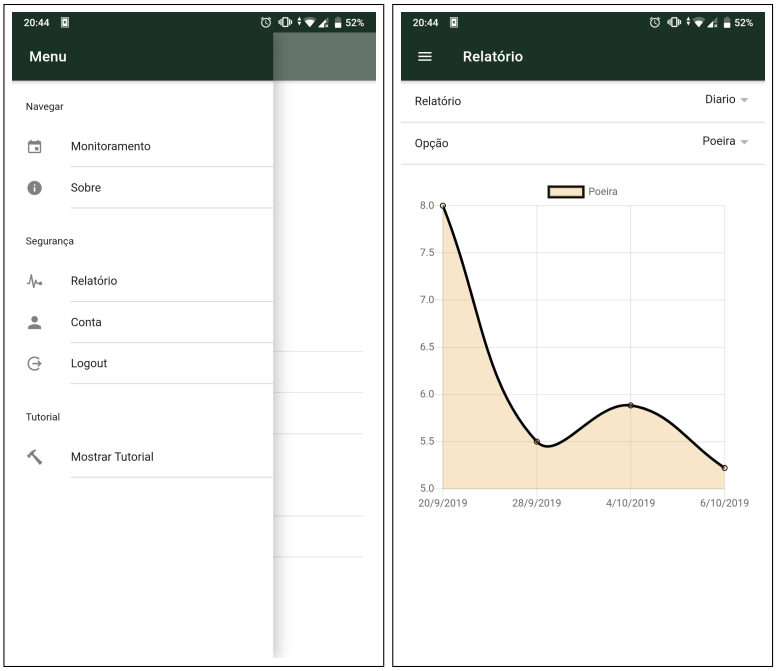
Tabela 3.16 – Descrição dos métodos de *report*.

Método	**Parâmetro**	**Descrição**	**Retorno**
ngOnInit	-	Inicializa os dados da interface e realiza uma consulta para um relatório diário	void
onChange	-	Realiza a nova consulta utilizando a opção de relatório escolhida	void
getChart	-	Atualiza o gráfico de acordo com as opções escolhidas	void

Fonte: Elaboração própria.

A interface gráfica exposta na Figura 3.23 conta com o visual do relatório.

Figura 3.23 – Visão do *report*.



Fonte: Elaboração própria.

3.8.7 *schedule*

Esta é a *page* principal do sistema, nela se encontram a descrição do silo monitorado e o monitoramento em tempo real, na Tabela 3.17 se encontra a descrição dos métodos utilizados.

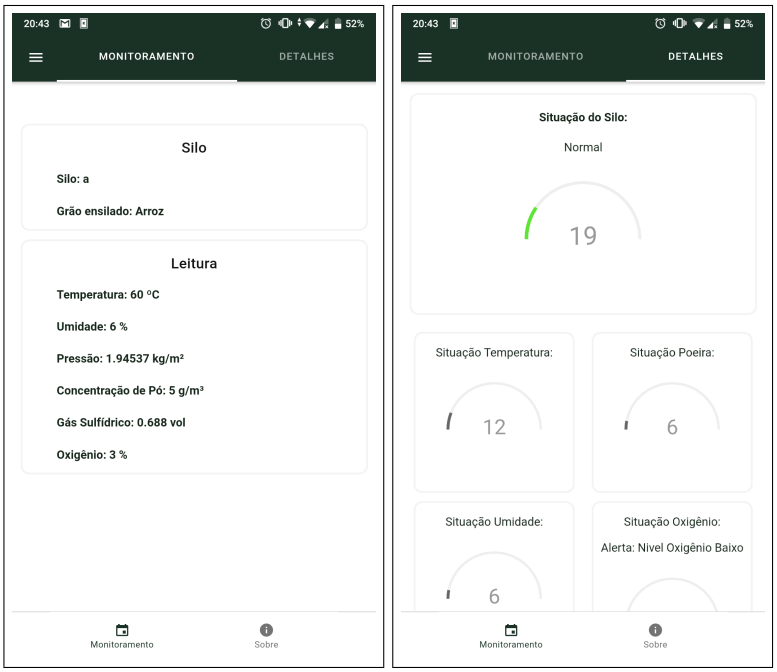
Tabela 3.17 – Descrição dos métodos de *schedule*.

Método	**Parâmetro**	**Descrição**	**Retorno**
color	value: number	Atualiza as cores do indicador da situação do silo	void
ngOnInit	-	Inicia as variáveis e define o intervalo de execução de 10 segundos do método *updateAmb*	void
segmentChanged	-	Controla o segmento atual da página	void
slideChanged	-	Controla a função de slide da página, voltada para a navegação entre os segmentos	void
ngOnDestroy	-	Destrói o intervalo de execução da função *updateAmb* para que não gere um funcionamento fatorial	void
updateAmb	-	Função assíncrona que atualiza os elementos gráficos baseado nos valores atuais de leitura	void

Fonte: Elaboração própria.

A interface gráfica exposta na Figura 3.24 retrata o painel principal do aplicativo e este painel possui dois seguimentos, o seguimento de Monitoramento e o seguimento de Detalhes, em Monitoramento são expostas informações à respeito do silo e da leitura atual e em Detalhes a classificação atual do sistema utilizando as variáveis normalizadas.

Figura 3.24 – Visão do *schedule*.



Fonte: Elaboração própria.

A função *color* responsável por atualizar a coloração do indicador a fim de apresentar uma resposta gráfica de fácil entendimento. A coloração é definida:

- **Verde:** Para valores de 0 a 39%.
- **Amarela:** Para valores de 40 a 55%.
- **Laranja:** Para valores de 56 a 69%.
- **Vermelho:** Para valores maiores ou igual a 70%.

A função *updateAmb* controla a atualização dos valores para realizar o monitoramento em tempo real, e para isso a função utiliza os métodos *isLoggedIn, getSilo, getAmbi, getLeitura* do *user-data*.

O funcionamento é definido da seguinte forma

- Verifica se o cliente está logado, se não está logado exibe os valores padrões.
- Se logado carrega os silo do cliente e a última leitura, se não houver leitura carrega o valor padrão.
- Se houver leitura exibe a leitura e gera os textos de detalhe a respeito da concentração de oxigênio, concentração de gás e situação do silo.

