UNIVERSIDADE DE CAXIAS DO SUL
ÁREA DO CONHECIMENTO DE CIÊNCIAS EXATAS E ENGENHARIAS

RAFAEL SILVESTRE ADAMATTI

DESENVOLVIMENTO DE UM PROTÓTIPO DE SISTEMA DE GERENCIAMENTO DE MANUTENÇÃO

CAXIAS DO SUL

2024

RAFAEL SILVESTRE ADAMATTI

DESENVOLVIMENTO DE UM PROTÓTIPO DE SISTEMA DE GERENCIAMENTO DE MANUTENÇÃO

Trabalho de Conclusão de Curso apresentado como requisito parcial à obtenção do título de Bacharel em Engenharia de Controle e Automação na Área do Conhecimento de Ciências Exatas e Engenharias da Universidade de Caxias do Sul.

Orientador: Prof. Me. Ricardo Leal Costi

CAXIAS DO SUL

2024

RAFAEL SILVESTRE ADAMATTI

DESENVOLVIMENTO DE UM PROTÓTIPO DE SISTEMA DE GERENCIAMENTO DE MANUTENÇÃO

Trabalho de Conclusão de Curso apresentado como requisito parcial à obtenção do título de Bacharel em Engenharia de Controle e Automação na Área do Conhecimento de Ciências Exatas e Engenharias da Universidade de Caxias do Sul.

Aprovado(a) em 27/06/2024

BANCA EXAMINADORA

_____

Prof. Me. Ricardo Leal Costi
Universidade de Caxias do Sul - UCS

_____

Prof. Ma. Patricia Giacomelli
Universidade de Caxias do Sul - UCS

_____

Prof. Dr. Ricardo Vargas Dorneles
Universidade de Caxias do Sul - UCS

RESUMO

Para garantir o funcionamento constante de máquinas e equipamentos é imprescindível que sejam realizadas as manutenções adequadas. Em fábricas cada vez maiores, com um elevado número de máquinas se faz necessário a centralização dos dados para monitoramento. O avanço nas tecnologias da Internet das Coisas (IoT) possibilita a automação desses processo. Buscando um meio para gerenciar as rotinas de manutenção, esse trabalho propôs o desenvolvimento de um protótipo de sistema web, utilizando de dados coletados das máquinas. Através do cadastro dos equipamentos, sensores, rotinas e alarmes, foi possível armazenar dados e realizar validação que auxiliam na manutenção corretiva, preventiva e preditiva.

Palavras-chave: Sistema Web. IoT. Manutenção. CMMS.

ABSTRACT

To ensure the constant functioning of machines and equipment, it is essential that appropriate maintenance has been carried out. In increasingly larger factories, with a high number of machines, it is necessary to centralize data for monitoring. The advancement in Internet of Things (IoT) technologies enable the automation of these processes. Looking for a means to manage maintenance routines, this work proposed the development of a web system prototype, using data collected from machines. Through the registration of equipment, sensors, routines and alarms, it was possible to store data and carry out validation that assist in corrective, preventive and predictive maintenance.

Keywords: Web System. Iot. Maintenance. CMMS.

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

IoT *Internet of Things*

CMMS *Computerized Maintenance Management System*

ADC *Analog Digital Converter*

HTTP *Hypertext Transfer Protocol*

HTTPS *Hypertext Transfer Protocol Secure*

Wi-Fi *Wireless Fidelity*

WLAN *Wireless Local Area Network*

URL *Uniform Resource Locator*

TLS *Transport Layer Security*

MQTT *Message Queue Telemetry Transport*

MVC *Model–View–Controller*

SGBD Sistema de Gerenciamento de Banco de Dados

SQL *Structured Query Language*

ANSI *American National Standards Institute*

OEE *Overall equipment efficiency*

IDE *Integrated Development Environment*

EF *Entity Framework*

JSON *JavaScript Object Notation*

IIS *Internet Information Services*

SUMÁRIO

1 INTRODUÇÃO

Com o aumento da produção industrial viu-se uma necessidade de criar formas para manter as máquinas em pleno funcionamento por mais tempo, isso levou ao surgimento da chamada manutenção industrial. Essa que é caracterizada pelo desenvolvimento de técnicas de planejamento, organização e controle das manutenções (Gregório; Silveira, 2018).

Nas últimas décadas, os avanços tecnológicos têm provocado uma transformação radical nas indústrias, conduzindo-as à era da Indústria 4.0. Ela se caracteriza pela automação de vários sistemas, digitalização e crescimento na tendência de implantação da Internet das Coisas, do Inglês *Internet of Things* (IoT) (Adebayo; Chaubey; Numbu, 2019). O crescimento na implantação da IoT revolucionou a forma como as indústrias de todos os setores abordam a manutenção de ativos e equipamentos. Trazendo uma nova dimensão de precisão e eficácia para as manutenções preditiva e preventiva, que são essenciais para garantir eficiência operacional, reduzir os custos de reparo e evitar interrupções não planejadas. Isso se deve à coleta contínua de dados de sensores instalados em máquinas e sistemas, permitindo que as organizações antecipem situações de falha, programem intervenções antes que os problemas ocorram e otimizem os recursos de manutenção.

1.1 JUSTIFICATIVA

A ausência de uma correta execução da manutenção em máquinas pode acarretar uma série de problemas operacionais e financeiros para uma empresa. A falta de manutenção preventiva pode resultar em desgaste acelerado de peças, aumentando a probabilidade de falhas inesperadas e paradas não programadas na produção. Isso não apenas impacta a eficiência operacional, mas também pode levar a custos elevados de reparo e substituição de componentes danificados. Além disso, a negligência na implementação de uma manutenção corretiva rápida e eficiente pode prolongar ainda mais os períodos de inatividade, prejudicando a produtividade e comprometendo prazos de entrega.

A coleta contínua e em tempo real de dados proporcionada pela IoT permite que as organizações tenham uma abordagem mais proativa à manutenção. Para que esses dados sejam utilizados de forma mais eficiente e rápida utiliza-se uso de softwares de gerenciamento de manutenção, conhecidos como *Computerized Maintenance Management System* (CMMS). Esses são sistemas usados para planejar, programar, gerenciar e monitorar atividades de manutenção de equipamentos, máquinas, automóveis e outras instalações (Shankar; Singh; Singh, 2023).

Para acompanhar essas evoluções que estão ocorrendo na indústria, propõe-se o desenvolvimento de um protótipo de sistema de gerenciamento de manutenção, utilizando diversos conceitos estudados ao longo do curso, como aquisição de dados, protocolos de comunicação,

banco de dados e programação.

1.2 OBJETIVOS

Este trabalho tem como objetivo desenvolver um sistema de controle de manutenção através do monitoramento de parâmetros em máquinas. A partir do objetivo geral, identificou-se os objetivos específicos que se seguem:

1. Definir os de parâmetros que serão monitorados.
2. Construir uma base de dados para armazenar as informações.
3. Especificar o protocolo de comunicação entre o sistema de aquisição e o de controle.
4. Desenvolver o sistema de controle para monitoramento dos dados, com cadastro de alarmes.

1.3 ESTRUTURA DO TRABALHO

O presente trabalho está organizado da seguinte forma:

- O Capítulo 2 apresenta a revisão bibliográficas dos conceitos utilizados para a aquisição e tratamento dos dados.
- O Capitulo 3 apresenta a metodologia de desenvolvimento para o projeto proposto.
- O Capitulo 4 apresenta as conclusões finais do trabalho.

1.4 LIMITAÇÕES DO TRABALHO

Para que se atinja todos os objetivos propostos serão aplicadas algumas restrições ao escopo desse projeto. A primeira é referente ao dispositivo de aquisição de dados, será utilizado um *hardware* de mercado com um *firmware* desenvolvido somente para os testes do protótipo, seu desenvolvimento não será abordado no trabalho. Como segunda, o sistema de gerenciamento não irá realizar análises dos dados para previsão de falhas.

2 REVISÃO BIBLIOGRÁFICA

Neste capítulo serão apresentados os conceitos de manutenção, aquisição de dados, protocolos de comunicação e desenvolvimento web.

2.1 MANUTENÇÃO

A manutenção desempenha um papel fundamental na indústria moderna, mas para entender a sua importância é necessário primeiro entender alguns conceitos que estão atrelados a ela (Gregório; Silveira, 2018), que são:

- Confiabilidade: representa a probabilidade de um item operar de forma satisfatória em um intervalo de tempo.
- Produtividade: porcentagem do tempo que o funcionário de manutenção de fato trabalha em manutenção.
- Eficiência: capacidade do equipamento de operar com o mínimo de desperdício possível.
- Disponibilidade: é o período em que a máquina está disponível para executar a sua função.
- Qualidade: quando o produto está de acordo com o esperado.
- Performance: é a capacidade de produção de um equipamento em relação à produção teórica.
- Mantenabilidade: a capacidade de um item de receber manutenção.

Com a utilização desses conceitos é possível elaborar indicadores e definir ações que devem ser tomadas para cada caso, assim melhorando os resultados obtidos, um exemplo é a Eficácia Geral do Equipamento, do inglês *Overall equipment efficiency* (OEE). A manutenção pode ser dividida em três estratégias principais: corretiva, preventiva e preditiva.

Segundo Gregório e Silveira (2018) para que seja escolhida a melhor estratégia é necessário levar alguns pontos em consideração. Sendo recomendação do fabricante um deles, esse que indica intervalos de tempo e procedimentos para correção de falhas. Outro ponto é a segurança do trabalho e meio ambiente, que possibilita a integração entre homem, máquina e meio ambiente obedecendo as exigências legais. As características do equipamento também são um fator importante a ser considerado, como o tempo médio de reparo e tempo médio entre falhas. Por fim é necessário olhar para o fator econômico, que leva em conta os recursos humanos, material, perdas no processo e interferência na produção.

A manutenção corretiva é a estratégia caracterizada pela correção após o problema já ter aparecido, podendo ser dividido em dois tipos. A não planejada, que se deve a uma falha inesperado em algum componente, podendo acarretar na parada da máquina. E a programada, normalmente devido a um desempenho reduzido da máquina que exige a correção (Engeman, 2023).

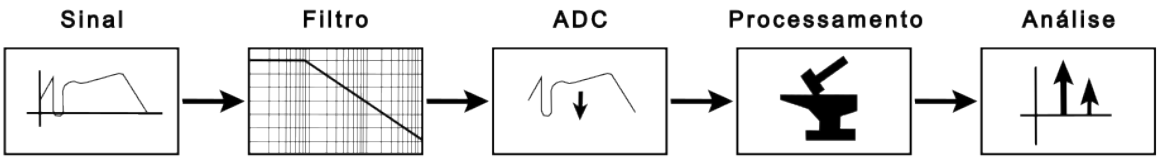
Outra bastante utilizada é a manutenção preventiva, que ocorre em intervalos definidos de tempo, que visa reduzir ou evitar falhas no equipamento, assim prevenindo paradas inesperadas. Uma desvantagem da sua utilização é possível troca de peças desnecessária, gerando custos e paradas adicionais (Engeman, 2023).

Por fim tem-se a manutenção preditiva como uma estratégia que emprega técnicas de análise de parâmetros, visando minimizar as intervenções corretivas e preventiva. Dessa forma, busca-se evitar paralisações não planejadas e a substituição desnecessária de componentes. Porém para que isso seja feito é necessário um investimento maior na parte de monitoramento dos equipamentos, para que seja possível realizar as análises necessárias (Gregório; Silveira, 2018).

