Alunos:
Cauan Nascimento Silva
Alexandre Minoru Zanoni Yassaka

Disciplina: Programação orientada a objetos

Professor: Carlos Verissimo

Centro Universitário Senac – SANTO AMARO

TEMA: Ferramenta de monitoramento de Maquinas e Equipamentos

1- Descrição do Domínio do Problema:

1.1 Descrição

Uma ferramenta que auxiliara o monitoramento de maquinas e equipamentos, armazenando informações e trabalhando com os dados de forma organizada e intuitiva, podendo acompanhar seu desempenho, estabelecendo uma comunicação entre o dispositivo e o operador.

2- Requisitos Funcionais e Não Funcionais:

2.1- Requisitos Funcionais

RF1 Registro de Dispositivos:

Permitir o cadastro de diferentes tipos de máquinas e equipamentos a serem monitorados.

RF2 Conexão e Comunicação:

Estabelecer conexão com os dispositivos para coletar dados.

Suportar diferentes protocolos de comunicação (por exemplo, TCP/IP, MQTT) para receber informações dos dispositivos.

RF3 Coleta de Dados:

Capturar dados operacionais das máquinas e equipamentos, como temperatura, pressão, velocidade, nível, etc.

Armazenar os dados coletados em um banco de dados.

RF4 Análise em Tempo Real:

Realizar análise em tempo real dos dados coletados para identificar anomalias ou condições fora do padrão.

Gerar alertas imediatos caso sejam detectadas situações críticas.

RF5 Visualização de Dados:

Fornecer uma interface gráfica para visualizar os dados coletados e o status das máquinas.

Apresentar gráficos, tabelas e indicadores de desempenho.

RF6 Histórico e Relatórios:

Manter um histórico dos dados coletados para permitir análises retrospectivas.

Gerar relatórios periódicos sobre o desempenho das máquinas e equipamentos.

RF7 Configuração de Alertas:

Permitir a configuração de parâmetros de alerta personalizados para cada dispositivo.

Notificar os usuários por meio de mensagens ou e-mails quando um alerta for acionado.

2.2- Requisitos Não Funcionais:

RNF1 Segurança:

Garantir a segurança dos dados coletados, transmitidos e armazenados, utilizando criptografia e medidas de proteção.

RNF2 Escalabilidade:

Lidar com um grande número de dispositivos e dados sem comprometer o desempenho.

RNF3 Disponibilidade:

Assegurar alta disponibilidade da ferramenta para que o monitoramento seja contínuo e confiável.

RNF4 Desempenho:

Responder às consultas e exibir dados de forma rápida, garantindo uma experiência fluida para os usuários.

RNF5 Usabilidade:

Oferecer uma interface intuitiva e de fácil uso para que os usuários possam configurar e interpretar os dados facilmente.

RNF6 Compatibilidade:

Ser compatível com diferentes dispositivos e sistemas operacionais, permitindo acesso a partir de diversas plataformas.

RNF7 Manutenção:

