Ives de Oliveira Furtado

# Desenvolvimento de uma Plataforma de Monitoramento e Controle de Processos Industriais baseada em Tecnologias Web

Viçosa, MG

2023

Ives de Oliveira Furtado

# Desenvolvimento de uma Plataforma de Monitoramento e Controle de Processos Industriais baseada em Tecnologias Web

Monografia apresentada ao Departamento de Engenharia Elétrica do Centro de Ciências Exatas e Tecnológicas da Universidade Federal de Viçosa, para a obtenção dos créditos da disciplina ELT 402 – Projeto de Engenharia II – e cumprimento do requisito parcial para obtenção do grau de Bacharel em Engenharia Elétrica.

Orientador Prof. Dr. André Gomes Torres

Viçosa, MG

2023

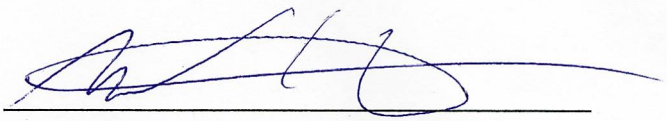
Ives de Oliveira Furtado

# Desenvolvimento de uma Plataforma de Monitoramento e Controle de Processos Industriais baseada em Tecnologias Web

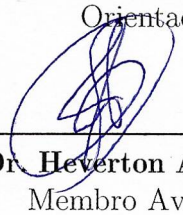
Monografia apresentada ao Departamento de Engenharia Elétrica do Centro de Ciências Exatas e Tecnológicas da Universidade Federal de Viçosa, para a obtenção dos créditos da disciplina ELT 402 – Projeto de Engenharia II – e cumprimento do requisito parcial para obtenção do grau de Bacharel em Engenharia Elétrica.

Trabalho aprovado em 12 de dezembro de 2023.

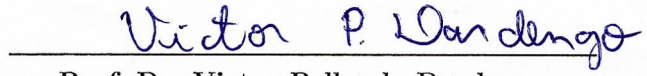
COMISSÃO EXAMINADORA



**Prof. Dr. André Gomes Torres**
Orientador



**Prof. Dr. Heverton Augusto Pereira**
Membro Avaliador



**Prof. Dr. Victor Pellanda Dardengo**
Membro Avaliador

Viçosa, MG
2023

# Agradecimentos

Todas as linhas desse projeto foram escritas com pedaços de cada um que passou pelo meu caminho. Sem o conhecimento e a experiência de vocês, nada disso seria possível.

Nos momentos de falta de clareza, agradeço a Deus por me iluminar e demonstrar o caminho a seguir. Sem a fé dificilmente teria chegado até aqui.

Agradeço à minha família, que sempre me apoiou e incentivou a seguir em frente. Me proporcionou uma base sólida e ensinou a ter perseverança.

Não poderia deixar de agradecer aos meus amigos de curso e aos que dividiram a casa comigo durante esses anos, me apoiando e incentivando a seguir em frente. Sem eles, a caminhada teria sido muito mais difícil.

Agradeço a todos os professores que tive a oportunidade de conhecer e que contribuíram para a minha formação acadêmica e pessoal. Em especial, agradeço ao Professor Dr. André Gomes Torres, que me orientou durante a realização deste trabalho.

Por fim, agradeço a todos que, de alguma forma, contribuíram para a minha formação.

*'Se eu vi mais longe foi por estar sobre ombros de gigantes." (Sir Isaac Newton, 1675)*

# Resumo

A supervisão eficiente de plantas industriais é essencial para a operação segura e otimizada desses complexos sistemas. A implementação de sistemas supervisórios permite aos operadores ter acesso imediato às condições reais da planta, facilitando a realização de ajustes e tomadas de decisão informadas. Este trabalho se concentra em desenvolver uma arquitetura capaz de coletar, armazenar e exibir dados de processos de uma planta industrial em pequena escala, além de habilitar a execução de operações de controle. Ao possibilitar a intervenção precisa e fundamentada na planta industrial, a arquitetura proposta tem o potencial de melhorar significativamente a operacionalidade e eficiência do sistema. A escolha do NodeJs como plataforma do projeto é estratégica, aproveitando-se da capacidade e versatilidade do interpretador JavaScript para o desenvolvimento do sistema supervisório. Paralelamente, o banco de dados não relacional MongoDB foi selecionado para gerenciar as informações devido à sua flexibilidade e escalabilidade, o que é essencial em ambientes que geram grandes volumes de dados, como é o caso do armazenamento de dados de diversos sensores. A arquitetura da planta didática da Smar (utilizada no trabalho) adota o protocolo de comunicação *Open Platform Communication*. Este padrão é amplamente reconhecido na indústria pelo fornecimento de um meio confiável de troca de dados entre múltiplos dispositivos e sistemas. A utilização dessa arquitetura de comunicação permite uma interação fluida e segura com os componentes da planta didática, garantindo que as operações sejam executadas de modo coeso e eficiente. Assim, essa combinação de ferramentas e protocolos constitui a espinha dorsal do sistema supervisório projetado.