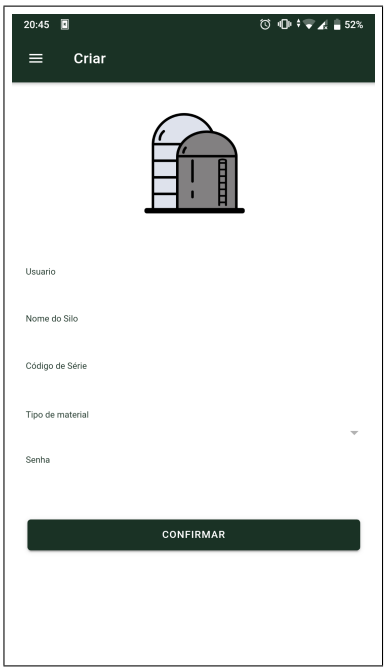
No seguimento de Detalhes são demonstradas as mensagens voltadas ao operador. As mensagens geradas baseadas na variáveis normalizadas são:

- **Situação da concentração de gás:** São alertas para 10% do LII atingido, para 25% de LII atingido, de nível crítico para 80% de LII atingido e de nível seguro.
- **Situação do silo:** Nível crítico para valores maiores ou igual a 70%, nível alerta para valores entre 40 e 70% e nível normal.
- **Situação do oxigênio:** Alerta para nível de oxigênio baixo para valores menores ou igual a 19%, alerta para nível de oxigênio alto para valores maiores que 23%, e de nível seguro.
- **Situação da concentração de poeira:** São alertas para 10% da concentração atingida, para 25% da concentração atingida, de nível crítico para 80% da concentração atingida e de nível seguro.

3.8.8 *singup*

Esta *page* é voltada para o cadastro do cliente e do silo, e só possui o método *onSingup(form: NgForm)* que verifica se o formulário foi preenchido e no caso de ter sido preenchido chama o método de cadastro do *provider user-data*.

Figura 3.25 – Visão do *singup*.

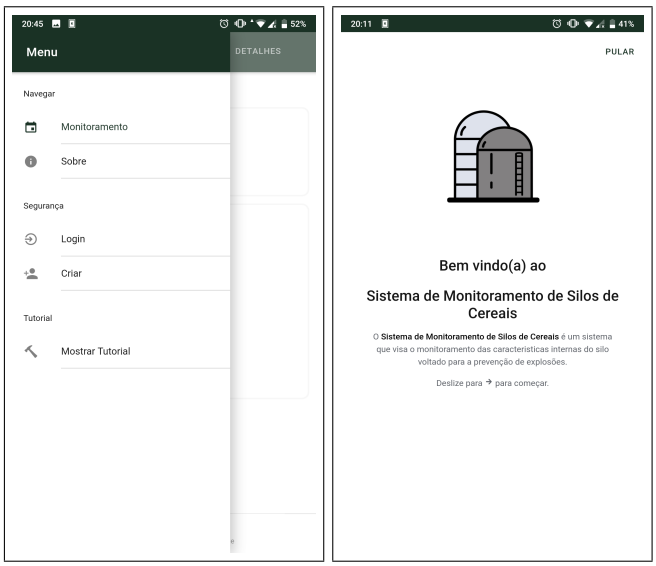


Fonte: Elaboração própria.

3.8.9 *tutorial*

Esta *page* é voltada para exibir um tutorial de utilização do sistema, ela é mostrado na primeira vez que o sistema é executado e também pode ser acessado outras vezes através do menu principal conforme demonstra a Figura 3.26 a esquerda, na Tabela 3.18 se encontra a descrição dos métodos utilizados e na Figura 3.26 a direita tela inicial do tutorial.

Figura 3.26 – Visão do *tutorial*.



Fonte: Elaboração própria.

Tabela 3.18 – Descrição dos métodos de *tutorial*.

Método	**Parâmetro**	**Descrição**	**Retorno**
startApp	-	Redireciona para a *page* principal do sistema (*schedule*)	void
onSlideChangeStart	-	Exibe o botão para pular o tutorial	void
ionViewWillEnter	-	Verifica se o tutorial foi realizado e se foi redireciona para a página principal	void
ionViewDidLeave	-	Ativa o menu	void

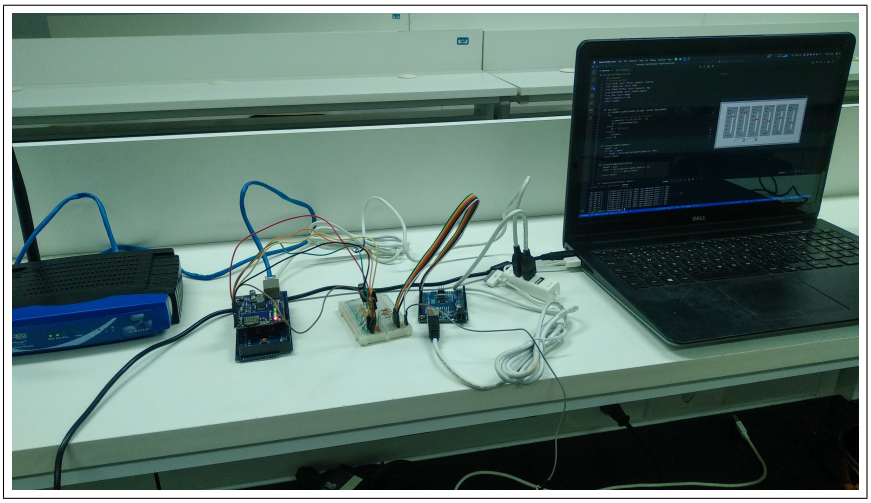
Fonte: Elaboração própria.

4 RESULTADOS

Neste capítulo serão abordados os desafios enfrentados no decorrer do desenvolvimento do projeto, tratando isoladamente cada uma das divisões do projeto e por fim os resultados obtidos.

Os testes foram realizados conforme demonstra a Figura 4.1 onde os dispositivos do projeto foram montados na bancada do laboratório.

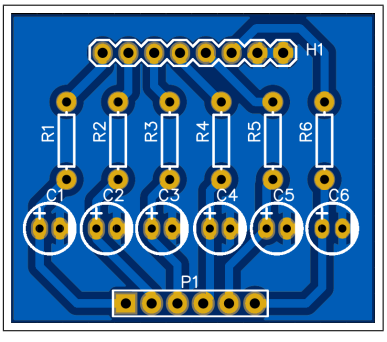
Figura 4.1 – Bancada.



Fonte: Elaboração própria.

O circuito do DAC foi implementado em uma *protoboard* para facilitar a troca de componentes caso fosse necessário. Uma *Printed circuit board* (Placa de circuito impresso) (PCB) foi desenhada para facilitar o transporte e diminuir a possibilidade de erros relacionados ao manuseio do circuito porém por questões relacionadas ao tempo ela não foi implementada. O modelo da PCB pode ser visto na Figura 4.2.

Figura 4.2 – Modelo PCB.



Fonte: Elaboração própria.

4.1 Sensoriamento

Nesta etapa do projeto conforme abordou a etapa do Sensoriamento foi realizada a simulação do sensores que são responsáveis por aferir o sistema embarcado. Uma das dificuldades enfrentadas nesta etapa foi a complexidade de se definir como os sensores se comportariam dado que este comportamento depende do grão escolhido. Para superar este obstáculo foi aliado aos controladores deslizantes o divisor de escala conforme explicado na sessão.