2.2 AQUISIÇÃO DE DADOS

Um sistema de aquisição de dados é qualquer arranjo que capacite a coleta de sinais digitais ou a conversão de sinais analógicos em sinais digitais, tornando possível a interpretação e a manipulação por sistemas digitais (Balbinot; Brusamarello, 2019). A Figura 1 representa um sistema genérico de aquisição de dados, com o sinal de entrada, filtro, *Analog Digital Converter* (ADC), unidade de processamento de sinal (hardware) e análise do sinal.

Figura 1 – Diagrama de blocos de um sistema de aquisição de dados genérico.



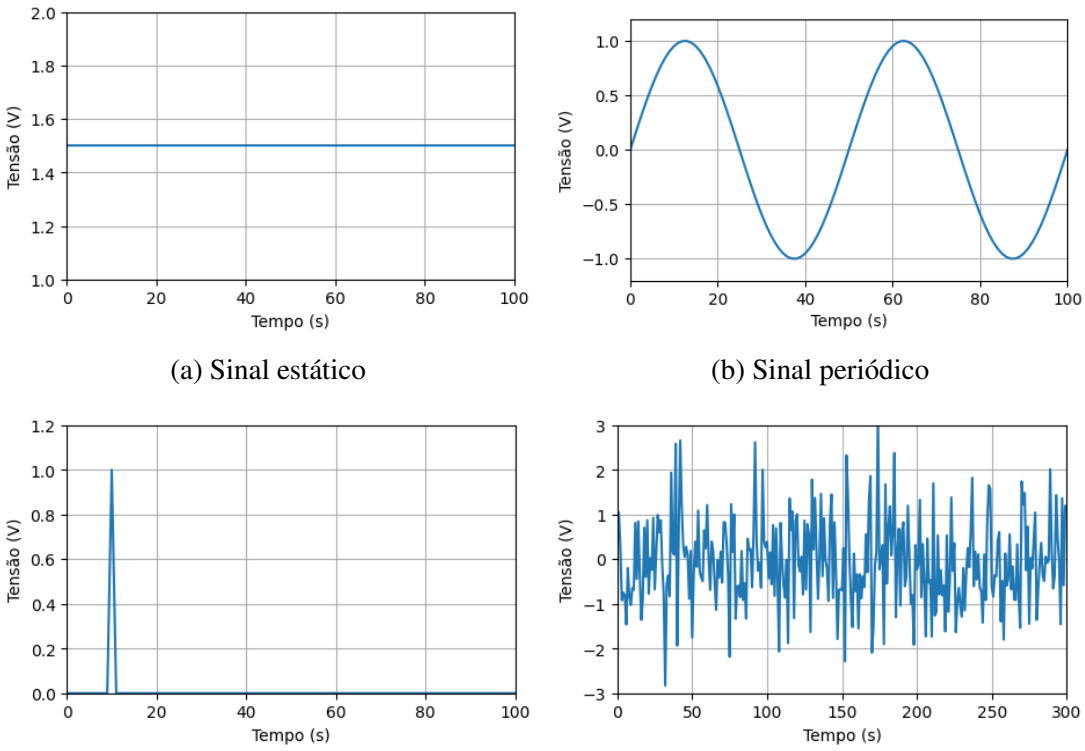
Fonte: Adaptado de Balbinot e Brusamarello (2019)

2.2.1 Sinais

Um sinal consiste em um conjunto de dados ou informações, podendo esses ser em função de variáveis independentes como o tempo, espaço, entre outras (Lathi, 2006). Segundo Balbinot e Brusamarello (2019), na área da instrumentação o comportamento dos sinais pode ser caracterizados de duas formas principais: no domínio do tempo e no domínio da frequência. Sendo divididos em cinco classes principais:

- Sinais estáticos: possuem um comportamento constante por um longo intervalo de tempo, conforme apresentado na Figura 2a.
- Sinais periódicos: apresentam uma repetição regular, como demonstrado na Figura 2b.
- Sinais transientes: em um determinado intervalo de tempo, ocorre um evento muito rápido se comparado com o todo, exemplificado na Figura 2c.
- Sinais determinísticos: aqueles que podem ser completamente expressados por funções matemáticas no tempo, como apresentado na Figura 2b.
- Sinais aleatórios ou estocásticos: não podem ser caracterizados com precisão ou antecipadamente, como demonstrado na Figura 2d.

Figura 2 – Exemplos das classes de sinais.



(a) Sinal estático

(b) Sinal periódico

(c) Sinal transiente

(d) Sinal aleatório ou estocástico

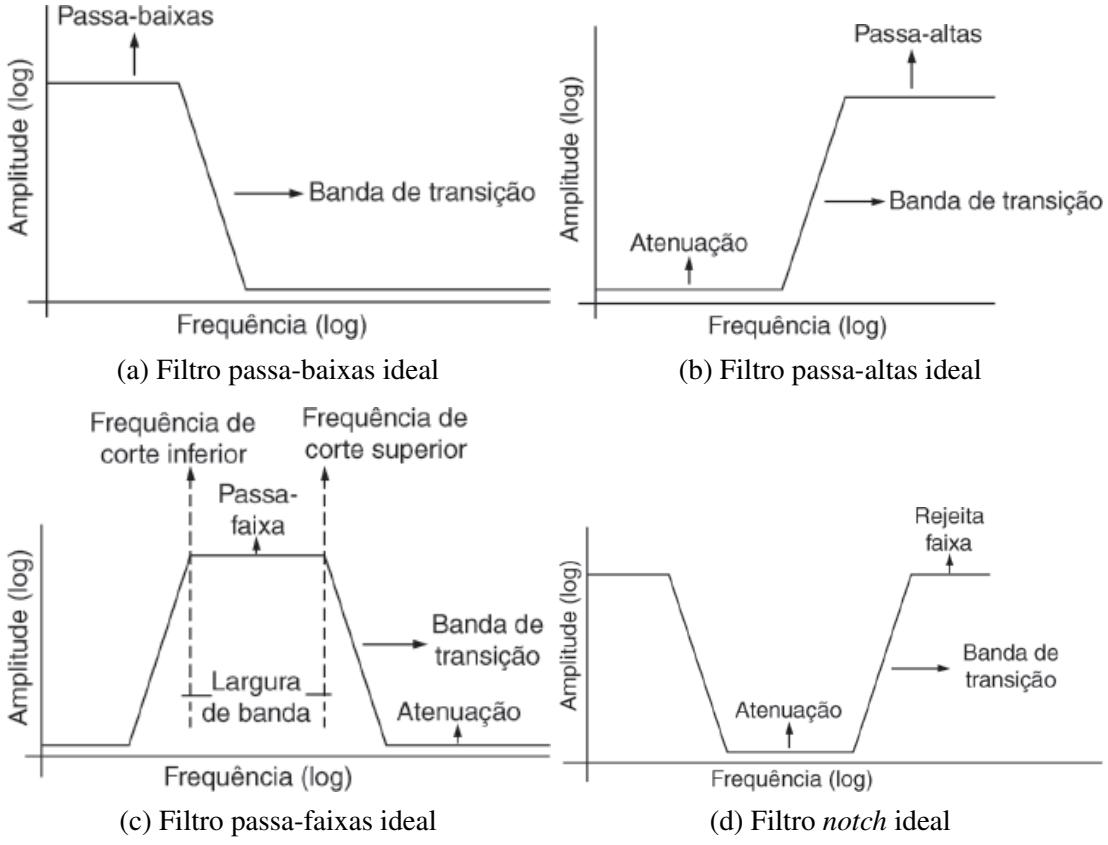
Fonte: Adaptado de Balbinot e Brusamarello (2019)

2.2.2 Filtragem

A filtragem é quando alguns componentes de frequência são atenuados ou completamente removidos, sendo interessante para diversas aplicações (Oppenheim; Willsky, 2010). Esse processo permite que os ruídos presentes nos sinais sejam minimizados ao eliminar ou reduzir as frequências desnecessárias.

Para isso podem ser usados alguns tipos de filtros: o Passa-baixas (Figura 3a) atua atenuando as altas frequências a partir de uma denominada frequência de corte; o Passa-altas (Figura 3b) que opera permitindo as frequências mais altas e reduzindo as baixas; o Passa-faixas (Figura 3c), uma combinação dos dois anteriores, que possui uma frequência de corte superior e inferior, atenuando qualquer frequência além dos pontos de corte, e o *notch* (Figura 3d), uma variação do Passa-faixas, que funciona de forma inversa atenuando somente uma faixa de frequências (Balbinot; Brusamarello, 2019).

Figura 3 – Tipos de filtros



(a) Filtro passa-baixas ideal

(b) Filtro passa-altas ideal

(c) Filtro passa-faixas ideal

(d) Filtro *notch* ideal

Fonte: Balbinot e Brusamarello (2019)

2.2.3 Digitalização

Para permitir que os sinais analógicos sejam processados por circuitos digitais é necessário a presença de um conversor que una esse dois mundos, conhecido como ADC. Ele é responsável por transformar em dados digitais e discretos sinais com características analógicas e contínuas (Lenz; Moraes, 2019).

Essa conversão é realizada por meio de uma relação, normalmente linear, do nível de tensão (entrada analógica) e de um número binário (saída digital), como exemplificado na Figura 4. Nela existe a resolução, ou erro de quantização, que é a maior variação possível do sinal de entrada em relação ao número de saída. Sendo o número de saída definido por $N = 2^n$, no

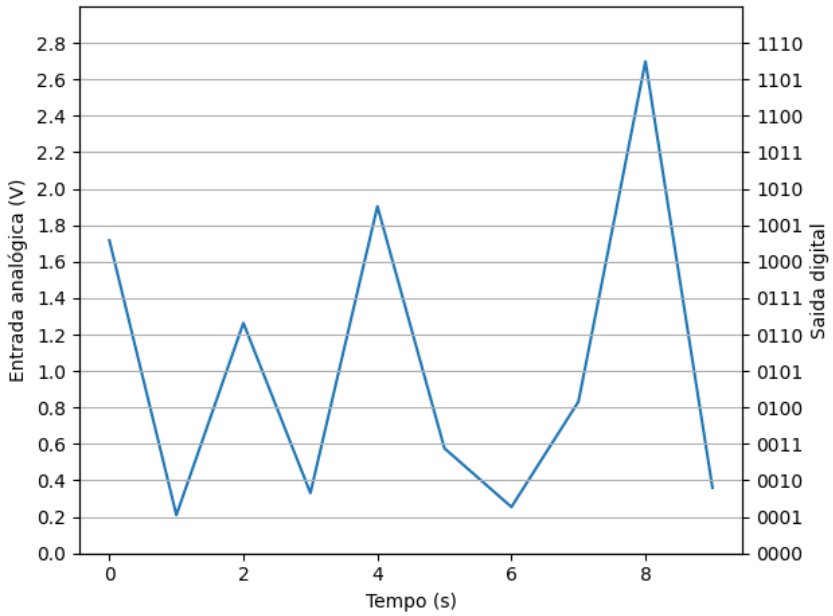
qual n é o número de bits do conversor (Balbinot; Brusamarello, 2019). Com isso, usando um sinal de entrada com uma faixa dinâmica de 5 V e um conversor de 4 bits, tem-se que:

$$N = 2^4 = 16$$

, e uma resolução de:

$$\frac{\Delta V_{in}}{N-1} = \frac{3}{16-1} = 0,2\ V$$

Figura 4 – Exemplo da relação em um ADC.