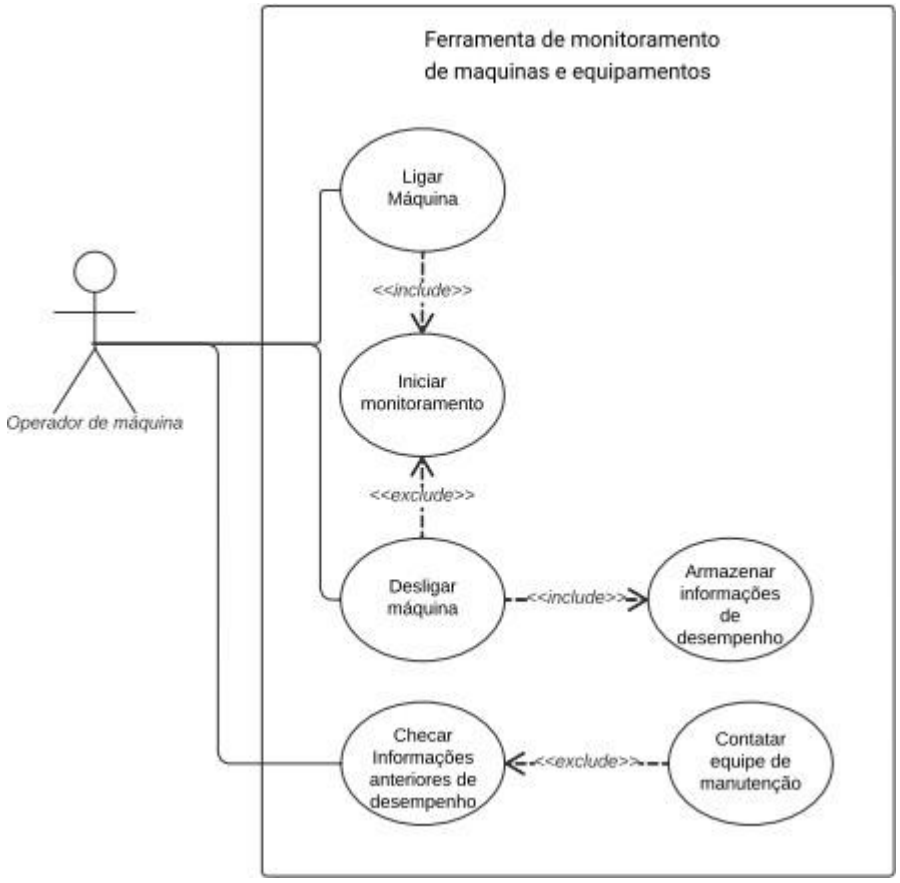
Facilitar a manutenção do sistema, permitindo atualizações e correções de forma eficiente.

RF8 Integração com Sistemas Externos:

Integrar-se a sistemas de gestão já existentes na organização para compartilhar informações relevantes.

3- Casos de Uso (UC)

3.1.1 - Diagrama



3.1.2 Detalhamento caso de uso 1 – Ferramenta de monitoramento de máquinas e equipamentos

Nome do caso de uso	1.1-Ligar Máquina
Atores	Operador de Máquina
Triggers	Necessidade de ligar a máquina
Pré-Requisito	Estar logado
Fluxo de eventos	O operador loga, procura pela máquina, clica sobre e ela e aperta o botão “ligar”

Nome do caso de uso	1.2-Iniciar Monitoramento
Atores	Operador de Máquina
Triggers	Ligar a máquina
Pré-Requisito	A máquina estar ligada
Fluxo de eventos	O operador liga a máquina e automaticamente o sistema começa a monitorar a máquina(temperatura, tempo de uso, vibração, óleo, espessura e ultrassom)

Nome do caso de uso	1.3-Desligar máquina
Atores	Operador de Máquina
Triggers	Problema encontrado no monitoramento/O operador desliga de forma manual
Pré-Requisito	A máquina estar ligada
Fluxo de eventos	O operador loga, procura pela máquina ligada, clica sobre e ela e aperta o botão “Desligar”/ Durante o monitoramento for encontrado alguma falha

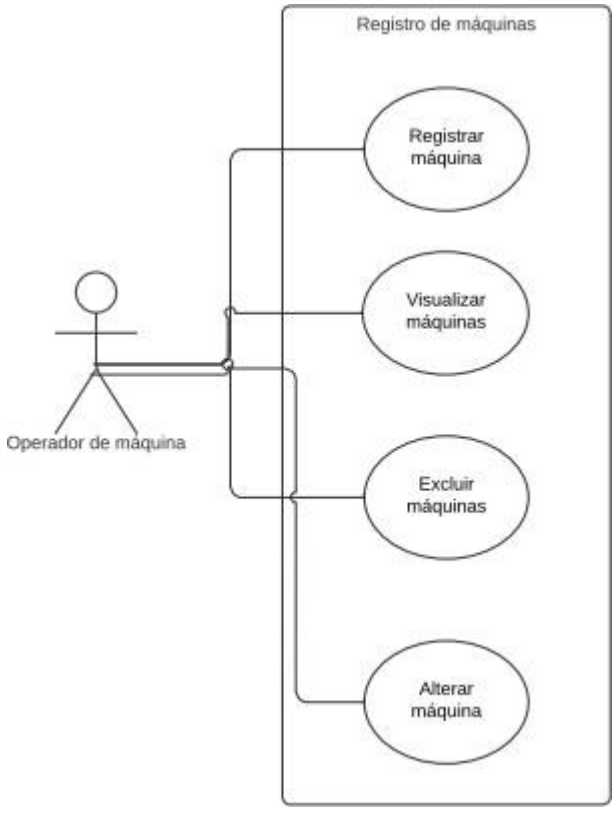
Nome do caso de uso	1.4- Armazenar informações de desempenho
Atores	Operador de Máquina
Triggers	Desligar máquina
Pré-Requisito	Ter feito o monitoramento durante o uso
Fluxo de eventos	Após o desligamento da máquina serão armazenados no banco de dados informações sobre o desempenho da máquina durante se uso(Temperatura média, tempo de funcionamento, vibração média, ciclo de vida e nível final do óleo)