Um dos resultados se deu pela utilização do MyOpenLab que se mostrou eficiente controlando tanto o Arduíno UNO como o MEGA, diferente do que era esperado já que o MEGA conta com um processamento superior em relação ao UNO. O Arduíno UNO foi escolhido por ter um custo inferior que o Arduíno MEGA e ser suficiente para o uso com 6 sensores. Caso seja necessária a adição de mais sensores, o Arduíno MEGA deverá ser utilizado.

4.2 Sistema Embarcado

Nesta etapa do projeto conforme abordado na sesssão referente ao Sistema Embarcado foi realizada a implementação do sistema embarcado. As principais dificuldades enfrentadas nesta etapa estavam relacionadas a qual abordagem deveria ser adotada. Inicialmente foi implementado um *WebServer* que realizava as seguintes funções:

- Recebia o tipo de grão ensilado da aplicação.
- Classificava a situação do silo em relação a leitura atual do ambiente e o grão armazenado.
- Retornava o JSON com os dados classificados.

Porém esta abordagem se mostrava pouco eficaz dado que durante os testes as requisições ao *WebServer* demoravam em média dois segundos[1] em uma rede interna. Com os testes também foi detectado uma dificuldade em fazer o *WebServer* se comunicar com a a aplicação ativa em um celular, e esta dificuldade se dava pela política de *Cross-Origin Resource Sharing* (Compartilhamento de Recursos de Origem Cruzada) (CORS) que bloqueava as requisições, logo, uma outra abordagem teve de ser adotada e a abordagem escolhida foi a implementação de um *WebClient*, com isso, o sistema embarcado deixou de ser o elemento central e passou a ser um elemento passivo que possui somente as leituras conforme foi abordado na etapa do Sistema Embarcado.

Outro resultado se deu pelo desenvolvimento do conversor digital-analógico de 6 canais, que converte o sinal de saída PWM do Arduíno em um sinal analógico equivalente. Esse conversor também poderá ser utilizado em demais placas desde que a frequência de corte seja respeitada.

[1] Os tempos foram medidos com a utilização do *software* Postman para os testes no Desktop e Restler para os testes Mobile.

4.3 Desenvolvimento API

Como abordado anteriormente, quando o sistema embarcado deixou de ser o elemento central para se tornar um elemento passivo no projeto, se fez necessário a implementação de uma estrutura que pudesse interligar os elementos, e baseado nisso, foi desenvolvida a API conforme a etapa da API especificou. Esta abordagem obteve bons resultados de execução com $13\ ms$ no desktop e $34\ ms$ no celular como tempo médio de execução se comparado ao tempo médio de execução da abordagem anterior que ficava na casa de $2\ s$.

Foi realizado um teste para verificar o funcionamento do *WebClient* presente no sistema embarcado, o teste consistiu em conectar o MEGA na rede e iniciar a API. Nas primeiras execuções a API não possuía nenhum cliente ou silo cadastrado, logo era esperado que a API descartasse as informações ao receber a requisição. Com os testes, foi possível comprovar que o comportamento esperado estava ocorrendo e as informações eram descartadas sem que o servidor local que hospedava a API perdesse a conexão. No caso do silo existente, a API conseguia salvar corretamente as informações no *database*. Os demais testes foram executados em conjunto com os testes do aplicativo.

Uma dificuldade enfrentada durante o desenvolvimento da API está relacionada a infraestrutura da rede adotada, houve uma dificuldade em realizar as requisições dado que ocasionalmente a rede local criada para os testes ficava indisponível, porém mediante testes em outras redes pode ser aferido que esta dificuldade estava ligada ao modem utilizado para criação da rede local.

4.4 Desenvolvimento do Aplicativo

Para realizar os demais testes se fez necessário o uso da interface gráfica descrita na etapa do Aplicativo. Os testes foram executados seguindo a ordem natural de utilização do sistema, onde os primeiros testes foram a respeito da criação do cliente e do silo, os demais testes foram a respeito da utilização e interpretação do ambiente gráfico. Todos os testes seguiram 3 cenários descritos na Tabela 4.1 os cenários 1 e 2 foram utilizados para aferir o funcionamento em 2 plataformas diferentes a fim de testar se o comportamento esperado estava sendo atendido, o terceiro cenário foi adotado para simular o funcionamento em uma situação real onde diversos silos estivessem sendo monitorados gerando assim várias requisições para a API.

Para utilizar o terceiro cenário foi necessária a criação de um programa escrito em Python para enviar periodicamente valores referente ao ambiente do silo gerados de maneira aleatória junto do codSerie codificado, conferindo assim ao programa um funcionamento análogo ao MEGA que pode ser facilmente replicado.

Tabela 4.1 – Descrição dos cenários de testes.

	Cenário 1	**Cenário 2**	**Cenário 3**
Dispositivos	Desktop	Celular	Celular, Desktop
MEGA	Presente	Presente	Presente
Simulação MEGA	Não presente	Não presente	Presente
Quantidade de Simuladores MEGA	Não se aplica	Não se aplica	3

Fonte: Elaboração própria.

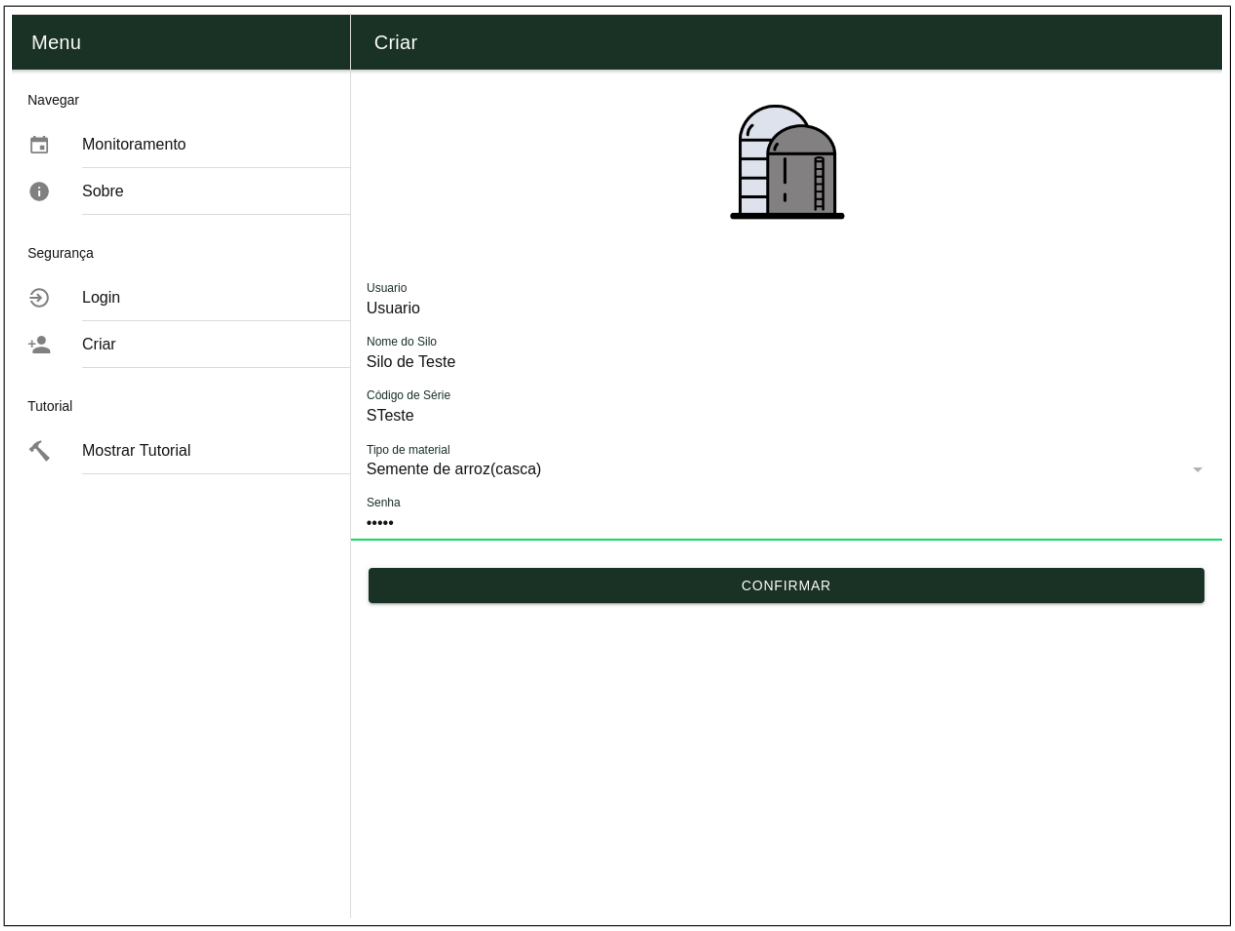
O primeiro teste foi um teste para aferir o funcionamento da criação do cliente e silo no aplicativo que verificou o preenchimento dos dados do formulário e conforme a Figura 4.3 demonstra que os campos ficam marcados e só é permitido cadastrar se todos os dados foram preenchidos.