Fonte: O autor (2023)

2.3 TECNOLOGIA WI-FI

No início dos anos 90 foi criado o Grupo de Trabalho 802.11 por autorização do IEEE[1] com o intuito de definir uma especificação para a conectividade sem fio entre estações de trabalho (PCs) de uma *Wireless Local Area Network* (WLAN). Em 1997, após anos de pesquisa e trabalhos, foi aprovado o primeiro padrão 802.11 pelo comitê de padronização, com taxas de transmissão entre 1 e 2 Mbits/s (Rochol, 2018).

Para garantir uma melhor usabilidade dessa nova tecnologia por parte dos usuários, independente da marca, houve a união de diversas empresas em 1999, criando assim a associação denominada *Wi-Fi Alliance*. A partir dela foi criado o termo *Wireless Fidelity* (Wi-Fi) como novo nome para a tecnologia. Ela também define a logomarca de certificação a ser usada por produtos que atendem aos padrões acordados pela indústria em termos de interoperabilidade, segurança e uma variedade de protocolos específicos de aplicação.

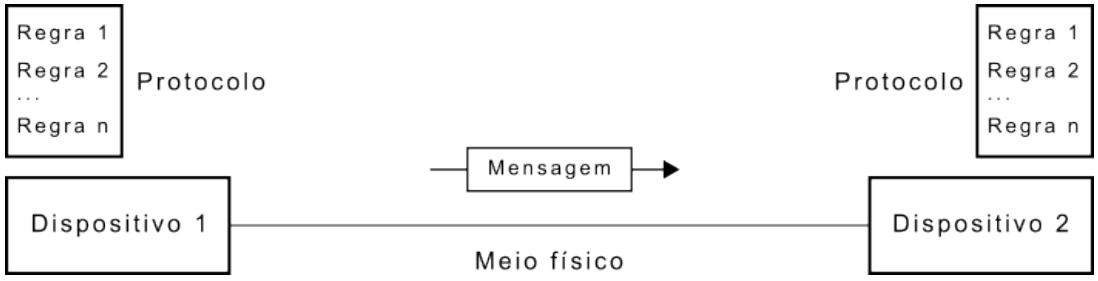
[1] *Institute of Electrical and Electronics Engineers*, fundada em 1963, é uma organização responsável pela criação de normas e padrões.

A tecnologia Wi-Fi permite a conexão entre uma vasta gama de dispositivos eletrônicos, como computadores, *smartphones*, *tablets*, TVs, entre outros, no qual a transmissão dos dados é feita através de ondas de rádio. Ao longo dos anos a Wi-Fi teve uma grande evolução em suas taxas de transmissão, indo de 2 Mbits/s para uma ordem de Gbits/s (Pahlavan; Krishnamurthy, 2021).

2.4 PROTOCOLOS DE COMUNICAÇÃO

Para que dois dispositivos possam se comunicar não basta estarem conectados por um meio físico, cabeado ou sem fio (Wi-Fi), é necessário que ambos utilizem as mesma regras, sendo esse conjunto o que define um protocolo de comunicação (Forouzan, 2010a). Na Figura 5 é exemplificado a utilização de um protocolo na comunicação entre dois dispositivos.

Figura 5 – Uso de protocolo na comunicação de dados



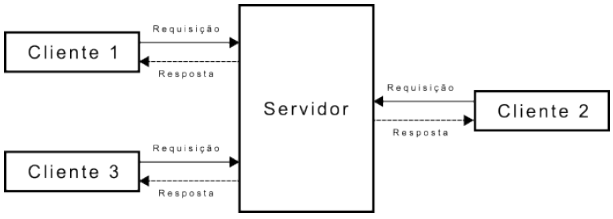
Fonte: Adaptado de Forouzan (2010a)

Eles possuem um papel fundamental nessa era digital, facilitando a comunicação e a troca de informações ao redor do mundo. Dentre uma ampla variedade de protocolos, em que cada um possui suas vantagens e desvantagens para certas aplicações, se destaca o *Hypertext Transfer Protocol* (HTTP).

2.4.1 *Hypertext Transfer Protocol*

O HTTP se utiliza de uma arquitetura do tipo requisição/resposta, que consiste no envio de uma solicitação por parte do cliente e na resposta processada pelo servidor (Wukkadada *et al.*, 2018), esse fluxo é representado na Figura 6.

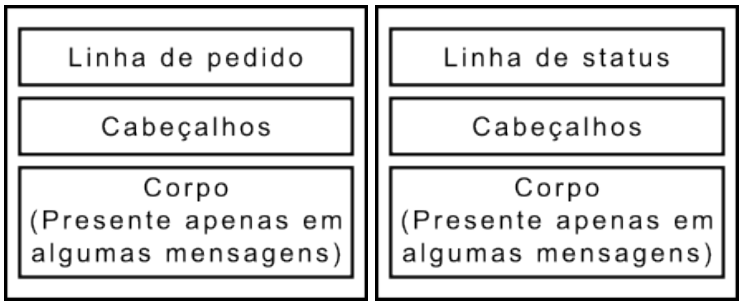
Figura 6 – Fluxo cliente servidor para HTTP



Fonte: Adaptado de Wukkadada *et al.* (2018)

Essa interação entre cliente e servidor é feita através de uma mensagem, podendo ser dívida em até 3 partes: a linha de pedido/status, o cabeçalhos, e as vezes o corpo (Forouzan, 2010b). A Figura 7 exemplifica essa estrutura.

Figura 7 – Mensagens de requisição e de resposta



(a) Mensagem requisição (b) Mensagem resposta

Fonte: Adaptado de Forouzan (2010b)

A linha de pedido/status contém as informações essenciais que identificam a requisição, sendo elas:

- Método ou tipo do pedido, conforme Tabela 1.
- *Uniform Resource Locator* (URL), é o endereço virtual dos recursos na internet.
- Versão do protocolo.
- Código de status, presente somente na mensagem de reposta, sendo dividido em 5 classes: 1xx para respostas informacionais, 2xx indicam sucesso, 3xx destinados para redirecionamento, 4xx erro nas informações enviadas, e 5xx para erros do lado do servidor. Na Tabela 2 estão listados alguns status comuns de cada classe.

Tabela 1 – Métodos HTTP

Método	**Ação**
GET	Solicita a representação de um recurso específico
HEAD	Solicita da mesma forma que o GET porém sem o corpo da resposta
POST	Submete uma entidade a um recurso específico
PUT	Atualiza as representações do recurso pela carga da requisição
DELETE	Remove um recurso específico
CONNECT	Estabelece um túnel para o servidor identificado pelo recurso de destino
OPTIONS	Descreve as opções de comunicação com o recurso de destino
PATCH	Aplica modificações parciais em um recurso

Fonte: Adaptado de Mozilla (2023b)

Como próxima parte da mensagem se tem os cabeçalhos, que possuem como finalidade adicionar informações adicionais à requisição. Por exemplo: especificar o formato de retorno

Tabela 2 – Principais códigos de status HTTP

Código	**Frase**
100	*Continue*
102	*Processing*
200	*OK*
201	*Created*
204	*No Content*
301	*Moved Permanently*
303	*See Other*
400	*Bad Request*
401	*Unauthorized*
403	*Forbidden*
404	*Not Found*
500	*Internal Server Error*
502	*Bad Gateway*
503	*Service Unavailable*

Fonte: Adaptado de Mozilla (2023a)

desejado ou solicitar informações extras (Forouzan, 2010b). Eles são estruturados na forma de uma lista de chave/valor[2] separados por dois-pontos.

Por fim tem-se o corpo da requisição, que pode estar presente na mensagem de pedido e de resposta. Normalmente carregando as informações referentes à entidade que foi solicitada, enviada ou atualizada.

2.4.1.1 Segurança

Para que seja possível garantir a integridade e autenticidade dos dados entre cliente e servidor é utilizado um protocolo de criptografia chamado *Transport Layer Security* (TLS). Ele consiste que ambos os lados estejam de acordo com os protocolos de autenticação da entidade, autenticação de mensagens e de encriptação/decifração, sendo realizada através da criação de chaves de sessão (Barbosa *et al.*, 2020).

Para que seja considerada uma conexão segura são necessárias três etapas: autenticar o servidor para onde os dados serão enviados; garantir que o conteúdo enviado não foi modificado durante a transição; certificar-se que as informações sigilosas não foram interceptadas por um impostor (Forouzan, 2010b). O *Hypertext Transfer Protocol Secure* (HTTPS) é o nome adotado para o protocolo HTTP com utilização de TLS.

[2] É uma coleção em que cada chave é associada a um determinado valor, permitindo armazenar e acessar dados relacionados de forma mais eficiente.

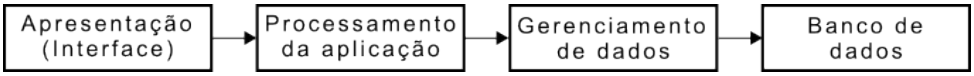
2.4.1.2 Aplicação em IoT

Com a atual infraestrutura de redes o HTTP é amplamente utilizado, sendo pouco provável que algum sistema de IoT não se utilize dele em algum nível da aplicação (Bziuk *et al.*, 2018). Mas apesar da sua facilidade de implantação ele pode não ser o mais adequado para todas as situações, por ser um protocolo que não tem como foco a utilização em dispositivos com limitação de recursos, como consumo de energia e largura de banda. Nas aplicações que essa limitação existe é interessante avaliar a utilização de outros protocolos, como por exemplo o *Message Queue Telemetry Transport* (MQTT).

2.5 DESENVOLVIMENTO WEB

O desenvolvimento web viabiliza o acesso seguro e remoto a um sistema por meio de um navegador, sem a necessidade da instalação de uma outra aplicação. No momento de sua criação é necessário que sejam utilizadas algumas boas práticas de desenvolvimento, assim criando uma robustez e consistência, facilitando a implementação de alterações e atualizações. Com essa finalidade, normalmente o sistema é dividido em quatro camadas, sendo elas mostradas na Figura 8 (Miletto; Bertagnolli, 2014).

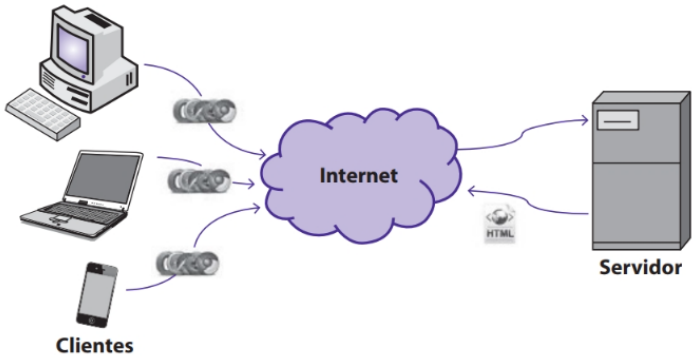
Figura 8 – Camadas de um sistema web



Fonte: Adaptado de Miletto e Bertagnolli (2014)

As aplicações web são hospedadas em um computador, denominado servidor, onde é realizado o processamento dos dados, que é acessado pelos clientes através da URL com o uso de um navegador (*browser*). Esse modelo é chamado de cliente-servidor, como exemplificado na Figura 9.