**Palavras-chaves**: Sistemas Supervisórios; Internet das Coisas; Automação e Controle de Processos; Planta Industrial.

# Abstract

Effective supervision of industrial plants is essential for the safe and optimized operation of these complex systems. The implementation of supervisory systems allows operators to have immediate access to the actual conditions of the plant, facilitating the execution of adjustments and informed decision-making. This work focuses on developing an architecture capable of collecting, storing, and displaying process data from a small-scale industrial plant, as well as enabling the execution of control operations. By enabling precise and well-founded intervention in the industrial plant, the proposed architecture has the potential to significantly improve the operability and efficiency of the system. The choice of NodeJs as the project platform is strategic, taking advantage of the capacity and versatility of the JavaScript interpreter for the development of the supervisory system. Concurrently, the non-relational database MongoDB was selected to manage the information due to its flexibility and scalability, which is essential in environments that generate large volumes of data, as is the case with storing data from various sensors. The educational plant architecture of Smar (used in this work) adopts the Open Platform Communication protocol. This standard is widely acknowledged in the industry for providing a reliable means of exchanging data between multiple devices and systems. The use of this communication architecture allows for fluid and secure interaction with the components of the educational plant.

**Keywords**: Supervisory Systems; Internet of Things; Automation and Process Control; Industrial Plant.

# Lista de figuras

# Lista de abreviaturas e siglas

| | |
|---|---|
| FF | Foundation Fieldbus |
| API | Application Programming Interface |
| REST | Representational State Transfer |
| ORM | Object-Relational Mapping |
| LEPP | Laboratório de Pesquisa de Processos Industriais e Padrões de Movimentos e Acústica |
| DEL | Departamento de Engenharia Elétrica |
| SCADA | Supervisory Control and Data Acquisition |

# Sumário

# 1 Introdução

Com o avanço das tecnologias de informação e comunicação, a quantidade de dados gerados e armazenados tem crescido exponencialmente. A análise desses dados tem se tornado cada vez mais importante para a tomada de decisões em diversas áreas, como por exemplo, na área de saúde, na área financeira, na área de segurança, entre outras (SAGIROGLU; SINANC, 2013).

O desenvolvimento de sistemas supervisórios conectados à internet tem facilitado a coleta e o armazenamento de dados no meio industrial, permitindo que esses dados sejam utilizados para a tomada de decisões informada.

Além dos avanços técnicos específicos ao sistema supervisório, o setor industrial como um todo tem passado por mudanças significativas, impulsionadas por melhorias nos dispositivos de conexão e pela ampliação tanto da velocidade como da disponibilidade da internet.

A crescente capacidade de processamento dos sistemas computacionais modernos desempenha um papel fundamental nesse contexto. Esses avanços têm permitido que a indústria eleve seu nível de tecnologia e produtividade, acompanhando um movimento mais amplo que vem sendo referenciado por especialistas como a Quarta Revolução Industrial, um termo cunhado por Klaus Schwab e detalhado em seu influente trabalho publicado em 2016 (SCHWAB, 2016).

## 1.1 Objetivo

Descrito o contexto que envolve o trabalho, o objetivo do mesmo é desenvolver uma aplicação que permita supervisorar a planta didática PD3 e que disponibilize esses dados por meio de uma API REST e WebSocket.

Vale notar que o trabalho não entrará em detalhes sobre como é implementado as configurações do servidor OPC e os softwares da SMAR, se dedicando a apresentar a aplicação desenvolvida e como ela se comunica com a planta didática.

Tem-se como objetivo também que o trabalho seja utilizado como uma base para futuros projetos que envolvam a planta didática PD3. Fornecendo uma estrutura para a aquisição dos dados da planta e para a visualização desses dados em tempo real.

# 2 Referencial Teórico

A seguir são apresentados os conceitos teóricos que fundamentam o trabalho, sendo eles: sistemas supervisórios, protocolo Foundation Fieldbus, planta didática SMAR, dispositivos de atuação e medição, OPC-DA, NodeJs, API REST, MongoDB e WebSocket.