Figura 4.3 – Criação do cliente e silo.

Fonte: Elaboração própria.

A Figura 4.4 ilustra um preenchimento de exemplo que foi utilizado para aferir o funcionamento da API no que diz a respeito da criação do cliente e do silo, neste teste a API conseguiu ler corretamente todos os dados populando assim a base de dados, a figura em questão demostra o funcionamento no cenário 2 e para os demais cenários o mesmo comportamento pode ser observado.

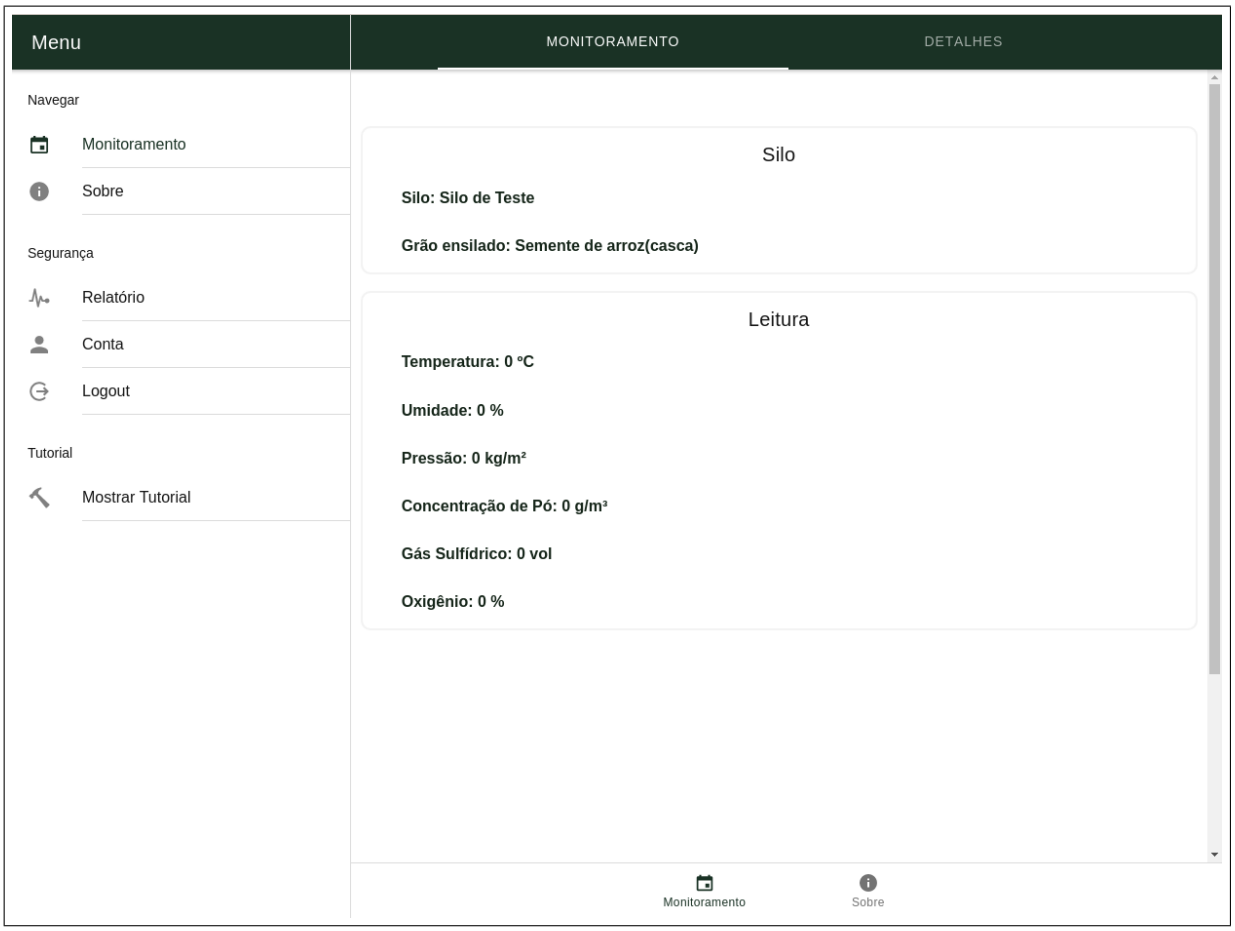
Figura 4.4 – Criação do cliente e silo.



Fonte: Elaboração própria.

A Figura 4.5 demostra o painel principal no seguimento de monitoramento, nele pode ser observado os valores de leitura de cada uma das características do ambiente interno do silo, essas características não possuíam valores pois se tratava de um silo recém cadastrado.

Figura 4.5 – Painel de monitoramento.