Figura 9 – Estrutura cliente-servidor



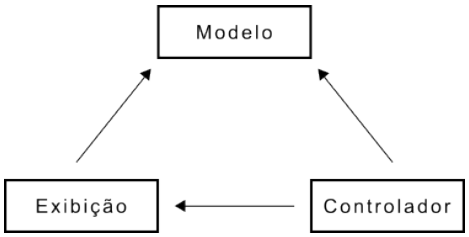
Fonte: Miletto e Bertagnolli (2014)

2.5.1 MVC

Entre os diversos padrões de desenvolvimento presentes no mercado um dos mais conhecidos é o *Model–View–Controller* (MVC), que unifica as camadas de apresentação e processamento em uma só, chamada de Exibição (*View*). Na Figura 10 é apresentado um diagrama da arquitetura e como cada uma de suas camadas se referenciam. Sendo que a divisão de responsabilidade delas é feita da seguinte forma:

- Modelo (*Model*): representa o estado do aplicativo e encapsula[3] qualquer lógica de negócio e lógica de implementação.
- Exibição (*View*): responsável por apresentar o conteúdo ao usuário por meio de uma interface.
- Controlador (*Controller*): trata a interação do usuário, utilizando o modelo e seleciona a exibição a ser renderizada.

Figura 10 – Arquitetura MVC



Fonte: Adaptado de Smith (2022)

Seus componentes possuem uma separação de interesses[4], o que permite uma escalabilidade mais fácil em termos de complexidade para codificar, depurar e testar. Já em situações que existe dependências entre duas ou mais dessas áreas acaba se tornando uma tarefa muito mais complicada de se executar. Por exemplo, uma aplicação em que a visualização e as regras de negócio estão combinadas em um único objeto, obriga que para ser feita uma mudança na visualização também seja feita na regra de negócio, o que muitas vezes acarreta em erros (Smith, 2022).

2.5.2 Banco de Dados

Desde os primórdios da humanidade, o ser humano sempre enfrentou a necessidade de registrar os eventos e as informações mais relevantes que eventualmente poderiam ser úteis no futuro, sendo que ao ser armazenada ela passa a ser um dado. O armazenamento pode ser

[3] Encapsulamento refere-se a limitação de acesso externo ao estado interno de um objeto.
[4] Separação de interesses (*separation of concerns*) é um principio de desenvolvimento que garante que o programa seja separado por tipos de trabalho a ser realizado.

realizado em papeis e de forma computadorizada, como disco rígido ou CD-ROM (Alvez, 2014). Como definição simples pode-se dizer que um banco de dados é um conjunto de dados com significado implícito. Para que seja possível gerenciar os conjuntos de dados surgiu o chamado Sistema de Gerenciamento de Banco de Dados (SGBD), que é uma coleção de ferramentas e programas que permitem a criação e manutenção de banco de dados pelo usuário (Alvez, 2014).

Os bancos podem ser classificados em quatro categorias: modelo relacional, modelo de entidade/relacionamento, modelo de dados semiestruturados e modelo de dados baseado em objetos (Silberschatz, 2020). O mais utilizado é o modelo relacional, sendo ele estruturado na forma de tabelas para representação dos dados e relação entre eles. Em cada uma das tabelas existe várias colunas com nome único e cada uma de suas linhas representa um pedaço da informação, a Tabela 3 apresenta um exemplo de um banco de dados com uma tabela.

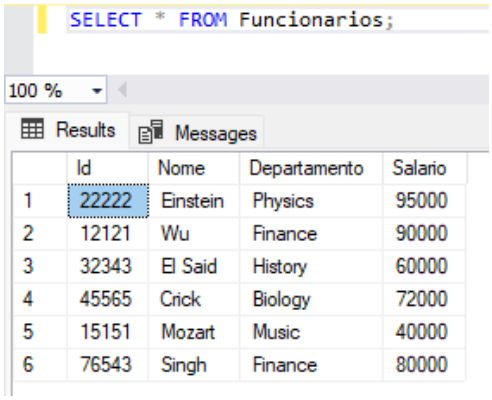
Tabela 3 – Exemplo banco relacional

ID	Nome	Departamento	Salario
22222	Einstein	Physics	95000
12121	Wu	Finance	90000
32343	El Said	History	60000
45565	Crick	Biology	72000
15151	Mozart	Music	40000
76543	Singh	Finance	80000

Fonte: Adaptado de Silberschatz (2020)

No ano de 1986 foi padronizada a linguagem *Structured Query Language* (SQL) para utilização em banco de dados pela *American National Standards Institute* (ANSI), e atualmente ela é amplamente utilizada pelos SGBD (Miletto; Bertagnolli, 2014). Através da SQL é possível criar *scripts* dos mais simples aos mais complexos, que permitem realizar qualquer interação com o banco de dados. Na Figura 11 é exemplificado como é feita uma consulta na tabela "Funcionarios"para recuperar os dados.

Figura 11 – Exemplo de consulta



Fonte: O Autor (2023)

3 METODOLOGIA

O sistema de manutenção proposto tem como objetivo permitir que todas as informações relacionadas aos equipamentos sejam centralizadas em um único lugar, permitindo a realização de análises de manutenção preditiva a partir dos dados coletados. Também possui um sistema de cadastro de alarmes e avisos que auxilia nas manutenções corretivas e preventivas dos equipamentos.

Neste capítulo será detalhado como foi desenvolvido o protótipo proposto, as tecnologias, *framework*, arquitetura e programações utilizadas.

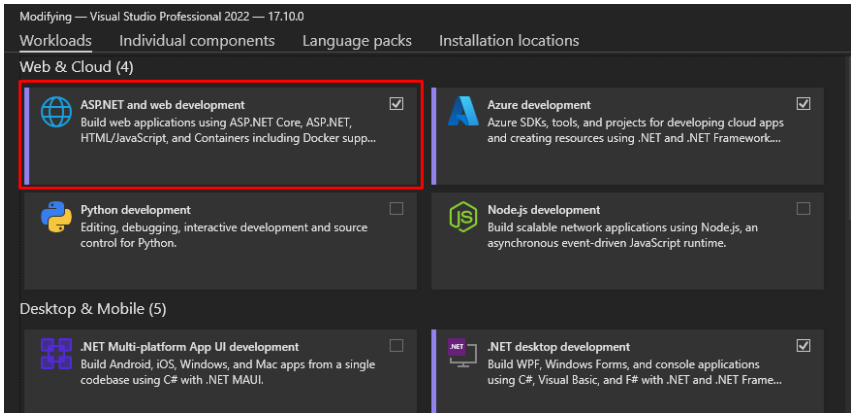
3.1 DESENVOLVIMENTO DA APLICAÇÃO

Para fins de testes e validação do protótipo, ele será simulado em um notebook com sistema operacional Windows. A aplicação foi desenvolvida utilizando a linguagem C#, na versão .NET 8.0 do *framework*. Para a arquitetura de desenvolvimento foi utilizado o MVC, apresentada na Seção 2.5.1.

3.1.1 Configuração do ambiente de desenvolvimento

Para ser possível iniciar o desenvolvimento foi escolhida como *Integrated Development Environment* (IDE) o Visual Studio 2022[1], essa já instala consigo as versões necessárias do *framework* da linguagem e disponibiliza também a instalação do *framework* para desenvolvimento web, Figura 12.

Figura 12 – Instalação *framework* desenvolvimento web



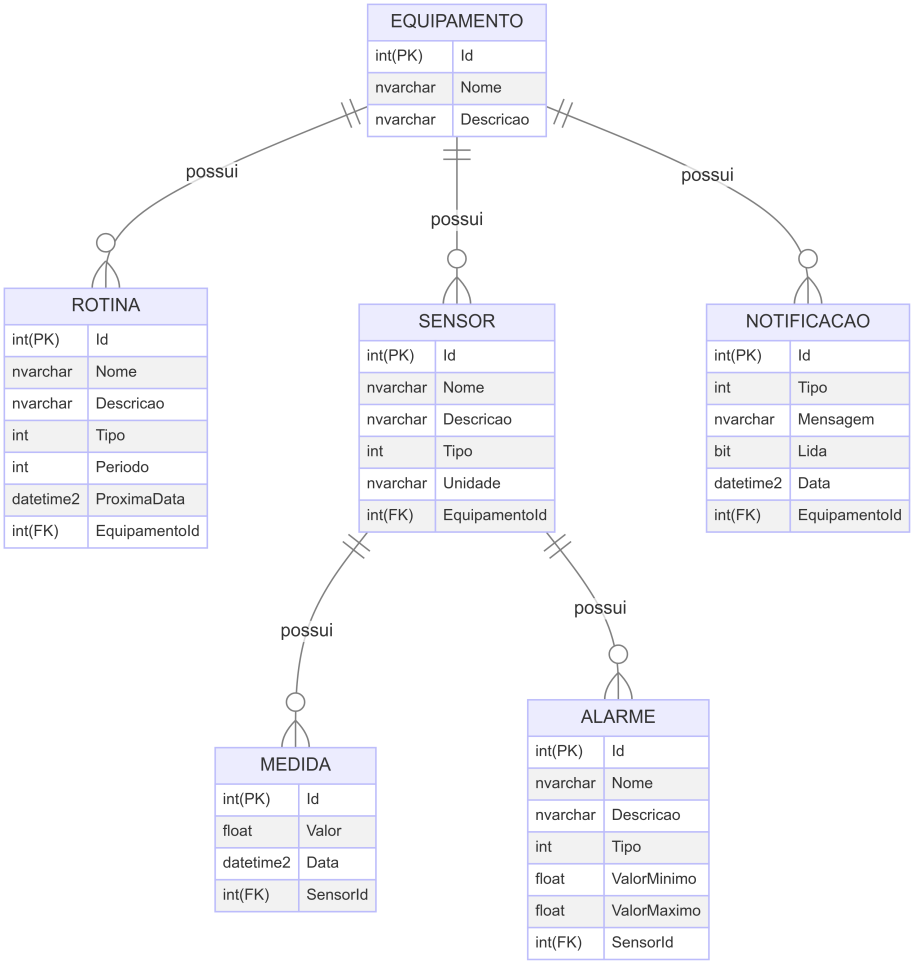
Fonte: O Autor (2024)

[1] Disponível em: https://visualstudio.microsoft.com/pt-br/downloads/. Acesso em 10 jan. 2024

3.1.2 Gerenciamento de Dados

O SQL Server Express foi selecionado como SGBD, para o gerenciamento dos dados. A escolha se sucedeu por ser uma opção gratuita e também pelo provedor da linguagem ser o mesmo, a Microsoft. Para que seja possível atender as necessidades do protótipo foi identificado a necessidade da criação de seis tabelas (Figura 13): quatro delas são informações cadastradas pelo usuário, uma para armazenar dados coletados e a última para notificações, detalhadas na Seção 3.1.2.1.

Figura 13 – Diagrama do modelo lógico do banco de dados



Fonte: O Autor (2024)

Na Figura 13 também está representada a relação existente entre as tabelas, conforme a notação apresentada na Figura 14, em que um equipamento pode possuir zero ou mais sensores, rotinas e notificações, mas cada um deles somente pode se relacionar com um equipamento. A mesma relação se aplica entre um sensor e suas medidas e alarmes.

Figura 14 – Notação da relação

Fonte: O Autor (2024)

3.1.2.1 Estrutura dos Dados

A primeira tabela apresentada representa o equipamento, em que pode ser cadastrado um nome para identificação e uma descrição para informações adicionais, as colunas dela podem ser visualizadas na Tabela 4 junto com os tipos de dados utilizados.