É necessário apresentar esses conceitos para que o leitor possa compreender os elementos que estão envolvidos no desenvolvimento do sistema.

## 2.1 Sistemas Supervisórios

Um sistema supervisório engloba um conjunto de ferramentas que permitem a supervisão e o controle de processos industriais. E seu propósito é permitir que o operador tenha uma visão geral do processo, de forma a facilitar a tomada de decisões. Um nome comum para esse tipo de sistema é SCADA (*Supervisory Control and Data Acquisition*), que em português significa Sistema de Supervisão e Aquisição de Dados.

O SCADA deve permitir que o usuário tenha acesso aos dados do processo em tempo real, permitindo ao operador tomar decisões rápidas e eficientes (ACKERMAN, 1992). Na Figura 1 é demonstrado um exemplo de sistema supervisório.

Figura 1 – Operador utilizando um sistema supervisório.

Fonte: (ALTUS, 2017).

## 2.2 Planta Didática SMAR PD3

A SMAR é uma das maiores fabricantes de sistemas de automação industrial do Brasil, tendo sido fundada em 1974. A empresa multinacional oferece produtos que envolvem CLPs, dispositivos de medição e atuação, software de supervisão e controle, plantas didáticas, entre outros (SMAR, 2023).

A planta didática SMAR - PD3 é um sistema de controle de processos industriais que permite a simulação de diversos processos industriais, como por exemplo, o controle de nível, o controle de temperatura e o controle de vazão. Fornecendo ao usuário a capacidade de operar diversas malhas de controle e de analisar o comportamento do processo realizado em tempo real.

A planta utiliza o protocolo Foundation Fieldbus para a comunicação entre os dispositivos de medição e atuação e o CLP. Do CLP tem-se uma conexão LAN que permite a comunicação com um computador, onde é executado o software proprietário da SMAR que gerencia os dados obtidos da planta e disponibiliza, por meio de um servidor OPC, os dados para a rede local. Na Figura 2 é possível visualizar a planta didática SMAR - PD3.

Figura 2 – Planta Didática SMAR - PD3.

### 2.2.1 Protocolo Foundation Fieldbus

O protocolo Foundation Fieldbus é um protocolo de comunicação digital utilizado em sistemas de automação industrial. Ele foi desenvolvido com o objetivo de substituir os protocolos analógicos 4-20 mA e HART.

O protocolo permite a comunicação entre os dispositivos de medição e atuação e o CLP, utilizando um par trançado de fios, que pode ser compartilhado entre vários dispositivos, reduzindo assim o custo de instalação e padronizando o meio de comunicação (THOMESSE, 2005).

### 2.2.2 Fieldbus Universal Bridge - DFI 302

O DFI 302 é um dispositivo que trabalha como um controlador e o sistema host de uma rede FF, sendo bastante flexível e modular. É um elemento principal na arquitetura

da Planta Didática PD3, combinando as funções de comunicação com acesso às saídas e entradas do processo.

E também integrado ao DFI 302, está o módulo de comunicação FF, que é responsável por realizar a comunicação com os dispositivos de campo. Na Figura 3 é possível visualizar o DFI 302.

Figura 3 – Fieldbus Universal Bridge - DFI 302.

Fonte: (ZEILMANN, 2002).

### 2.2.3 Dispositivos de Atuação e Medição

Os dispositivos de atuação e medição da Planta Didática PD3 utilizados no trabalho são:

- TT302 – Transmissor de Temperatura

Realiza a medição da temperatura no tanque e pode ser equipado com termoresistências ou termopares.

Figura 4 – Transmissor de Temperatura TT302.


Fonte: (SAMPAIO, 2013).

- LD302 – Transmissor de Vazão e Pressão Fieldbus

Realiza a medição de pressão em suas formas diferencial, absoluta e manométrica, além de medir níveis e fluxos.

Figura 5 – Transmissor de Vazão e Pressão Fieldbus LD302.

Fonte: (SAMPAIO, 2013).

- FY302 – Posicionador de Válvula Fieldbus

Atua como um intermediário entre a rede Fieldbus e as válvulas pneumáticas. O seu método de medição baseia-se no uso de um sensor de efeito Hall para determinar o grau de abertura da válvula e utiliza um diferencial de pressão para ajustá-la conforme a abertura especificada (RIBEIRO et al., 2012).

Figura 6 – Posicionador de Válvula Fieldbus FY302.

Fonte: (SAMPAIO, 2013).