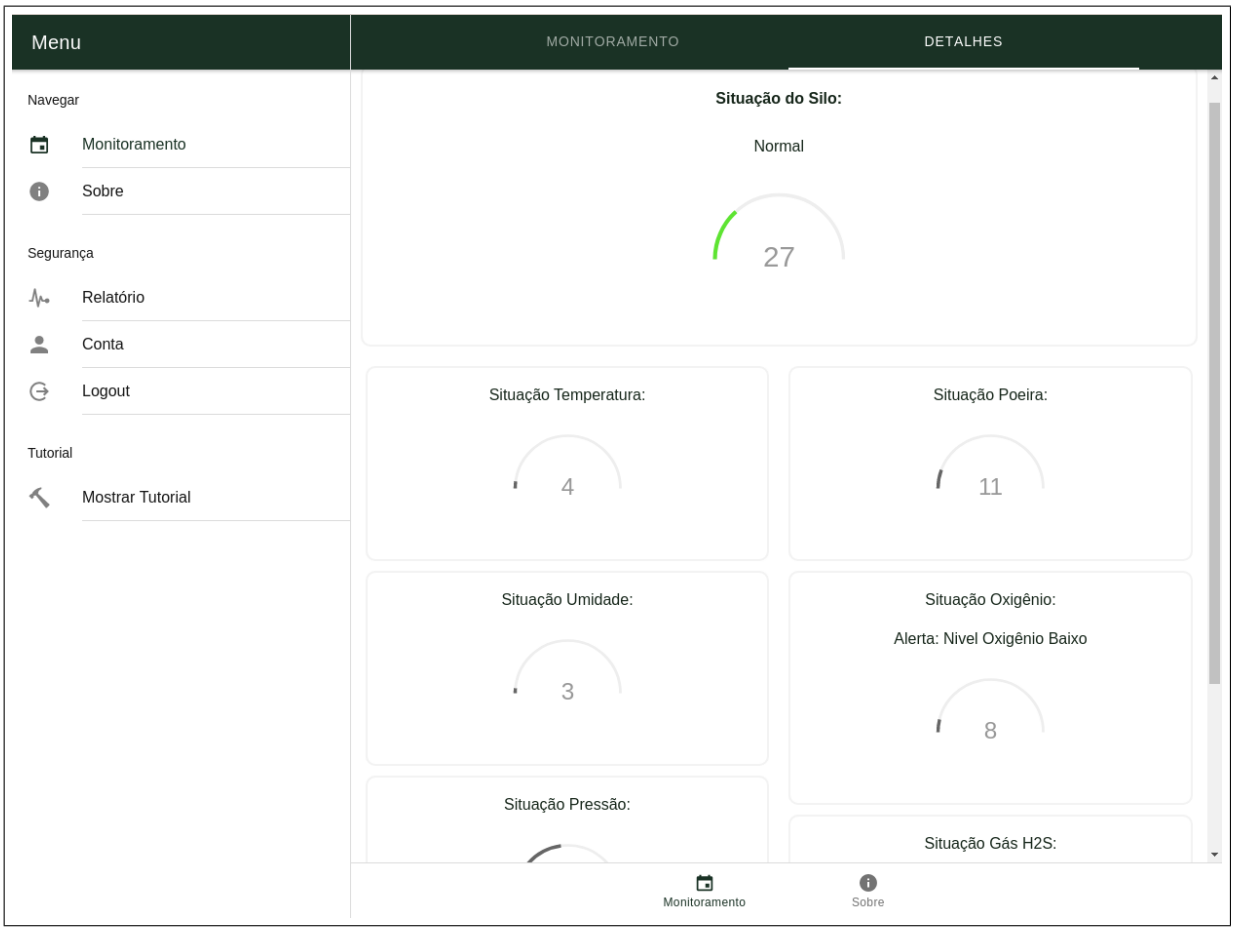


Fonte: Elaboração própria.

A Figura 4.6 demonstra o resultado do teste realizado para aferir as leituras, as situações do silo foram calculadas respeitando as equações demonstradas na seção referente a API gerando assim o comportamento esperado, a figura demostra o cenário 1 e os demais cenários demonstraram o mesmo comportamento e em especial no terceiro cenário a API conseguiu controlar corretamente cada um dos dispositivos.

Além de demonstrar a porcentagem em cada um dos blocos o aplicativo também informava a respeito da situação do ponto de vista do utilizador, deste modo o sistema exibia por exemplo se o nível de oxigênio estava alto, baixo ou seguro naquele instante.

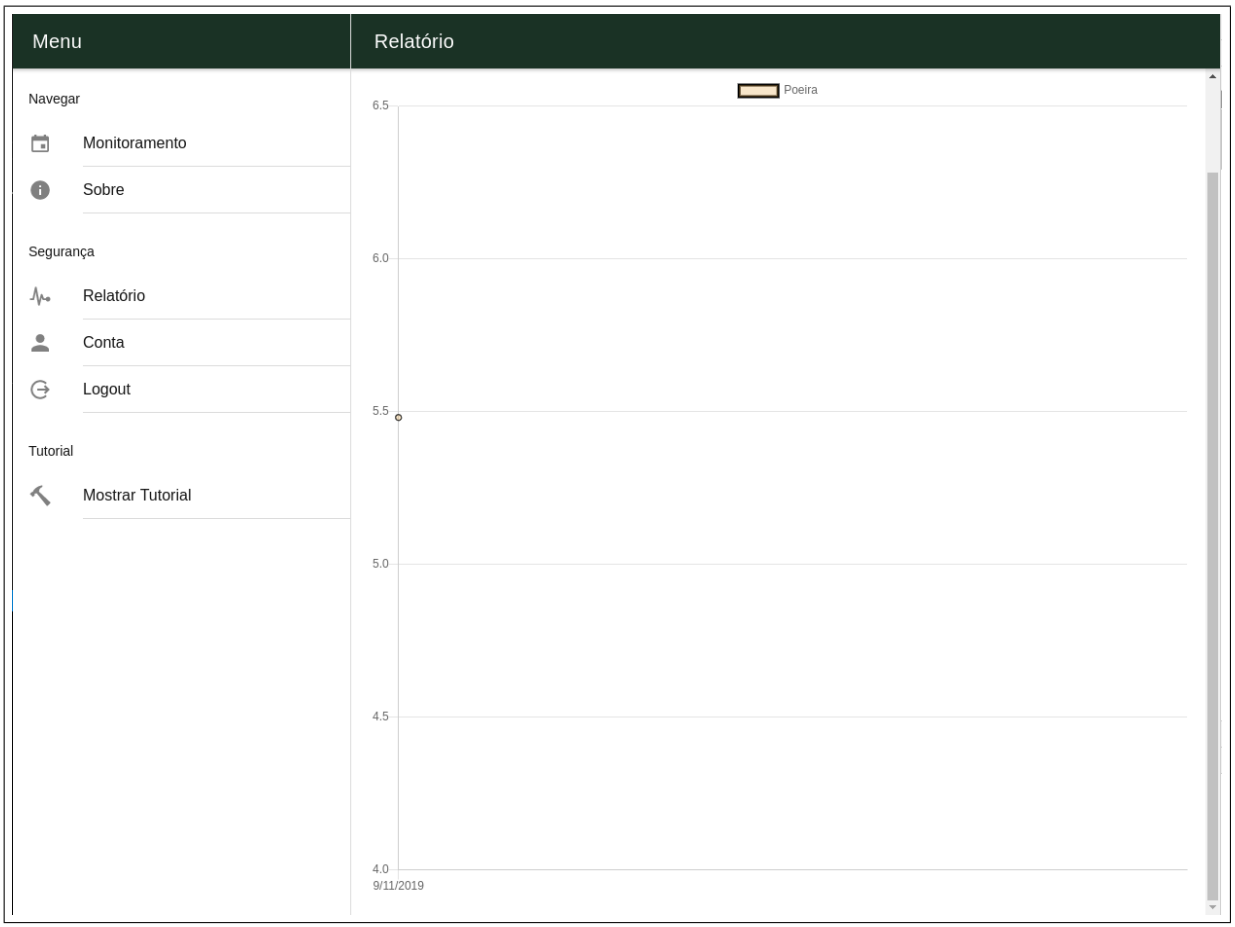
Figura 4.6 – Painel de detalhes.