Tabela 4 – Equipamento

Coluna	**Tipo**
Id	int
Nome	nvarchar
Descrição	nvarchar

Fonte: O Autor (2024)

Na sequência tem-se a tabela *rotina*, utilizada para armazenar as rotinas de manutenção para cada equipamento. Cada rotina cadastrada pode ser de dois tipos: única ou periódica. A única irá gerar apenas uma notificação na data definida e a periódica irá gerar uma notificação a cada período de tempo. Na Tabela 5 é apresentado as colunas criadas para atingir esse objetivo, bem como os tipos utilizados.

Tabela 5 – Rotina

Coluna	**Tipo**
Id	int
Nome	nvarchar
Descricao	nvarchar
Tipo	int
Periodo	int
ProximaData	datetime2
EquipamentoId	int

Fonte: O Autor (2024)

A tabela *sensor* é utilizada para armazenar os sensores cadastrados de cada equipamento. Nela, da mesma forma que nas anteriores, é possível definir um nome e descrição. Também é possível definir se o sensor é do tipo digital (valor de 0 ou 1) ou analógico (valor numérico), bem como a unidade de medida do valor analógico. As suas colunas e tipos estão detalhadas na Tabela 6.

Para salvar os dados coletados pelo sensor foi criado a tabela *medida*. Na qual é armazenado o valor e data e hora do salvamento, sua estrutura pode ser observada na Tabela 7.

Tabela 6 – Sensor

Coluna	Tipo
Id	int
Nome	nvarchar
Descricao	nvarchar
Tipo	int
Unidade	nvarchar
EquipamentoId	int

Fonte: O Autor (2024)

Tabela 7 – Medida

Coluna	Tipo
Id	int
Valor	float
Data	datetime2
SensorId	int

Fonte: O Autor (2024)

Para ser possível a definição de alarmes para os sensores foi criada a tabela *alarme*. Nela é cadastrado um nome e uma descrição para o alarme, sendo possível definir um tipo. Os tipos disponíveis dependem do tipo do sensor, para sensores analógicos é disponibilizado alarmes do tipo: "menor que", "maior que" e "dentro do intervalo". Já para os sensores digitais tem-se: "sinal ligou" e "sinal desligou". As colunas e tipos podem ser visualizados na Tabela 8.

Tabela 8 – Alarme

Coluna	Tipo
Id	int
Nome	nvarchar
Descricao	nvarchar
Tipo	int
ValorMinimo	float
ValorMaximo	float
SensorId	int

Fonte: O Autor (2024)

Por fim tem-se a tabela *notificação*, ela é responsável por armazenar as notificações geradas por rotinas e por alarmes. Podendo ser de dois tipos: informativa ou aviso. As informativas são geradas pelas rotinas e os avisos pelos alarmes cadastrados. Já as notificações geradas possuem uma mensagem, um status de leitura e a data em que foram criadas. Sua estrutura está detalhada na Tabela 9.

Tabela 9 – Notificação

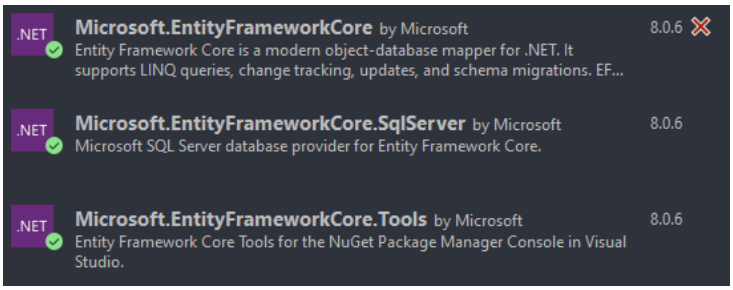
Coluna	Tipo
Id	int
Tipo	int
Mensagem	nvarchar
Lida	bit
Data	datetime2
EquipamentoId	int

Fonte: O Autor (2024)

3.1.2.2 Acesso ao banco de dados

Como meio de acesso ao banco de dados foi utilizado o *Entity Framework* (EF), empregando a abordagem *Code First*. Essa abordagem possibilita a criação dos modelos de dados a partir de classes da linguagem, nesse caso o C#. Então o EF utiliza essas definições para criar o banco de dados correspondente. Para utilização desse *framework* foi necessário a instalação de três pacotes, dois gerais e um específico para o SGBD utilizado. Esses pacotes são apresentados na Figura 15.

Figura 15 – Pacotes necessários para o *Entity Framework*



Fonte: O Autor (2024)

Na sequência foi necessário configurar a conexão com o banco, para isso foi utilizado o usuário padrão "sa". Então criou-se a *string* de conexão como uma variável de ambiente no arquivo de inicialização da aplicação, como mostrado na Figura 16, e adicionado por meio de injeção de dependência à aplicação, conforme a Figura 17.

Figura 16 – *String* de conexão com o banco

```
"https": {
  "commandName": "Project",
  "dotnetRunMessages": true,
  "launchBrowser": true,
  "applicationUrl": "https://localhost:7040;http://localhost:5284",
  "environmentVariables": {
    "ASPNETCORE_ENVIRONMENT": "Development",
    "SqlConnectionString": "Server=localhost\\SQLEXPRESS;Database=maintenancedb;User Id=sa;Password=e6TT5V#D==s6;TrustServerCertificate=True;"
  }
},
```

Fonte: O Autor (2024)

Após, foram criados os modelos de dados, conforme especificado na Seção 3.1.2.1, e uma classe para servir de contexto para o EF. A classe de contexto, dentro do EF, é a responsável

Figura 17 – Injeção de dependência da *string* de conexão

```
services.AddDbContext<ManagementContext>( optionsAction: options =>
{
    options.UseSqlServer( connectionString: Environment.GetEnvironmentVariable("SqlConnectionString"));
});
```

Fonte: O Autor (2024)

por criar uma conexão com o banco e, é através dela que todos as interações serão realizadas. Na Figura 18 é possível visualizar a classe criada.

Figura 18 – Classe de contexto

```
public class ManagementContext : DbContext
{
    7 references | 0 changes | 0 authors, 0 changes
    public DbSet<Alarme> Alarmes { get; set; }
    10 references | 0 changes | 0 authors, 0 changes
    public DbSet<Equipamento> Equipamentos { get; set; }
    5 references | 0 changes | 0 authors, 0 changes
    public DbSet<Medida> Medidas { get; set; }
    8 references | 0 changes | 0 authors, 0 changes
    public DbSet<Notificacao> Notificacoes { get; set; }
    7 references | 0 changes | 0 authors, 0 changes
    public DbSet<Rotina> Rotinas { get; set; }
    10 references | 0 changes | 0 authors, 0 changes
    public DbSet<Sensor> Sensores { get; set; }
```

Fonte: O Autor (2024)

Com o contexto criado foi necessário realizar a migração, através do comando "Add-Migration ’nome da migração’" no *Package Manager Console* da IDE. Ela gerou os arquivos necessários, a partir dos modelos, para realizar a criação das tabelas no banco. E por fim é realizada a atualização do banco pelo comando "Update-Database".

3.1.3 Back-end

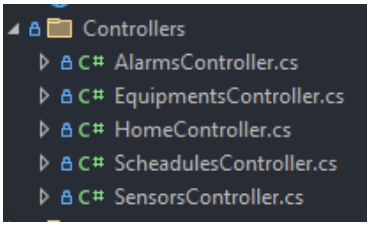
Esta seção apresenta o desenvolvimento realizado no *back-end* da aplicação, como foram estruturados os controladores da arquitetura MVC (responsáveis pela criação dos *end-points*), e os serviços responsáveis pela interação com o banco de dados através do contexto apresentado na Seção 3.1.2.2.

3.1.3.1 Controladores

O desenvolvimento se iniciou com a divisão das responsabilidades por controlador, sendo criado um para gerir a página inicial e um para cada um dos itens cadastráveis pelos usuários, sendo eles: equipamentos, rotinas, sensores e alarmes. Para o correto funcionamento da aplicação os controladores precisam ser criados dentro da pasta "Controllers" e devem conter o sufixo *Controller* em seu nome, como apresentado na Figura 19.

Para as rotas optou-se por manter o caminho padrão, que consiste no nome do controlador, sem o sufixo *Controller*, seguido pelo nome da ação e por fim um ID opcional, sendo

Figura 19 – Controladores criados



Fonte: O Autor (2024)

esses divididos por uma "/". A configuração pode ser visualizada na Figura 20, em que a ação é qualquer método público criado dentro do controlador e o ID é passado como parâmetro desse método. Na Tabela 10 estão detalhados todos os *endpoints* de cada controlador, especificando o método HTTP utilizado e uma descrição da sua funcionalidade.

Figura 20 – Definição das rotas

```
app.MapControllerRoute(
    name: "default",
    pattern: "{controller=Home}/{action=Index}/{id?}");
```

Fonte: O Autor (2024)

3.1.3.2 Serviços

Nesta aplicação foi optado pela criação de serviços para acesso a banco, assim separando as responsabilidades entre serviço e controlador, de forma a melhorar a expansibilidade e facilitar manutenções futuras.

Para cada uma das tabelas criadas foi criado um serviço, em que está definido as regras de consulta, inserção, atualização e remoção. Dessa forma, as regras ficam centralizadas e evita a replicação de código, na Figura 21 é exemplificado algumas lógicas que são realizadas. Para facilitar o acesso aos serviços, eles foram adicionados por meio de injeção de dependência.

Figura 21 – Exemplo de lógicas realizadas nos serviços

```
public async Task<Equipamento> InsertAsync(Equipamento equipamento)
{
    _context.Equipamentos.Add(equipamento);
    await _context.SaveChangesAsync();
    return equipamento;
}


public async Task<Equipamento> UpdateAsync(Equipamento equipamento)
{
    var existingEquipment:Equipamento = await _context.Equipamentos.FindAsync(equipamento.Id);
    if (existingEquipment == null)
        return null;

    existingEquipment.Nome = equipamento.Nome;
    existingEquipment.Descricao = equipamento.Descricao;

    _context.Equipamentos.Update(existingEquipment);
    await _context.SaveChangesAsync();
    return existingEquipment;
}
```

Fonte: O Autor (2024)

Tabela 10 – *Endpoints* da aplicação

Controlador	Método	URL relativa	Descrição
Home	GET	/Home/Index	Retorna o HTML da página principal
	GET	/Home/Table	Retorna a tabela de medições
	POST	/Home/Read/<id>	Marca a notificação como lida
	GET	/Home/Download	Retorna um arquivo CSV com as medições
Equipments	GET	/Equipments/Index	Retorna a exibição com todos os equipamentos
	GET	/Equipments/Create	Retorna a exibição de cadastro de equipamento
	POST	/Equipments/Create	Insere o novo equipamento no banco
	GET	/Equipments/Edit/<id>	Retorna a exibição de edição de equipamento
	POST	/Equipments/Edit	Atualiza o equipamento no banco
	DELETE	/Equipments/Delete/<id>	Deleta o equipamento no banco
Scheadules	GET	/Scheadules/Index/<id>	Retorna a exibição com as rotinas do equipamento
	GET	/Scheadules/Create/<id>	Retorna a exibição de cadastro de rotina
	POST	/Scheadules/Create	Insere a nova rotina no banco
	GET	/Scheadules/Edit/<id>	Retorna a exibição de edição de rotina
	POST	/Scheadules/Edit	Atualiza a rotina no banco
	DELETE	/Scheadules/Delete/<id>	Deleta a rotina no banco
Sensors	GET	/Sensors/Index/<id>	Retorna a exibição com os sensores do equipamento
	GET	/Sensors/Create/<id>	Retorna a exibição de cadastro de sensor
	POST	/Sensors/Create	Insere o novo sensor no banco
	GET	/Sensors/Edit/<id>	Retorna a exibição de edição de sensor
	POST	/Sensors/Edit	Atualiza o sensor no banco
	DELETE	/Sensors/Delete/<id>	Deleta o sensor no banco
	POST	/Sensors/Measure	Insere a nova medida no banco
Alarms	GET	/Alarms/Index/<id>	Retorna a exibição com os alarmes do sensor
	GET	/Alarms/Create/<id>	Retorna a exibição de cadastro de alarme
	POST	/Alarms/Create	Insere o novo alarme no banco
	GET	/Alarms/Edit/<id>	Retorna a exibição de edição de alarme
	POST	/Alarms/Edit	Atualiza o alarme no banco
	DELETE	/Alarms/Delete/<id>	Deleta o alarme no banco

Fonte: O Autor (2024)

Para que fosse possível validar diariamente as rotinas cadastradas e lançar as notificações de forma correta houve a necessidade da criação de um serviço a parte que execute as validações de tempo em tempo, independente do web site estar aberto ou não. Para isso foi escolhido criar uma *Azure Function*, uma solução que independe de um servidor, assim reduzindo custos na implantação. Nela foi configurado um gatilho, conforme Figura 22, para que seja executada uma vez ao dia, verificando todas rotinas cadastradas e criando a notificação caso necessário.