Nome do caso de uso	1.5-Checar informações anteriores de desempenho
Atores	Operador de Máquina
Triggers	Necessidade de checar desempenho e status da máquina
Pré-Requisito	Estar logado
Fluxo de eventos	O operdador loga, clica sobre a máquina e clica no botão “detalhes”, então uma nova tela aparece com todas as informações sobre a máquina e seu último desempenho

Nome do caso de uso	1.6- Contatar equipe de manutenção
Atores	Operador de Máquina
Triggers	Ao analisar as informações na máquina, o operador encontra algum defeito
Pré-Requisito	Ser operador, estar logado
Fluxo de eventos	Após o checar as informações da máquina, o operador pode clicar no botão: “Contatar manutenção” então uma janela será aberta e ele poderá criar um chamado que será enviado a equipe de manutenção

3.2- Casos de Uso

3.2.1- Diagrama 2



3.2.2- Detalhamento caso de uso 2 – Registro de máquinas

Nome do caso de uso	2.1 – Registrar Máquina
Atores	Operador de Máquina

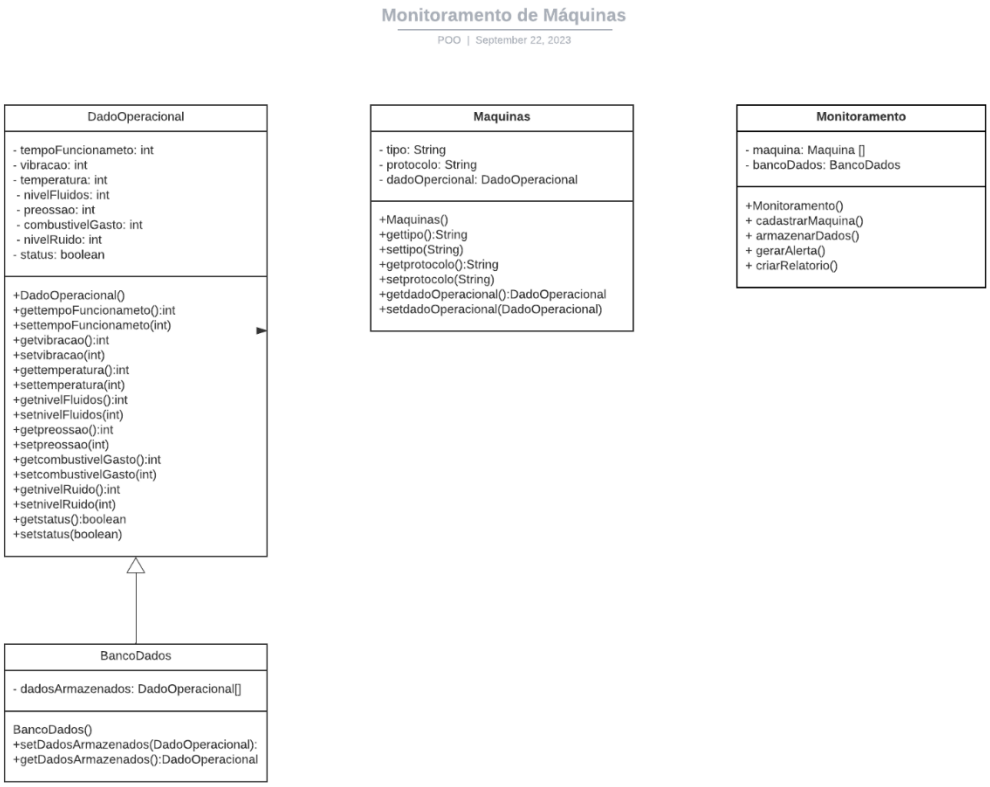
Triggers	Necessidade de registrar uma nova máquina
Pré-Requisito	Estar logado
Fluxo de eventos	Na lista de máquinas, o operador clicará no botão: “Nova máquina” onde será redirecionado para uma página que ele informará informações sobre ela e efetuará o registro(Marca, função, descrição, local, cor)