### 2.2.4 OPC

O OPC (*Open Platform Communications*) é um padrão de comunicação industrial que permite a comunicação entre sistemas de diferentes fabricantes. A sua padronização é definida pela *OPC Foundation*, que é uma organização sem fins lucrativos que tem como objetivo desenvolver padrões de comunicação para sistemas de automação industrial (FONSECA, 2002).

A especificação de acesso a dados (*Data Access*) é um padrão de comunicação que permite a troca de dados entre um servidor OPC e um cliente OPC. Pode-se utilizar o OPC-DA para acessar dados de um CLP, de um sistema supervisório ou de qualquer outro dispositivo que implemente o padrão OPC-DA. Esse sistema de comunicação está presente nos softwares disponibilizados pela Smar para a Planta Didática PD3, permitindo que os dados da planta sejam acessados por um cliente configurado com os dados de autenticação.

#### 2.2.4.1 DCOM

O DCOM (*Distributed Component Object Model*) é um protocolo de comunicação desenvolvido pela Microsoft que permite a comunicação entre processos em computadores diferentes (MICROSOFT, 2014). Ele é utilizado pelo software da SMAR para realizar a comunicação entre um cliente e o servidor OPC. Assim permitindo que um usuário pela rede local tenha acesso aos dados da planta.

A porta padrão de comunicação utilizada pelo DCOM é a porta 135, e para que a comunicação seja realizada é necessário que essa porta esteja aberta no computador que está executando o servidor OPC e também na rede local (roteador). A configuração do DCOM é realizada por meio do *Component Services*, que é um programa que permite a configuração de componentes do Windows. O próprio Windows, após ser configurado, redireciona as requisições que chegam na porta 135 para a porta que foi configurada para o servidor OPC. Geralmente as portas reservadas para o servidor OPC variam entre 1000 e 1100.

## 2.3 NodeJs

O NodeJs é uma plataforma de desenvolvimento de aplicações web, que utiliza a linguagem de programação JavaScript. Ele foi desenvolvido por Ryan Dahl em 2009, e desde então tem ganhado popularidade, principalmente por ser uma plataforma de desenvolvimento de aplicações web que utiliza a linguagem JavaScript, que é uma linguagem de programação bastante popular e que possui uma grande comunidade de desenvolvedores.

No presente trabalho o NodeJs foi utilizado para desenvolver a aplicação que realiza a aquisição dos dados da planta didática PD3 e os disponibiliza por meio de uma API REST e WebSocket. O desenvolvimento do servidor foi realizado utilizando o *framework* Express, que é um para desenvolvimento de aplicações web que utiliza o NodeJs (EXPRESS, 2010). E também foi utilizado o ORM Mongoose, que realiza a modelagem de dados e permite a conexão com o banco de dados MongoDB (MONGOOSE, 2013).

Para desenvolver a parte de interação da aplicação, tem-se a utilização do framework React, que é um framework para desenvolvimento de interfaces de usuário que utiliza a linguagem JavaScript (REACT, 2013). Além da utilização de WebSocket para realizar a comunicação em tempo real entre o servidor Express e o cliente React.

### 2.3.1 API REST

A API REST (*Representational State Transfer*) é um estilo de arquitetura de software que define um conjunto de restrições para o desenvolvimento de aplicações web.

Essas restrições são definidas por Roy Fielding em sua tese de doutorado (FIELDING, 2000).

A definição de restrições busca padronizar o desenvolvimento de sistemas, permitindo então definir as interações entre eles, buscando assim facilitar o desenvolvimento e também a comunicação. Algumas restrições definidas por Fielding são: interface de comunicação uniforme, arquitetura interna do servidor independente do cliente, sistema de cache de recursos, armazenamento de estado somente no cliente e código sob demanda (a capacidade do servidor enviar um código que possa ser executado no cliente) (TECHTARGET, 2020).

Ela é bastante utilizada para o desenvolvimento de aplicações web, principalmente por ser uma arquitetura bastante simples e que permite a comunicação entre sistemas desenvolvidos em diferentes linguagens de programação.

### 2.3.2 MongoDB

O MongoDB é um banco de dados orientado a documentos, que utiliza o formato JSON para armazenar os dados (ROUTER, 2021). Ele é um banco de dados não relacional. O MongoDB foi utilizado para armazenar os dados obtidos da planta didática PD3 e também os dados dos usuários do sistema.

### 2.3.3 WebSocket

O WebSocket é uma tecnologia que permite a comunicação bidirecional entre um cliente e um servidor por meio de mensagens e eventos, tendo seu uso cada vez mais crescente devido a necessidade de comunicação em tempo real entre dispositivos (e sistemas) com acesso à internet (DUBEY, 2023).