Figura 22 – Configuração do gatilho da função

```
[Function(name: "VerificarRotinas")]
1 reference | 0 changes | 0 authors, 0 changes
public void Verify([TimerTrigger(schedule: "0 0 5 * * *")]TimerInfo myTimer)
```

Fonte: O Autor (2024)

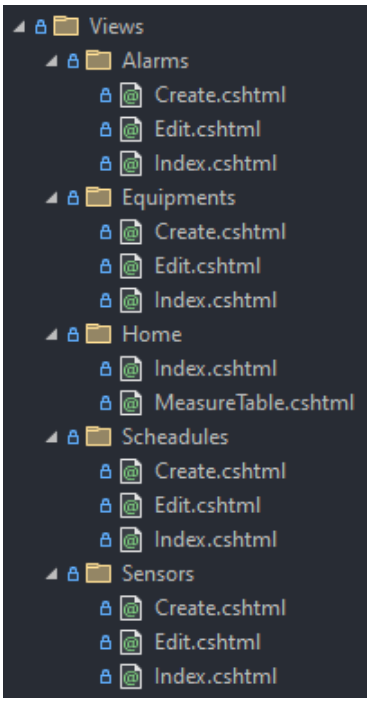
3.1.4 Front-end

Esta seção aborda o desenvolvimento do *front-end* da aplicação, detalhando como as exibições foram criadas e suas funcionalidades.

3.1.4.1 Exibições

Para criação e organização das exibições foi seguido o padrão da arquitetura. Em que é necessário criar, dentro da pasta "Views", uma subpasta que possua o mesmo nome do controlador relacionado sem o sufixo *Controller*. Na Figura 23 é apresentado como foi criada a estrutura de exibições, seguindo o padrão da nomenclatura e dos controladores detalhados na Seção 3.1.3.1.

Figura 23 – Exibições criadas



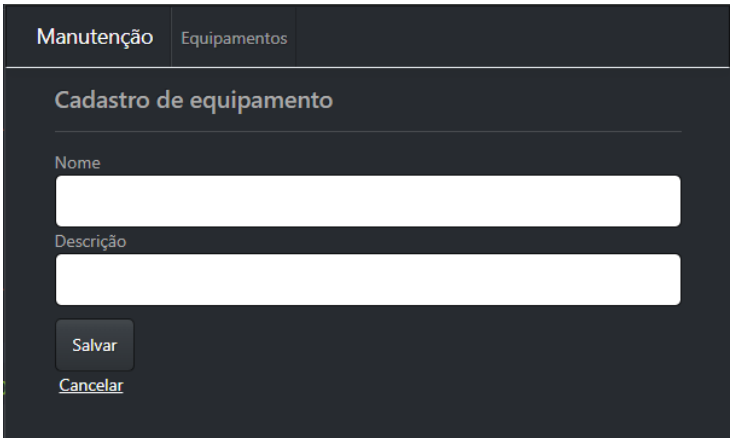
Fonte: O Autor (2024)

3.1.4.2 Telas de Cadastro

A aplicação possui um total de quatro telas de cadastros que são acessadas através de um navegador web. Todas elas possuem a mesma identidade visual e disposição dos campos, sendo que os campos foram criados para os tipos de dados definidos, facilitando o preenchimento.

A primeira é o cadastro do equipamento, apresentada na Figura 24. Essa permite a atribuição de um nome para identificação do equipamento, e um campo de descrição com a finalidade de armazenar informações adicionais.

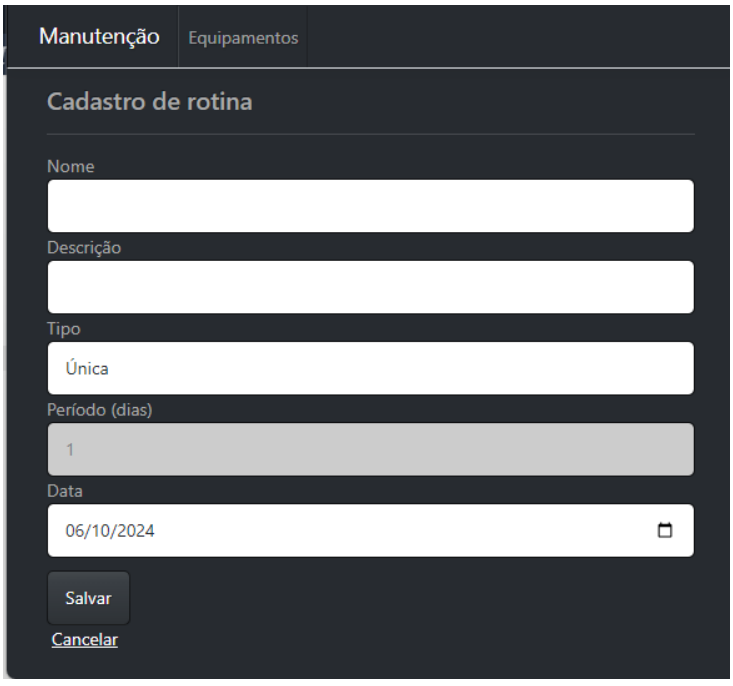
Figura 24 – Tela de cadastro de equipamento



Fonte: O Autor (2024)

A segunda é o cadastro de rotina, conforme a Figura 25. Da mesma forma que na de equipamento é possível definir um nome e descrição. Além disso, pode-se selecionar o tipo da rotina entre "Única" ou "Periódica". Ao selecionar o tipo "Única"o campo de data fica habilitado para que ela seja definida. Quando for selecionada o tipo "Periódica" o campo data fica desabilitado e é possível definir o período de dias que a notificação será lançada.

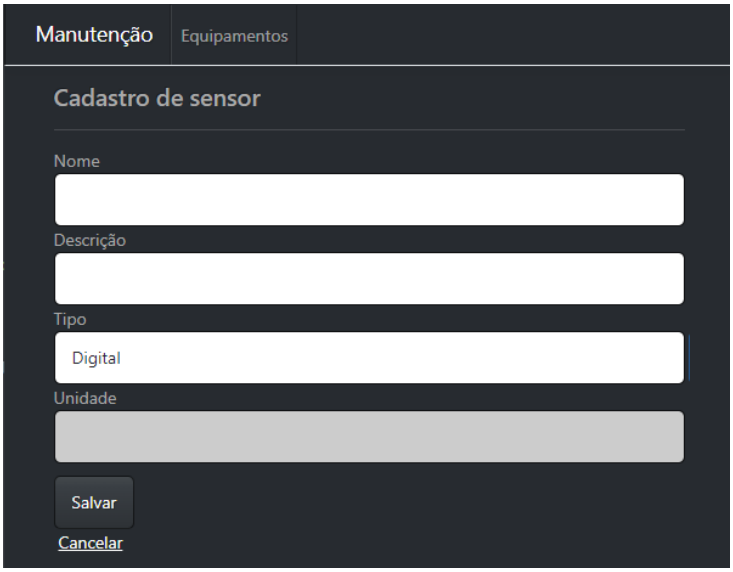
Figura 25 – Tela de cadastro de rotina



Fonte: O Autor (2024)

A terceira tela é o cadastro de sensor, que pode ser visualizada na Figura 26. Nela é possível atribuir um nome e descrição adicional ao sensor, também é permitida a seleção entre o tipo "Digital" e "Analógico". Sendo que, para os sensores do tipo analógico é habilitado o campo para definir a unidade de medida a ser utilizada.

Figura 26 – Tela de cadastro de sensor



Fonte: O Autor (2024)

Por fim tem-se a tela de cadastro de alarme, conforme a Figura 27. Como nas anteriores ela permite o cadastro de um nome e descrição, já os tipos disponíveis depende do tipo de sensor utilizado. Para os sensores analógicos é disponibilizado os tipo Intervalo, Menor que e Maior que. Para os analógicos também são exibidos os campos "Valor mínimo" e "Valor máximo", esses são habilitados e desabilitados de acordo com o tipo selecionado. Já para os sensores digitais somente é apresentado o tipo, podendo ser "Sinal ligou"ou "Sinal desligou".

Figura 27 – Tela de cadastro de alarme

Fonte: O Autor (2024)

3.1.4.3 Telas de Visualização

Para as telas de visualização dos itens cadastrados foi optado pelo uso de *cards*, pois através deles é possível apresentar as informações de forma clara e organizada. Dentro deles estão contidas todas as informações adicionadas no momento do cadastro, além de possuírem botões para deleção, edição e visualização dos itens filhos. Por itens filhos entende-se os itens cadastráveis que possuem uma relação de zero ou mais, conforme a definição apresentada na Figura 13.

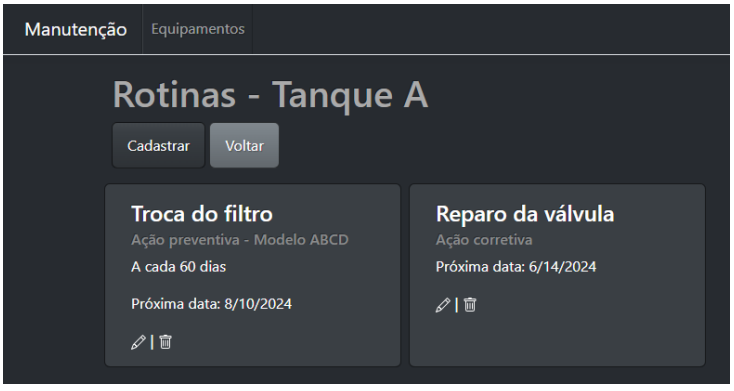
Na tela inicial de equipamentos, conforme Figura 28. Está disponibilizado um botão para cadastro de novos equipamentos e em cada item já cadastrado é apresentado um *card* com à informações inseridas. Além disso, são criados campos para cada tipo de item filho com a quantidade associada ao equipamento, sendo que ao clicar em cada um desses campos o usuário é direcionado à tela de visualização respectiva.

Figura 28 – Tela de visualização de equipamentos

Fonte: O Autor (2024)

Nas telas de itens filhos é adicionado no título o nome do item pai, e também é criado um botão extra para direcionar o usuário de volta à tela de visualização do item pai, como pode ser observado na Figura 29.