Nome do caso de uso	2.2 – Visualizar máquinas
Atores	Operador de Máquina
Triggers	Necessidade de visualizar as máquinas
Pré-Requisito	Estar logado
Fluxo de eventos	O operador entra no sistema e uma lista de máquinas aparecerá. Será filtrada por região da fábrica e o operador poderá clicar sobre a máquina para ver mais informações

Nome do caso de uso	2.3 – Excluir Máquina
Atores	Operador de Máquina
Triggers	Necessidade de excluir uma máquina
Pré-Requisito	Estar logado
Fluxo de eventos	O operador entra no sistema, clica sobre a máquina e clica no botão “Deletar”, informa sua senha para confirmar a ação

Nome do caso de uso	2.3 – Alterar Máquina
Atores	Operador de Máquina
Triggers	Necessidade de alterar alguma informação sobre a máquina
Pré-Requisito	Estar logado
Fluxo de eventos	O operador entra no sistema, clica sobre a máquina e clica no botão “Modificar”, uma tela para modificar as informações será apresentada e ao final o operador deverá apertar no botão “confirmar” e inserir sua senha para confirmar a ação.

4. Casos de uso



5. Encapsulamento e acoplamento do código

5.1 Classe Monitoramento:

Esta classe um alto nível de acoplamento, pois ele depende das classes Maquina e BancoDados
Encapsulando os atributos através dos modificadores de acessso private
Getters e setters para encapsulamento e manipulação dos atributos da classe.

```
package Monitoramentos;

import DadoOperacional.DadoOperacionais;

import Maquinas.Maquina;

import BancoDados.BancoDados;

//Esta classe um alto nível de acoplamento, pois ele depende das classes Maquina e
BancoDados
```

```
public class Monitoramento {

//Encapsulando os atributos através dos modificadores de acessso private

private Maquinas.Maquina[] maquinas;

private BancoDados bancoDados;

public Monitoramento (int CapacidadeBancoDados, int CapacidadeMaquina) {

maquinas = new Maquina[CapacidadeMaquina];

bancoDados = new BancoDados(CapacidadeBancoDados);

}

public void CadastrarMaquina(String tipo, String protocolo, DadoOperacionais[]
DadoOperacional) {

for(int i = 0; i < maquinas.length; i++) {

if(maquinas[i] == null) {

Maquina maquina = new Maquina();

maquina.setTipo(tipo);

maquina.setProtocolo(protocolo);

maquina.setDadoOperacional(DadoOperacional);

maquinas[i] = maquina;

break;

}

}

}

public void ArmazenarDados() {

for (int i = 0; i < maquinas.length;i++) {

Maquina maquina = maquinas[i];

if (maquina != null) {

maquina.getDadoOperacional();

bancoDados.setDadosArmazenados(null);

}

}

}
```

```
public void gerarAlerta() {

System.out.println("Alerta!!!");

}

public void CriarRelatorio() {

DadoOperacionais exibe = new DadoOperacionais();

System.out.println("Tempo de Funcionamento: " + exibe.getTempoFuncionamento());

System.out.println("\nVibração: " + exibe.getVibracao());

System.out.println("\nTemperatura: " + exibe.getTemperatura());

System.out.println("\nNivel de Fluídos: " + exibe.getNivelFluido());

System.out.println("\nPressão: " + exibe.getPressao());

System.out.println("\nCombustível Gasto: " + exibe.getCombustivelGasto());

System.out.println("\nNivel de Ruído: " + exibe.getNivelRuido());

System.out.println("\nVibração: " + exibe.getVibracao());

System.out.println("\nStatus: " + exibe.isStatus());

}

}
```