Na próxima seção serão demonstrados os tipos de eventos do sistema.

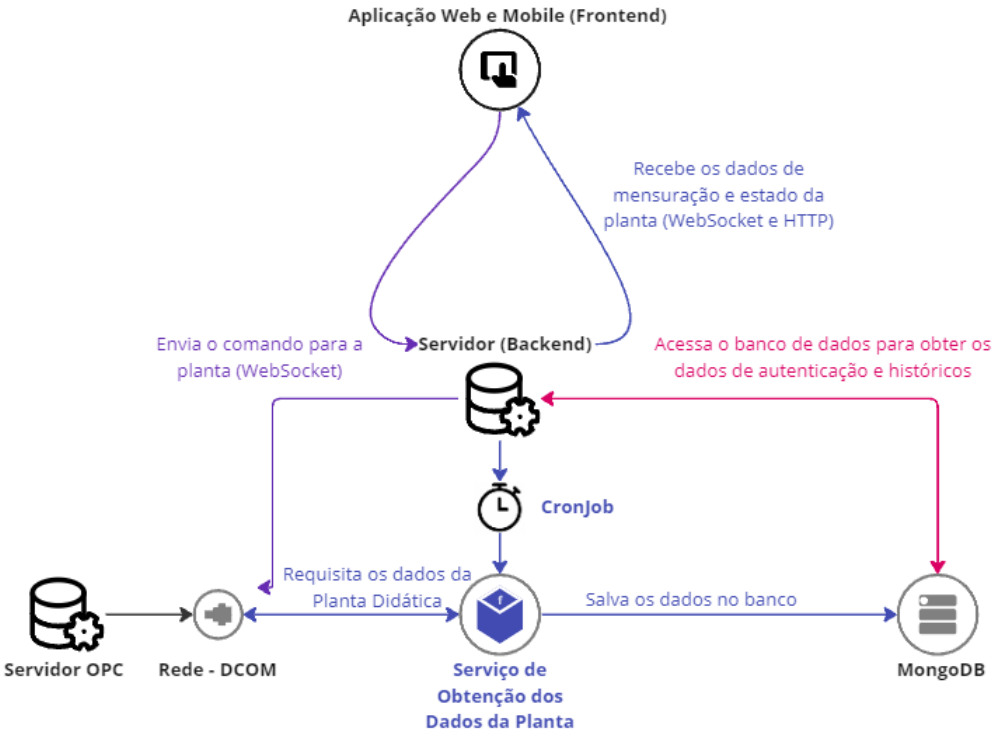
# 3 Materiais e Métodos

O desenvolvimento desse trabalho foi realizado no Laboratório de Pesquisa de Processos Industriais e Padrões de Movimentos e Acústica (LEPP) do Departamento de Engenheria Elétrica da Universidade Federal de Viçosa.

Nesse trabalho utiliza-se primariamente o Javascript com sua extensão Typescript para desenvolver todo o sistema. O Typescript é um superconjunto de Javascript que adiciona tipagem estática e alguns recursos de orientação a objetos e é compilado para Javascript.

A Figura 7 apresenta a arquitetura do sistema.

Figura 7 – Diagrama de arquitetura do sistema.



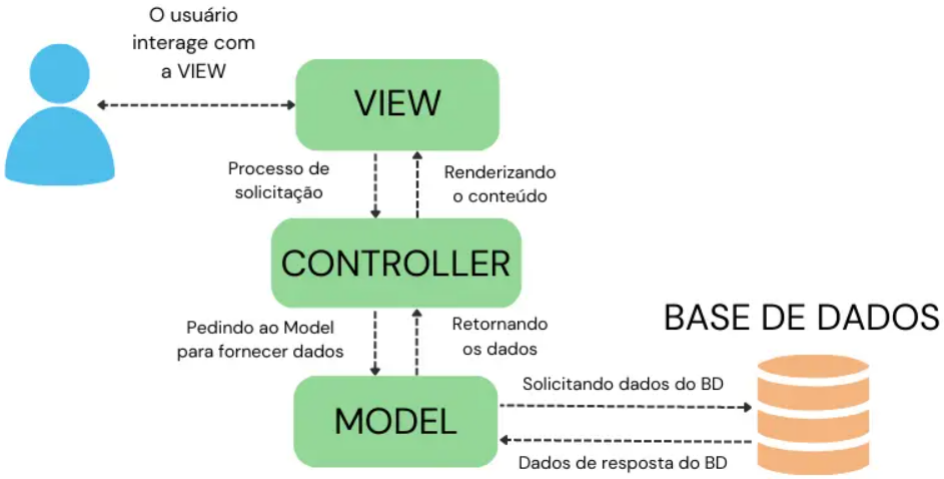

A divisão em subtópicos auxiliará o leitor a compreender melhor o trabalho. A seguir, tem-se a descrição de cada um dos componentes do sistema.