Figura 29 – Tela de visualização de rotinas



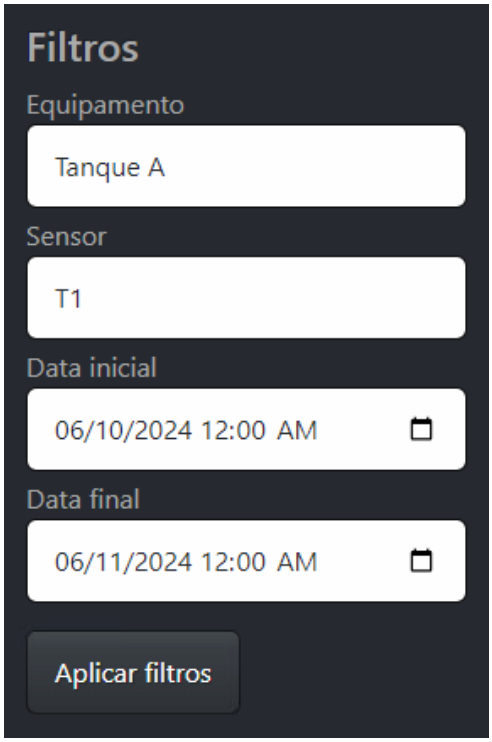
Fonte: O Autor (2024)

3.1.4.4 Tela Principal

A tela principal tem como objetivo permitir que o usuário visualize as medidas coletadas de cada sensor e apresentar as notificações de alarmes e rotinas. Para melhor organização da tela ela foi segmentada em três partes.

A primeira delas, localizada na parte esquerda da tela, é responsável pela definição de filtros dos dados apresentados. Nela é possível definir o equipamento e o sensor que o usuário gostaria de visualizar os dados. Também é configurável o período de tempo em que os dados devem ser apresentados. A estrutura de campos pode ser vista na Figura 30.

Figura 30 – Filtros dos dados



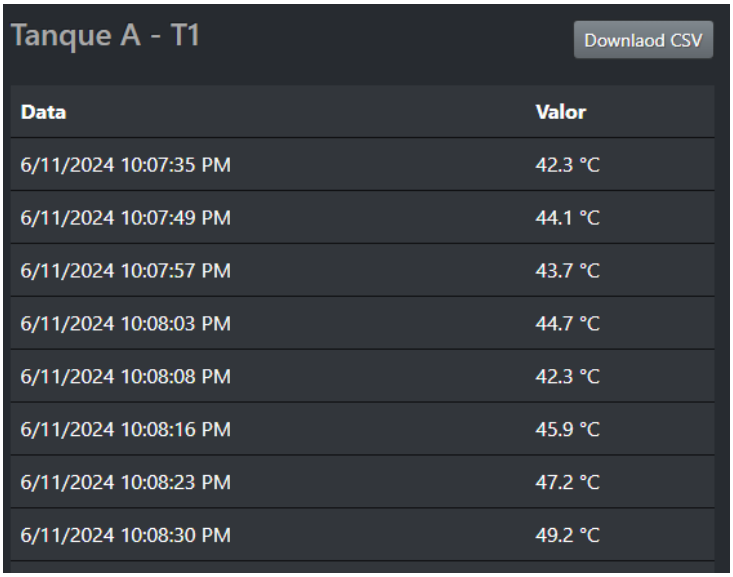
Fonte: O Autor (2024)

Por segundo tem-se a exibição dos dados, conforme Figura 31. Essa parte é responsável por apresentar os dados filtrados ao usuário, no formato de tabela. Entendendo que somente por meio de uma tabela o usuário não conseguiria extrair todas as informações que ele necessita, foi criado uma função para exportar os dados coletados para um arquivo CSV[2]. Permitindo assim análises mais aprofundadas com os dados a partir de outro software, como o Excel.

Por fim são apresentadas as notificações, na parte direita da tela. Em que a visualização é dividida em duas: informações e avisos, como pode ser observado na Figura 32. Sendo que as informações são geradas por rotinas e os avisos por alarmes. Também é possível que o usuário oculte após realizar a leitura, evitando assim o acumulo de notificações.

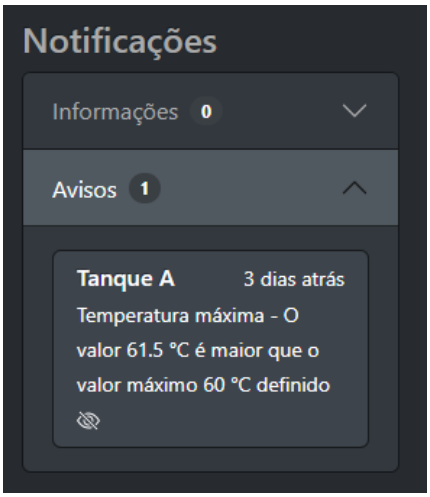
[2] Arquivo de texto com valores separados por vírgula

Figura 31 – Exibição dos dados



Fonte: O Autor (2024)

Figura 32 – Notificações



Fonte: O Autor (2024)

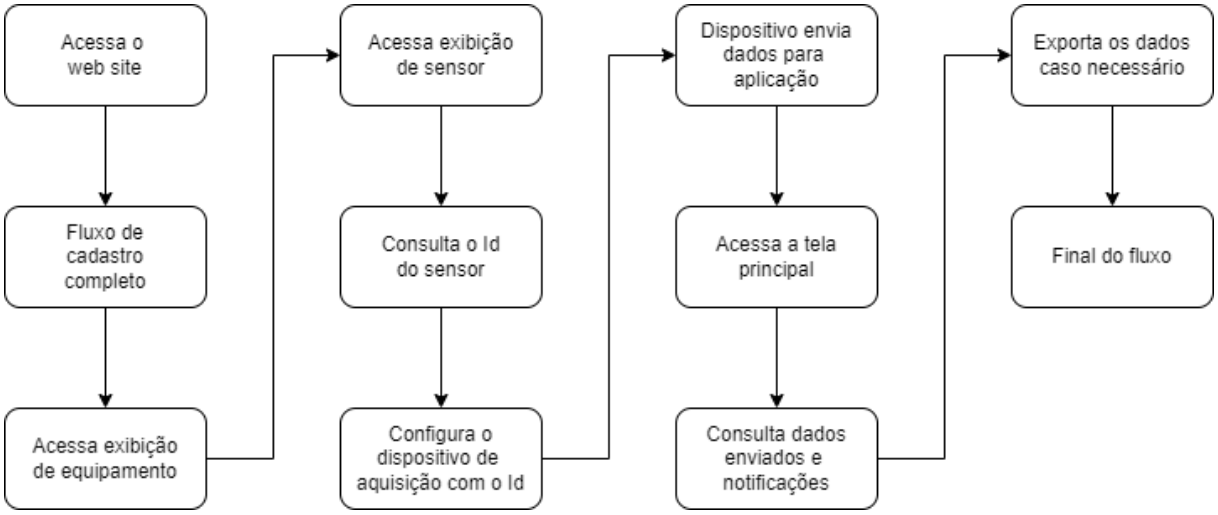
3.2 FLUXO DA APLICAÇÃO

Esta seção contempla todos os passos necessários para utilização do protótipo proposto neste trabalho. Sendo ele dividido em duas partes principais, cadastros e aquisição de dados. Na Figura 33 é exibido o fluxo geral da aplicação, e as etapas serão detalhadas na sequência.

3.2.1 Cadastros

O fluxo de cadastro inicia-se ao acessar o web site na exibição de equipamentos, para isso basta clicar no botão "Equipamento" na barra de navegação superior. Nessa tela o usuário deve cadastrar os seus equipamentos, para então poder prosseguir com os cadastros subsequen-

Figura 33 – Fluxo geral da aplicação



Fonte: O Autor (2024)

tes.

A partir de um equipamento o usuário pode seguir em duas direções, o cadastro de uma rotina e/ou cadastro de sensores. Acessando a tela de rotinas é possível realizar o cadastro de quantas desejar, em que todas as rotinas cadastradas serão automaticamente vinculadas ao equipamento, o mesmo comportamento é aplicado aos sensores. Adicionalmente no cadastro de sensores é possível acessar a exibição de alarmes de cada sensor, onde o usuário pode criar quantos achar necessário. Esse fluxo é apresentado na Figura 34 no formato de fluxograma, para melhor entendimento.

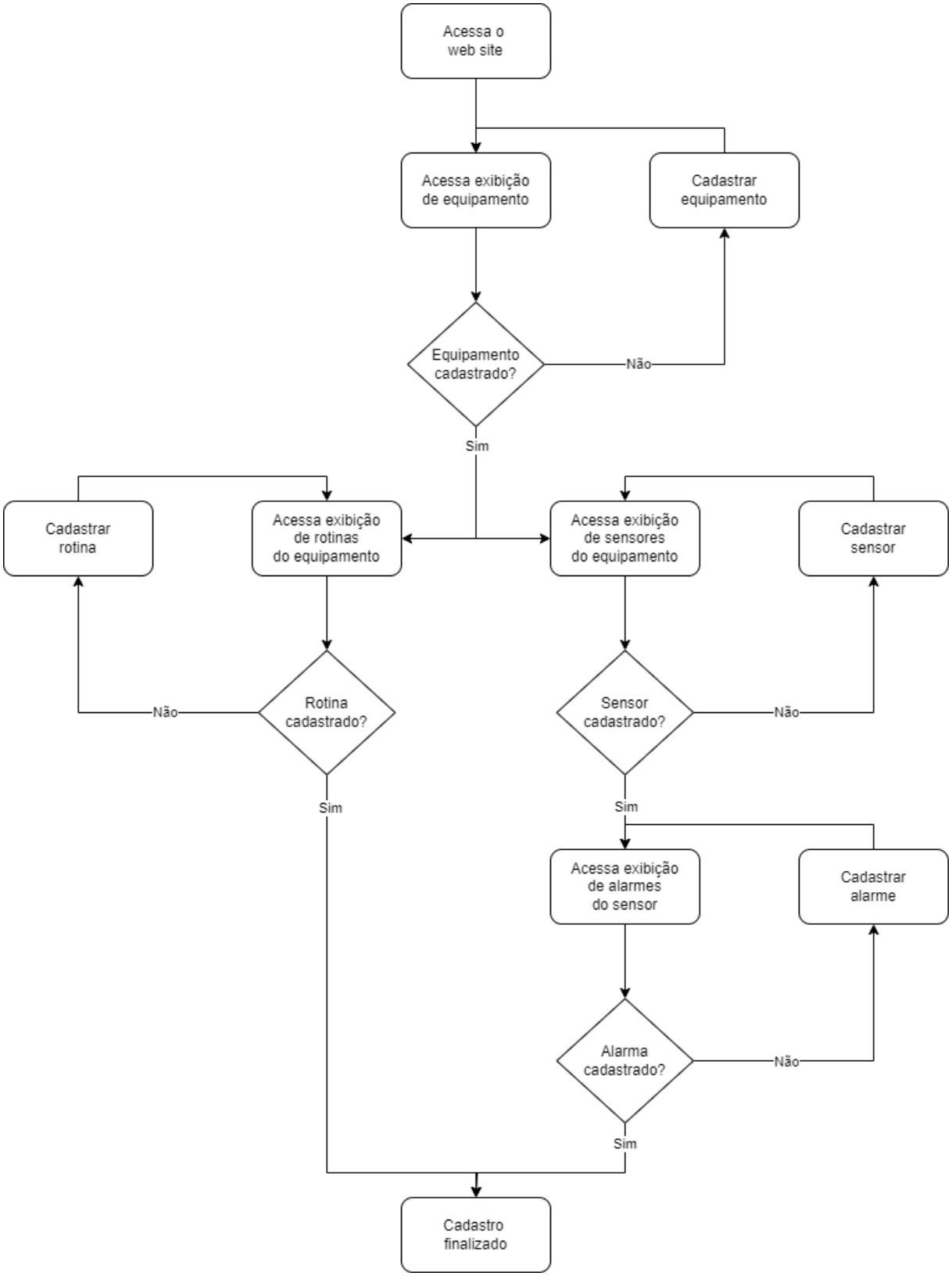
3.2.2 Aquisição de Dados

Para realizar aquisição e envio de dados, o usuário primeiramente deve buscar no site o Id gerado de cada sensor. Essa informação é encontrada no *card* do sensor cadastrado, sendo o valor numérico que se encontra ao lado do nome, conforme Figura 35.