Fonte: Elaboração própria.

Na Figura 4.7 está demonstrado o exemplo do relatório diário sobre a poeira do silo que foi criado para exemplificar os testes. Para os silos que não possuiam leituras, nenhum dado era exibido.

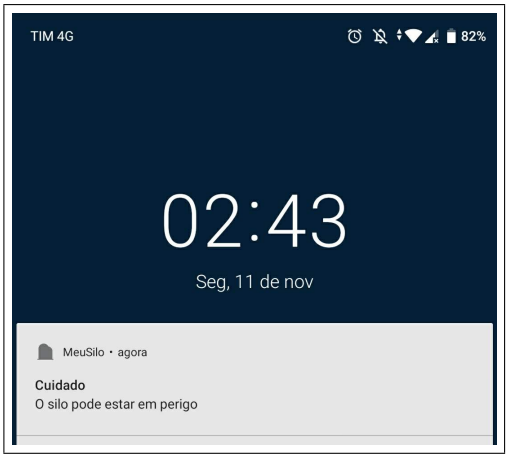
Figura 4.7 – Relatório diário teste.



Fonte: Elaboração própria.

A Figura 4.8 demonstra a notificação que foi gerada quando as características do ambiente foram elevadas para níveis não seguros.

Figura 4.8 – Notificação Celular.

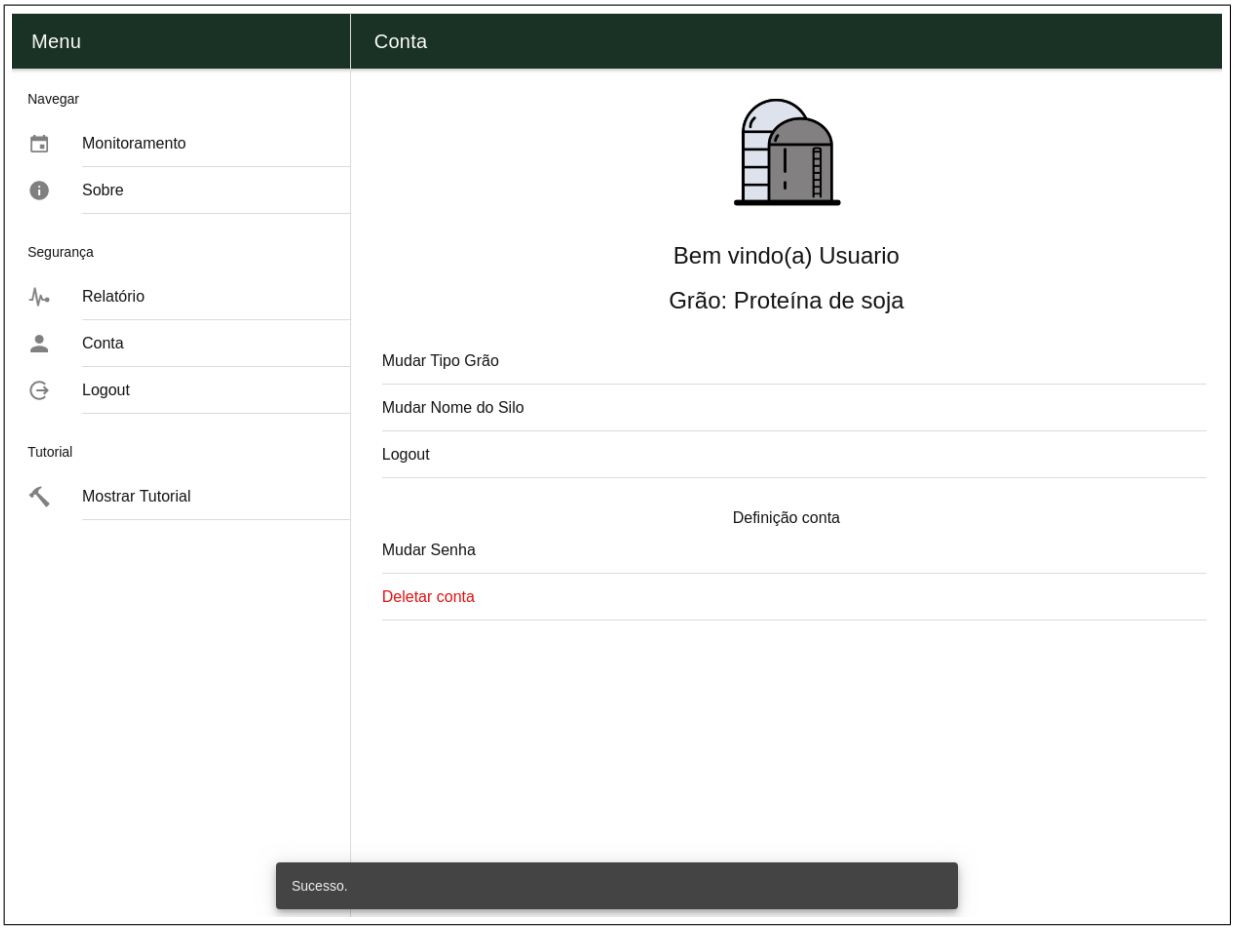


Fonte: Elaboração própria.

Por fim a Figura 4.9 demonstra a janela de alteração da conta, os testes realizados

demonstraram que para um usuário que estivesse conectado em mais de um lugar a alteração de características como o tipo do grão resultavam na alteração do tipo do grão e consequentemente na classificação geral da situação do silo em todos os dispositivos conectados. A Figura 4.9 também demostra a mensagem de sucesso para uma alteração bem sucedida. As outras mensagens referentes a segurança do operador também foram testadas.