5.2 Classe Máquina :

Esta classe tem um médio nível de acoplamento, por ter uma dependência da classe DadoOperacionais.
Encapsulando os atributos através dos modificadores de acessso private.
Getters e setters para encapsulamento e manipulação dos atributos da classe.

```
package Maquinas;

//Esta classe tem um médio nível de acoplamento, por ter uma dependência da classe DadoOperacionais

public class Maquina {

//Encapsulando os atributos através dos modificadores de acessso private

private String tipo;

private String protocolo;

private DadoOperacional.DadoOperacionais [] DadoOperacional;

public Maquina() {
```

```
}

//Getters e setters para encapsulamento e manipulação dos atributos da classe.

public String getTipo() {

return tipo;

}

public void setTipo(String tipo) {

this.tipo = tipo;

}

public String getProtocolo() {

return protocolo;

}

public void setProtocolo(String protocolo) {

this.protocolo = protocolo;

}

public DadoOperacional.DadoOperacionais[] getDadoOperacional() {

return DadoOperacional;

}

public void setDadoOperacional(DadoOperacional.DadoOperacionais[] dadoOperacional)
{
```

5.3 Classe DadoOperacional:

Esta classe tem um baixo acoplamento, por ser baixa/ nula interdepêndecia de outras classes
Encapsulando os atributos através dos modificadores de acessso private
Getters e setters para encapsulamento e manipulação dos atributos da classe.

```java
package DadoOperacional;

//Esta classe tem um baixo acoplamento, por ser baixa/ nula interdepêndecia de outras classes

public class DadoOperacionais {

//Encapsulando os atributos através dos modificadores de acessso private

private int tempoFuncionamento;

private int vibracao;

private int temperatura;

private int nivelFluido;

private int pressao;

private int combustivelGasto;

private int nivelRuido;

private boolean status;

public DadoOperacionais() {

}

//Getters e setters para encapsulamento e manipulação dos atributos da classe.

public int getTempoFuncionamento() {

return tempoFuncionamento;

}

public void setTempoFuncionamento(int tempoFuncionamento) {

this.tempoFuncionamento = tempoFuncionamento;

}

public int getVibracao() {

return vibracao;

}

public void setVibracao(int vibracao) {

this.vibracao = vibracao;

}

public int getTemperatura() {

return temperatura;
```

```
}

public void setTemperatura(int temperatura) {

this.temperatura = temperatura;

}

public int getNivelFluido() {

return nivelFluido;

}

public void setNivelFluido(int nivelFluido) {

this.nivelFluido = nivelFluido;

}

public int getPressao() {

return pressao;

}

public void setPressao(int pressao) {

this.pressao = pressao;

}

public int getCombustivelGasto() {

return combustivelGasto;

}

public void setCombustivelGasto(int combustivelGasto) {

this.combustivelGasto = combustivelGasto;

}

public int getNivelRuido() {

return nivelRuido;

}

public void setNivelRuido(int nivelRuido) {

this.nivelRuido = nivelRuido;

}

public boolean isStatus() {

return status;
```

```
}

public void setStatus(boolean status) {

this.status = status;

}

}
```

5.4 Classe BancoDados:

Esta classe tem um médio nível de acoplamento, por ter uma dependência da classe DadoOperacionais
Encapsulando os atributos através dos modificadores de acessso private
Getters e setters para encapsulamento e manipulação dos atributos da classe.

```
package BancoDados;

import DadoOperacional.DadoOperacionais;

//Esta classe tem um médio nível de acoplamento, por ter uma dependência da classe DadoOperacionais

public class BancoDados {

//Encapsulando os atributos através dos modificadores de acessso private

private DadoOperacionais[] dadosArmazenados;

private int tamanho;

public BancoDados (int capacidade) {

dadosArmazenados = new DadoOperacionais[capacidade];

}

//Getters e setters para encapsulamento e manipulação dos atributos da classe.

public DadoOperacionais[] getDadosArmazenados() {

return dadosArmazenados;

}

public void setDadosArmazenados(DadoOperacionais dadoOperacional) {

if(tamanho < dadosArmazenados.length) {

dadosArmazenados[tamanho] = dadoOperacional;

tamanho++;

}else {

System.out.println("Banco de dados Cheio...");
```

```
}

}

}
```