## 3.1 Servidor Backend

Para o desenvolvimento do servidor backend foi utilizado o framework ExpressJs com a arquitetura MVC (Model-View-Controller) e o banco de dados MongoDB. A

escolha do ExpressJs se deu pela sua simplicidade e facilidade de uso, além de ser um dos frameworks mais utilizados para o desenvolvimento de aplicações web em NodeJs. Na Figura 8 é demonstrada a arquitetura MVC.

Figura 8 – Arquitetura MVC.




Fonte: (USANDOPY, 2023).

Além disso, tem-se a vantagem de utilizar o mesmo linguagem tanto no frontend quanto no backend, o que facilita a comunicação entre os dois. Principalmente com a capacidade de compartilhar-se os contratos de tipagens do Typescript por meio de uma biblioteca.

Pode-se dividir o backend em três partes: o servidor HTTP, o banco de dados e o servidor WebSocket.

### 3.1.1 Servidor HTTP

O servidor HTTP é responsável por receber as requisições HTTP do frontend e encaminhá-las para o controlador responsável.

As principais partes do servidor HTTP são:

- **Rotas**: As rotas são responsáveis por receber as requisições HTTP e encaminhá-las para o controlador responsável.
- **Controladores**: Os controladores são responsáveis por receber as requisições HTTP, processá-las e retornar uma resposta.
- **Middlewares**: Os middlewares são funções que são executadas antes de uma rota ser executada.

- **Serviços**: Os serviços são funções que são executadas dentro dos controladores e são responsáveis por realizar operações customizadas e genéricas.
- **Modelos**: Os modelos são responsáveis por definir a estrutura dos dados que serão armazenados no banco de dados.
- **Repositórios**: Os repositórios são responsáveis por realizar as operações de CRUD (*Create, Read, Update, Delete*) no banco de dados.

### 3.1.2 Servidor WebSocket

O servidor WebSocket é responsável por controlar toda a comunicação em tempo real entre o frontend e o backend. Ele é responsável por receber as requisições WebSocket e retornar as respostas.

Pode-se separar a funcionalidade do servidor WebSocket entre emitir e ouvir eventos.

A aplicação tem três eventos principais: o evento de conexão, o evento de mensuração e o evento de operação.

- **Evento de conexão**: O evento de conexão é recebido pelo servidor quando um usuário se conecta.
- **Evento de mensuração**: O evento de mensuração é disparado pelo servidor quando o serviço de obtenção de dados da planta é executado com sucesso.
- **Evento de operação**: O evento de operação é disparado pelo cliente para o servidor quando o usuário realiza uma operação.

O servidor WebSocket é conectado ao servidor HTTP e compartilha o mesmo contexto de execução. Isso é possível pois foi utilizado a biblioteca Socket.IO que é compatível com o Express (SOCKET.IO, 2014).

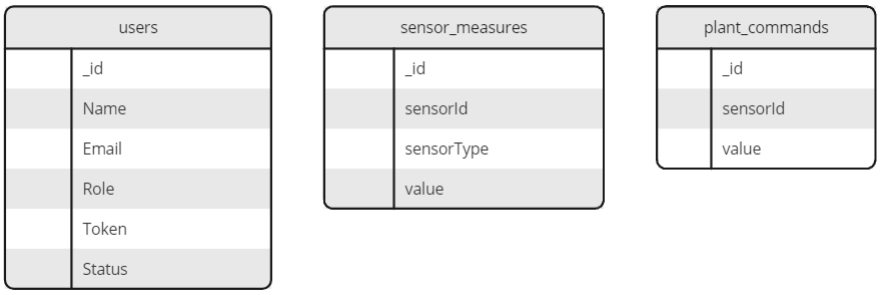
### 3.1.3 Banco de dados

O banco de dados é responsável por armazenar os dados dos usuários, as mensurações realizadas pelos dispositivos da planta e as operações realizadas pelos usuários.

A escolha do MongoDB se deu pela sua simplicidade e facilidade de uso, além de ser um dos bancos de dados mais utilizados para o desenvolvimento de aplicações web em NodeJs. Sua estrutura de dados é baseada em documentos, o que facilita a manipulação.