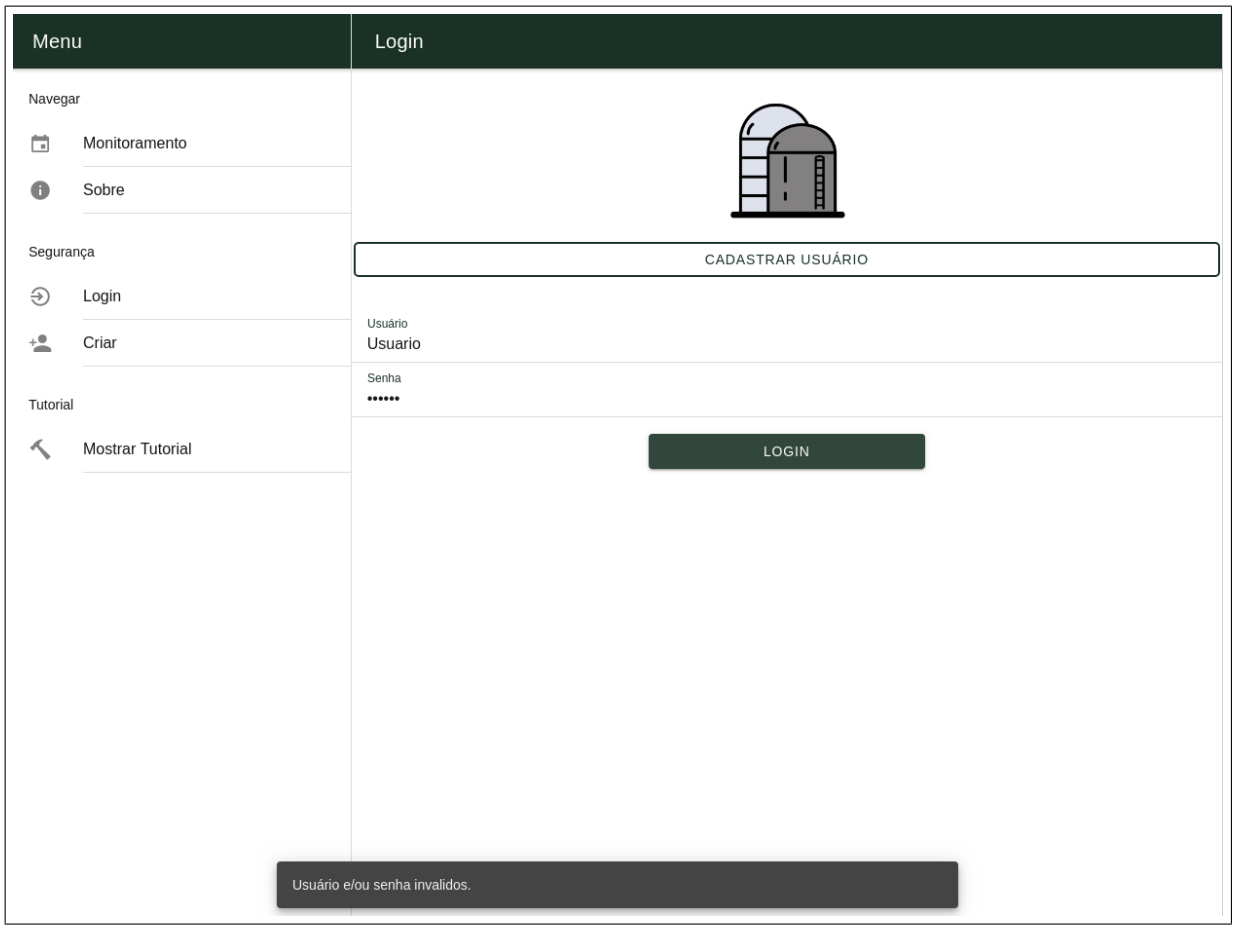
Figura 4.9 – Alteração das características.



Fonte: Elaboração própria.

Para a exclusão do cliente, a API exclui todas as informações relacionadas ao cliente que foram coletadas, e conforme demonstra a Figura 4.10 o sistema não pode mais ser acessado pelo cliente.

Figura 4.10 – Demonstração da exclusão.



Fonte: Elaboração própria.

Logo a utilização da API conferiu ao projeto:

- Boa escalabilidade, dado que a API conseguiu trabalhar com um volume médio de requisições.
- Ambiente central que confere ao sistema a possibilidade da modularização.
- Ambiente onde o processamento de dados pode ser realizado, o que diminuiu a necessidade de processamento dos dispositivos terminais.

Vale ressaltar que mediante os testes, a utilização do aplicativo também conferiu ao projeto um comportamento similar independente da plataforma utilizada.

4.5 Análise e Discussão dos Resultados

Uma análise sobre a validade do projeto pode ser aferida comparando as funcionalidades do presente trabalho com outros trabalhos relacionados ao tema que foram expostos na sessão dos Trabalhos Correlatos.

Deste modo um quadro sobre as implicações do presente trabalho pode ser elaborado. A Tabela 4.2 apresenta o comparativo do trabalho presente com os trabalhos abordados.

Tabela 4.2 – Comparativo com trabalhos correlatos.

	Projeto Atual	**Ferrasa, Biaggioni e Dias (2010)**	**Citolin (2012)**
Protótipo	Não possui	Presente	Presente
Características monitoradas	Pressão, Umidade, Temperatura, Poeira, Gás H2S e Oxigênio	Temperatura e Umidade	Temperatura
Interface Gráfica	Presente	Presente	Presente
Tipo da interface gráfica	Web	Desktop Local	Desktop Local

Fonte: Elaboração própria.

De tal modo pode ser aferido que o presente trabalho possui uma deficiência relacionada a aplicação imediata do mesmo. Esta deficiência se deu pelo fato de que o monitoramento dessas características envolvem o uso de sensores complexos e de alto custo, que culminaram na opção da simulação dos mesmos. Um ponto positivo em relação aos trabalhos correlatos é o monitoramento levar em consideração tanto a possibilidade de ocorrer uma explosão bem como um monitoramento que classifica o ambiente do ponto de vista das normas técnicas para ambientes confinados, além do monitoramento das condições ideais para manter a qualidade do grão.

No que diz a respeito da interface gráfica para visualização dos dados pelo usuário, todos os sistemas adotaram uma interface porém o presente trabalho se diferencia por adotar uma interface que pode ser acessada por vários tipos de dispositivos que possuam uma interface *Web browser* com suporte ao JavaScript.

5 CONCLUSÃO

A partir dos resultados obtidos e discutidos no Capítulo 4 pode ser concluído que o sistema completo composto pelos módulos de sensoriamento, sistema embarcado, API e aplicativo operou bem de tal modo a prover um sistema que conseguiu monitorar, classificar e disponibilizar as informações referentes ao ambiente interno do silo.

As etapas de sensoriamento e do sistema embarcado proveram um ferramental que pode ser aplicado em outras necessidades, enquanto as etapas da API e do aplicativo demonstraram que as diferentes tecnologias que foram empregadas no projeto trabalharam bem em conjunto, gerando assim um ponto de partida para que outros projetos relacionados ao tema possam ser realizados, colaborando assim para uma expansão na utilização de soluções tecnológicas na agricultura.

Em conclusão, o presente trabalho atendeu os objetivos específicos bem como o objetivo geral.

