# **Proposta Técnica: Plataforma de Manutenção Preditiva Industrial**

## **1. Justificativa do Problema e Descrição da Solução**

### **1.1. Problema**

* **Paradas inesperadas** em linhas de produção geram alto custo financeiro e atrasos operacionais.
* A falta de monitoramento contínuo impede a **identificação precoce** de falhas nos equipamentos.

### **1.2. Solução Proposta**

Desenvolver uma **plataforma inteligente** que:

1. **Monitore** em tempo real parâmetros críticos de equipamentos (vibração, temperatura, umidade).
2. **Detecte anomalias** operacionais via análise de dados e Machine Learning.
3. **Preveja falhas** antes que ocorram, reduzindo custos com manutenção corretiva.
4. **Gere alertas e recomendações** preventivas automatizadas (dashboards, relatórios).

## **2. Tecnologias Utilizadas**

|  |  |  |
| --- | --- | --- |
| **Camada** | **Ferramenta / Serviço** | **Justificativa** |
| Coleta de Dados | Python + ESP32 (MQTT) ou Simulador em Python | Leve, flexível e fácil de simular dados de sensores |
| Ingestão/API | FastAPI (Python) | Simples, performático e bem integrado com ML |
| Armazenamento | PostgreSQL (AWS RDS ou local) | Open source, escalável e compatível com análises |
| Processamento / ML | Scikit-learn / TensorFlow | Ampla comunidade e suporta modelos de classificação e detecção de anomalias |
| Agendamento | AWS Lambda + EventBridge ou Cron local | Execução periódica de pipelines |
| Front‑end / Dashboard | React.js + Dash (Plotly) / Grafana | Integração rápida e visualizações interativas |
| Infraestrutura | AWS EC2 (API + Front‑end), RDS, S3 (logs) | Escalável, pay-per-use e familiar para a equipe |
| Diagramação | diagrams.net | Grátis e colaborativo para criar a arquitetura |

## **3. Arquitetura e Pipeline de Dados**

[Sensores ESP32] --MQTT--> [API Coleta (FastAPI)] --> [PostgreSQL RDS]

|

Agendamento (Lambda/EventBridge)

v

[API Análise (FastAPI + ML)]

|

Atualiza resultados no RDS

|

------------------------------------------------------

| |

[Dashboard React/Dash] [Relatórios Automáticos]

1. **Coleta**: dispositivos ESP32 publicam via MQTT para a API de Coleta.
2. **Armazenamento**: dados brutos guardados no PostgreSQL.
3. **Processamento**: rotina agendada dispara API de Análise, executando modelos de ML.
4. **Visualização**: front‑end consulta RDS e apresenta painel interativo.
5. **Alertas**: relatórios gerados via script Python + envio por e‑mail ou webhook.

## **4. Estratégia de Coleta de Dados**

|  |  |  |
| --- | --- | --- |
| **Modo** | **Descrição** | **Passos de Implementação** |
| Simulado | Gerar dados de vibração, temperatura e umidade em Python. | 1. Criar script simulator.py com valores randômicos e padrões de falha. |
|  |  | 2. Publicar via HTTP/MQTT na API de Coleta a cada segundo. |
| Sensores Reais | Usar ESP32 com sensores DHT22 (umidade/temperatura) e acelerômetro. | 1. Configurar firmware em MicroPython. |
|  |  | 2. Enviar via MQTT para o broker na nuvem. |

**Observação**: Na fase atual, começar pelo modo simulado para validar pipeline.

## **5. Plano Inicial de Desenvolvimento e Papéis**

|  |  |
| --- | --- |
| **Responsável** | **Atividades Principais** |
| Líder de Projeto | Coordenação geral, revisão de entregáveis, integração com tutores. |
| Engenheiro de Dados | Desenvolver API Coleta, simulação de dados e schema do banco. |
| Cientista de Dados | Construir e validar modelos de detecção de anomalias (Scikit-learn). |
| Backend Engineer | Implementar API Análise, agendamento e integração com ML. |
| Front‑end Developer | Criar dashboard em React/Dash e configurar visualizações. |
| DevOps / Infra | Provisionar AWS (EC2, RDS, Lambda) e configurar CI/CD básico. |

**Cronograma (2 semanas)**

1. **Semana 1**: Setup do repositório e simulação de dados + API Coleta.
2. **Semana 2**: Pipeline de análise + dashboard protótipo + documentação.

## **6. Estrutura do Repositório (README)**

repo-name/

├── README.md # Este arquivo

├── simulator/ # Scripts de simulação de dados

│ └── simulator.py

├── api-coleta/ # FastAPI de ingestão

│ └── main.py

├── api-analise/ # FastAPI de processamento e ML

│ └── main.py

├── dashboard/ # Front‑end (React/Dash)

├── infra/ # Scripts Terraform / CloudFormation (opcional)

└── docs/

└── arquitetura.drawio

**Próximos Passos**:

* Validar fluxo simulado e pipeline básico.
* Configurar AWS e testes de carga mínima.
* Refinar modelo de ML com dados históricos fictícios.