Em seguida, é necessário configurar o corpo da requisição que será enviada. O corpo deve estar no formato *JavaScript Object Notation* (JSON) e conter o Id do sensor e valor a ser inserido, conforme exemplo na Figura 36. Como *endpoint* deve ser usado a ação *Measure* do controlador *Sensors* apresentado na Tabela 10. O dispositivo de aquisição também possui como responsabilidade o controle do tempo entre cada envio.

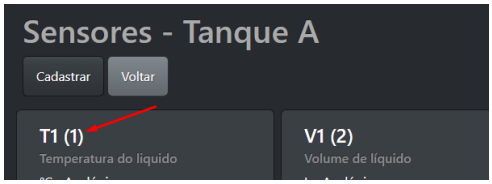
Ao receber o dado a aplicação irá inseri-lo no banco com a data e hora atual. Também será realizada a validação se o valor inserido atende às condições de algum alarme, gerando assim uma notificação de aviso para o equipamento. Após isso os dados já ficam disponíveis na tela principal para que sejam consultados e exportados.

Figura 34 – Fluxo de cadastro



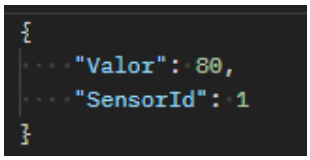
Fonte: O Autor (2024)

Figura 35 – Localização do Id do sensor



Fonte: O Autor (2024)

Figura 36 – Corpo da requisição para inserção de medida



Fonte: O Autor (2024)

3.3 TESTES DA APLICAÇÃO

Para testes da aplicação foi utilizado um notebook Dell Latitude 3420 como servidor. Por primeiro foi instalado e configurado o acesso ao banco de dados, e na sequência foi configurada a hospedagem da aplicação utilizando o *Internet Information Services* (IIS)[3]. Além disso foi necessário realizar a liberação da porta usada pela aplicação, porta 7040, no *firewall* do Windows de forma a permitir conexão de outros dispositivos na mesma rede.

Com o site já hospedado e comunicando com o banco foi realizado o cadastro de um equipamento. Foi criada uma rotina, um sensor e um alarme, para que fosse possível validar todo o fluxo da aplicação.

Como dispositivo de aquisição foi optado pelo uso do microcontrolador ESP32, por já possuir conexão Wi-Fi, e em conjunto foi utilizado um sensor de temperatura DHT11. O microcontrolador foi programado, utilizando a Arduino IDE, para realizar a leitura da temperatura por uma de suas portas analógicas e enviar o dado coletado para a aplicação através da rede Wi-Fi.

Para testes foi definido um intervalo de aquisição de um minuto entre medidas. Durante o período de aquisição foi forçado o aumento e diminuição de temperatura próxima ao sensor para que houvesse uma variação nas medições, permitindo assim a validação do funcionamento dos alarme.

Como último teste foi realizada a validação do serviço de validação de rotinas. Para fins de testes de validação da lógica foi ajustado o gatilho do serviço, de forma que ele execute a cada hora ao invés de diariamente, assim permitindo os testes pontuais.

[3] É um servidor web criado pela Microsoft para os seus sistemas operacionais.

4 CONCLUSÕES

O desenvolvimento deste protótipo de sistema de manutenção representa um passo importante na busca por soluções que auxiliem na gestão de ativos e na otimização da eficiência operacional em diversos setores industriais. Através da coleta de dados automatizada, cadastro de rotinas de manutenção e a geração de alarmes para eventos críticos, é oferecido um suporte abrangente para as atividades corretivas, preventivas e preditivas.

A partir dos tipos de dados definidos (analógicos e digitais), foram desenvolvidos cadastros de equipamentos, rotinas, sensores e alarmes, todos armazenados em banco de dados. Além de ter sido desenvolvida também uma tela principal para monitoramento de dados coletados e notificações. Essa permite que os dados sejam exportados, possibilitando análises mais aprofundadas.

O presente trabalho tem como limitação o desenvolvimento de um protótipo de sistema de gerenciamento de manutenção e se mostrou eficiente para o seu objetivo. Disponibilizando meios para organização de manutenções preventivas através das rotinas, avisos de problemas em equipamentos por meio dos alarmes e, por fim, auxilia em processos de manutenção preditiva ao disponibilizar os dados que foram centralizados na aplicação.

O protótipo foi desenvolvido utilizando tabelas para exibição dos dados sendo o suficiente para alcançar os objetivos propostos. Porém entende-se que a representação por meio de gráficos pode facilitar algumas análises e comparações entre equipamento, o que sugere uma possibilidade de trabalho futuro para melhoria do modo de visualização dos dados. O trabalho também se limitou a não realizar análises nos dados para previsão de falhas, o que sugere outra possibilidade de trabalho futuro. A simulação do protótipo foi realizada de forma local, permitindo um trabalho futuro, relacionado a hospedar a aplicação em um servidor externo, sendo necessário o desenvolvimento de permissões de acesso e validações de segurança para as requisições.

REFERÊNCIAS

ADEBAYO, A. O.; CHAUBEY, M. S.; NUMBU, L. P. Industry 4.0: The fourth industrial revolution and how it relates to the application of internet of things (iot). **Journal of Multidisciplinary Engineering Science Studies (JMESS)**, v. 5, n. 2, p. 2477–2482, 2019.

ALVEZ, W. P. **Banco de Dados**. São Paulo: Grupo A, 2014. ISBN 9788536518961. Disponível em: <https://integrada.minhabiblioteca.com.br/#/books/9788536518961/>. Acesso em: 25 nov. 2023.

BALBINOT, A.; BRUSAMARELLO, V. J. **Instrumentação e Fundamentos de Medidas**. Rio de Janeiro: Grupo GEN, 2019. v. 1. ISBN 9788521635864. Disponível em: <https://integrada.minhabiblioteca.com.br/#/books/9788521635864/>. Acesso em: 16 out. 2023.

BARBOSA, C. S. *et al.* **Arquitetura TCP/IP I**. Porto Alegre: Grupo A, 2020. ISBN 9786556900766. Disponível em: <https://integrada.minhabiblioteca.com.br/#/books/9786556900766/>. Acesso em: 11 nov. 2023.

BZIUK, W. *et al.* On http performance in iot applications: An analysis of latency and throughput. In: **2018 41st International Convention on Information and Communication Technology, Electronics and Microelectronics (MIPRO)**. [S.l.: s.n.], 2018. p. 0350–0355.

ENGEMAN. Manutenção: tipos e tendências. 2023. Disponível em: <https://blog.engeman.com.br/manutencao-tipos-e-tendencias/>. Acesso em: 26 nov. 2023.

FOROUZAN, B. A. **Comunicação de dados e redes de computadores**. 4. ed. São Paulo: Grupo A, 2010. ISBN 9788563308474. Disponível em: <https://integrada.minhabiblioteca.com.br/#/books/9788563308474/>. Acesso em: 04 nov. 2023.

____. **Protocolo TCP/IP**. São Paulo: Grupo A, 2010. ISBN 9788563308689. Disponível em: <https://integrada.minhabiblioteca.com.br/#/books/9788563308689/>. Acesso em: 05 nov. 2023.

GREGÓRIO, G. F. P.; SILVEIRA, A. M. d. **Manutenção industrial**. São Paulo: Grupo A, 2018. ISBN 9788595026971. Disponível em: <https://integrada.minhabiblioteca.com.br/#/books/9788595026971/>. Acesso em: 26 nov. 2023.

LATHI, B. P. **Sinais e sistemas lineares**. 2. ed. São Paulo: Grupo A, 2006. ISBN 9788577803910. Disponível em: <https://integrada.minhabiblioteca.com.br/#/books/9788577803910/>. Acesso em: 20 out. 2023.

LENZ, M. L.; MORAES, M. L. **Eletrônica digital**. Porto Alegre: Grupo A, 2019. ISBN 9788595028579. Disponível em: <https://integrada.minhabiblioteca.com.br/#/books/9788595028579/>. Acesso em: 28 out. 2023.

MILETTO, E. M.; BERTAGNOLLI, S. C. **Desenvolvimento de software II: introdução ao desenvolvimento web com HTML, CSS, javascript e PHP**. São Paulo: Grupo A, 2014. ISBN 9788582601969. Disponível em: <https://integrada.minhabiblioteca.com.br/#/books/9788582601969/>. Acesso em: 24 nov. 2023.

MOZILLA. Códigos de status de respostas http. 2023. Disponível em: <https://developer.mozilla.org/pt-BR/docs/Web/HTTP/Status/>. Acesso em: 26 nov. 2023.

____. Métodos de requisição http. 2023. Disponível em: <https://developer.mozilla.org/pt-BR/docs/Web/HTTP/Methods/>. Acesso em: 26 nov. 2023.

OPPENHEIM, A. V.; WILLSKY, A. S. **Sinais e sistemas**. 2. ed. São Paulo: Pearson, 2010. Disponível em: <https://plataforma.bvirtual.com.br>. Acesso em: 20 out. 2023.

PAHLAVAN, K.; KRISHNAMURTHY, P. Evolution and impact of wi-fi technology and applications: A historical perspective. **International Journal of Wireless Information Networks**, v. 28, n. 1, p. 3–19, Mar 2021. ISSN 1572-8129. Disponível em: <https://doi.org/10.1007/s10776-020-00501-8>.

ROCHOL, J. **Sistemas de comunicação sem fio**. São Paulo: Grupo A, 2018. ISBN 9788582604564. Disponível em: <https://integrada.minhabiblioteca.com.br/#/books/9788582604564/>. Acesso em: 03 nov. 2023.

SHANKAR, L.; SINGH, C. D.; SINGH, R. Impact of implementation of cmms for enhancing the performance of manufacturing industries. **International Journal of System Assurance Engineering and Management**, v. 14, n. 5, p. 1599–1620, Oct 2023. ISSN 0976-4348. Disponível em: <https://doi.org/10.1007/s13198-021-01480-6>. Acesso em: 12 out. 2023.

SILBERSCHATZ, A. **Sistema de Banco de Dados**. São Paulo: Grupo GEN, 2020. ISBN 9788595157552. Disponível em: <https://integrada.minhabiblioteca.com.br/#/books/9788595157552/>. Acesso em: 25 nov. 2023.

SMITH, S. Overview of asp.net core mvc. 2022. Disponível em: <https://learn.microsoft.com/en-us/aspnet/core/mvc/overview>. Acesso em: 24 nov. 2023.

WUKKADADA, B. *et al.* Comparison with http and mqtt in internet of things (iot). In: **2018 International Conference on Inventive Research in Computing Applications (ICIRCA)**. [S.l.: s.n.], 2018. p. 249–253.