5.1 Trabalhos Futuros

Para o desenvolvimento de trabalhos futuros, tem-se o uso de sensores que consigam mensurar fisicamente o ambiente interno do silo de modo a eliminar as etapas de simulação trazendo assim o projeto para uma aplicação real e não mais em um ambiente simulado e controlado.

O projeto possui uma boa aplicação para a área de mineração de dados, podendo ser utilizado em uma situação real monitorando apenas a temperatura, umidade e nível de gás durante um período de tempo obtendo assim os dados de várias safras podendo verificar dentre essas características quais as condições ideais para se obter a melhor conservação dos grãos.

O projeto também pode ser estendido na adoção e controle de atuadores que seriam acionados sempre que uma situação crítica ocorresse.

Melhorias podem ser feitas no controle de usuário e sessão além do controle dos silos, adicionando telas destinadas para o cadastro e controle de silos permitindo que o sistema seja expandido para o uso com vários silos.

5.2 Considerações Finais

Como abordado anteriormente o presente trabalho se trata de um trabalho inicial e requer evoluções principalmente no que diz respeito ao emprego de sensores reais para posterior prototipação que eventualmente pode gerar um produto que seria de grande utilidade para o cenário nacional atual.

Referências

ABNT. **NBR 14787 Espaço confinado - Prevenção de acidentes, procedimentos e medidas de proteção**. [S.l.], 2001. Citado na página 9.

ARDUÍNO. **Arduino Mega 2560 REV3**. 2018. Disponível em: <https://store.arduino.cc/usa/arduino-mega-2560-rev3>. Citado 2 vezes nas páginas 15 e 16.

ARDUÍNO. **Arduino UNO REV3**. 2019. Disponível em: <https://store.arduino.cc/usa/arduino-uno-rev3>. Citado na página 15.

CARNEIRO, O. Silos e sua construção. **Anais da Escola Superior de Agricultura Luiz de Queiroz**, FapUNIFESP (SciELO), v. 5, n. 0, p. 3–34, 1948. Disponível em: <https://doi.org/10.1590/s0071-12761948000100001>. Citado na página 3.

CHEREMISINOFF, N. **Dust explosion and fire prevention handbook : a guide to good industry practices**. Hoboken, New Jersey: John Wiley and Sons, Inc, 2014. ISBN 9781118773505. Citado 5 vezes nas páginas 2, 4, 6, 7 e 8.

CITOLIN, R. S. **Termometrix sistema de termometria para silos**. 2012. Monografia (Bacharel em Engenharia Elétrica), UFRGS (Universidade Federal do Rio Grande do Sul), Porto Alegre, Brasil. Citado 2 vezes nas páginas 12 e 58.

ESTADO, A. **Explosão do silo no Porto de Paranaguá fere 18 - Política**. Estadão, 2012. Disponível em: <https://politica.estadao.com.br/noticias/geral,explosao-do-silo-no-porto-de-paranagua-fere-18,20011116p34191>. Citado na página 1.

FAWCETT, J. **Beginning XML**. Indianapolis, IN: Wiley, 2012. ISBN 9781118162132. Citado na página 10.

FERRASA, M.; BIAGGIONI, M. A. M.; DIAS, A. H. Sistema de monitoramento da temperatura e umidade em silos graneleiros via radiofrequência (rf). **Revista Energia na Agricultura, Botucatu, vol. 25, n.2**, 2010. Citado 2 vezes nas páginas 11 e 58.

GASQUES, J. G.; BACCHI, M. R. P.; BASTOS, E. T. **Crescimento e Produtividade da Agricultura Brasileira de 1975 a 2016**. IPEA, 2018. Disponível em: <http://www.ipea.gov.br/cartadeconjuntura/index.php/2018/03/02/crescimento-e-produtividade-da-agricultura-brasileira-de-1975-a-2016/>. Citado na página 1.

HOEFS, J. **Firmata Protocol**. 2018. Disponível em: <https://github.com/firmata/protocol>. Citado na página 14.

IBGE. **Tabela 259: Número de informantes e Capacidade útil dos armazéns e silos para produtos a granel**. SIDRA, 2019. Disponível em: <https://sidra.ibge.gov.br/tabela/259>. Citado na página 1.

LOPES, S. **Aplicações mobile híbridas com Cordova e PhoneGap**. [S.l.]: Casa do Código, 2016. ISBN 9788555191572. Citado 2 vezes nas páginas 17 e 18.

MAINIER, F. B.; VIOLA, E. D. M. O sulfeto de hidrogênio (h2s) e o meio ambiente. **II Simpósio de Excelência em Gestão e Tecnologia – SEGeT 2005**, 2005. Citado na página 7.

MASSE, M. **REST API Design Rulebook: Designing Consistent RESTful Web Service Interfaces**. [S.l.]: O'Reilly Media, 2011. ISBN 9781449319908. Citado na página 11.

NR33. **Segurança e saúde nos trabalhos em espaços confinados**. [S.l.], 2012. Citado na página 8.

PALMA, G. **Pressões e fluxo em silos esbeltos (h/d≥1,5)**. 2005. Exame de Dissertação (Mestrado) – Escola de Engenharia de São Carlos, Universidade de São Paulo. Citado na página 3.

REATIVA. **Tabela Poeiras Agrícolas, NFPA adaptado**. 2018. Disponível em: <http://www.reativa.com/>. Citado na página 5.

SADOWSKI, A.; ROTTER, J. Steel silos with different aspect ratios: I — behaviour under concentric discharge. **Journal of Constructional Steel Research**, Elsevier BV, v. 67, n. 10, p. 1537–1544, out. 2011. Disponível em: <https://doi.org/10.1016/j.jcsr.2011.03.028>. Citado na página 3.

SALAFIA, C.; VELáSQUEZ, J. R. **MyOpenLab**. 2017. Disponível em: <https://myopenlab.org>. Citado na página 14.

SCARDINO, P. **NR-33 e atualização NBR 16.577 – Espaços Confinados**. [S.l.], 2017. Citado na página 7.

SESHADRI, S.; GREEN, B. **Desenvolvendo com AngularJS: Aumento de Produtividade com Aplicações Web Estruturadas**. [S.l.]: Novatec Editora, 2014. ISBN 9788575224090. Citado na página 17.

SILOGRÃO. **SILO ELEVADO**. 2017. Disponível em: <http://silograo.com.br/produto/1138>. Citado na página 4.

Sá, A. **Prevenção e Controle dos Riscos com Poeiras Explosivas**. 1997. Citado 5 vezes nas páginas 4, 5, 6, 7 e 8.

TANENBAUM, A. **Redes de computadores**. São Paulo: Pearson Prentice Hall, 2011. ISBN 857605924X. Citado 2 vezes nas páginas 9 e 10.

WIZNET. **W5100 Datasheet**. 2008. Disponível em: <https://img.filipeflop.com/files/download/Datasheet_W5100_v1_1_6.pdf>. Citado na página 16.