Na Figura 9 é demonstrado os modelos do banco de dados.

Figura 9 – Modelos no Banco.




A escolha dessa estrutura de dados se deu pela compatibilidade com os dados recebidos da planta. Foi possível armazenar os dados recebidos da planta sem a necessidade de realizar conversões. Facilitando a obtenção dos dados e a manipulação dos mesmos.

Abaixo é demonstrada a definição das interfaces dos modelos do banco de dados.

Algoritmo 3.1 – Definição dos ids dos sensores.

```
export type SensorIds =
  | "left_temp"
  | "right_temp"
  | "left_flow"
  | "right_flow"
  | "left_flow_controller"
  | "right_flow_controller"
  | "temperature_controller"
  | "motor_1"
  | "motor_2"
  | "level_transmitter";
```

Algoritmo 3.2 – Definição dos tipos de mensuração.

```
export type SensorMeasureTypes =
  | "temperature"
  | "flow"
  | "flow_controller"
  | "temperature_controller"
  | "motor"
  | "level";
```

Algoritmo 3.3 – Definição dos tipos e status de usuários.

```
export type UserRoles = "admin" | "manager" | "technician";

export type UserStatus = "active" | "inactive" | "banned";
```

Algoritmo 3.4 – Definição das interfaces dos modelos.

```
export interface SensorMeasure {
  sensorId: SensorIds;
  sensorType: SensorMeasureTypes;
  value: number | boolean | string;
}

export interface PlantCommand {
  sensorId: SensorIds;
  value: number | boolean | string;
}

export interface UserData {
  email: string;
  name: string;
  role: UserRoles;
  status: UserStatus;
  token: string;
}
```

## 3.2 Cliente Frontend

Para o desenvolvimento do cliente frontend foi utilizado o framework ReactJS e o Material-UI para a criação da interface gráfica. A escolha do ReactJS se deu pela sua simplicidade e facilidade de uso, além de ser o framework mais utilizado para o desenvolvimento de aplicações web em Javascript.

A estrutura do frontend é separada em três partes: os componentes, os contextos e a interface gráfica.

### 3.2.1 Componentes

Os componentes são responsáveis por controlar a lógica da aplicação. Eles são responsáveis por receber os eventos do usuário e realizar as operações necessárias.

Os componentes são divididos em dois tipos: os componentes de página e os componentes de interface.

- **Componentes de página**: Os componentes de página são responsáveis pela exibição dos dados da aplicação em uma página.
- **Componentes de interface**: Os componentes de interface são responsáveis por controlar a lógica da aplicação em pequenos contextos.

Para a aplicação foi desenvolvido um componente que cria um dashboard completo e altamente customizável. Esse componente é responsável por exibir todos os gráficos históricos, atuais e de controle da planta. Abaixo é exibido um exemplo de exibição e configuração de um item do dashboard.

Algoritmo 3.5 – Exemplo de código para exibição de um item do dashboard.

```
import ReactApexcharts from "@/components/client/react-apexcharts";
import { useMemo } from "react";
import { DashboardItem } from "../../types";
import mergeDeep from "@/utils/merge-deep";

const DashboardItemChart: React.FC<DashboardItem<"chart">> = (props) => {
  const chartData = useMemo(() => {
    return mergeDeep(props.data, {
      options: {
        chart: { foreColor: "rgba(0, 0, 0, 0.87)" },
        tooltip: { theme: "light" },
      },
    });
  }, [props.data]);

  return <ReactApexcharts {...chartData} width={"100%"} />;
};

export default DashboardItemChart;
```

Algoritmo 3.6 – Exemplo de código para configuração de um item do dashboard.

```
const sensorsDataLineChart: DashboardItem<"chart"> = {
  title: "Historico dos Sensores de Fluxo",
  type: "chart",
  data: {
    type: "line",
    height: 200,
    options: {
      xaxis: {
        categories: labels,
        labels: {
          show: false,
        },
      },
      yaxis: {
        max: maxValue[0] * 1.5,
      },
      chart: {
        animations: {
          enabled: true,
          easing: "linear",
          dynamicAnimation: {
            speed: 1000,
          },
        },
      },
      },
    },
    series: [
      {
        name: "Sensor de Fluxo (Esquerdo)",
        data: leftSensorsValues,
      },
      {
        name: "Sensor de Fluxo (Direito)",
        data: rightSensorsValues,
      },
    ],
  },
};
```

Nos algoritmo acima é possível notar que o componente é altamente customizável. O Algoritmo 3.5 é responsável por receber a configuração definida pelo Algoritmo 3.6 e exibir o gráfico. A função de *chartData* é onde o componente realiza a customização do gráfico.

No Algoritmo 3.6 é exibido o objeto de configuração, onde é possível definir o título do gráfico (*title*), o tipo do gráfico (*type*), os dados do gráfico (*data*) e as séries do gráfico (*series*).

### 3.2.2 Contextos

Os contextos são responsáveis por controlar o estado da aplicação. Eles são responsáveis por armazenar os dados que serão compartilhados entre os componentes.

Na aplicação o dashboard é um contexto que armazena os dados dos sensores e dos controladores da planta. Além disso, ele armazena os dados das operações realizadas pelo usuário.

Para configurar quais itens serão exibidos, o programador pode alterar a ordem de criação dos itens na aplicação.

Algoritmo 3.7 – Código de configuração de quais itens serão exibidos.

```
const dashboard = useMemo(() => {
  return createDashboard({
    items: [
      "left_temperature_gauge",
      "right_temperature_gauge",
      "left_flow_controller_gauge",
      "right_flow_controller_gauge",
      "level_transmitter_line_chart",
      "flow_controller_line_chart",
      "flow_sensor_line_chart",
      "temperature_sensor_line_chart",
      "plant_commands_history_logger",
    ],
    data: {
      sensorsData: {
        left_temp: leftTempSensorData.current,
        right_temp: rightTempSensorData.current,
        left_flow: leftFlowSensorData.current,
        right_flow: rightFlowSensorData.current,
        left_flow_controller: leftFlowControllerData.current,
        right_flow_controller: rightFlowControllerData.current,
        level_transmitter: levelTransmitterData.current,
      },
      plantCommandsData: plantCommandsData.current,
    },
  });
}, []);
```

### 3.2.3 Interface gráfica

A interface gráfica é responsável por exibir os dados da aplicação, como base para o projeto a escolha foi utilizar o framework Material-UI. O Material-UI é um framework que implementa o Material Design da Google para o ReactJS.

O framework disponibiliza componentes de interface como botões, tabelas, gráficos, etc. O que facilita o desenvolvimento e a manutenção da aplicação.

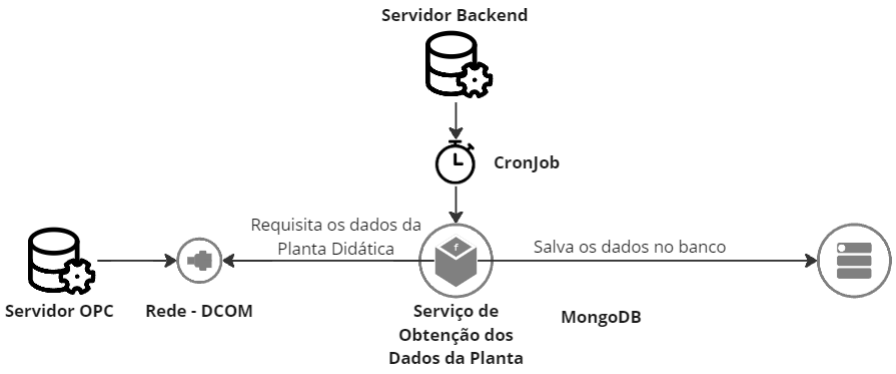
## 3.3 Serviços

Para a o servidor obter os dados de estado da Planta Didática PD3 foi desenvolvido um serviço em Javascript que utiliza uma biblioteca de DCOM modificada para o NodeJs.

No servidor backend tem-se a configuração de uma CronJob, que é uma aplicação

que é executada em um intervalo de tempo pré-definido (no sistema está configurado para executar a cada um segundo). Na Figura 10 é demonstrada a arquitetura de obtenção dos dados da planta.

Figura 10 – Arquitetura de Obtenção dos Dados da Planta.




Essa aplicação é responsável por executar o serviço de obtenção de dados da planta. Após a obtenção dos dados, o servidor envia os dados para o frontend através do servidor WebSocket por meio de um evento e salva os dados no banco de dados para serem utilizados posteriormente.

Algoritmo 3.8 – Código de configuração da CronJob.

```
1  Cron("*/1 * * * * *", async () => {
2    try {
3      const data = await getPlantState(opcSyncIO, clientHandles, serverHandles);
4      const formattedData = formatPd3DataDB(data);
5      await repositories.sensorMeasureRepository.createMany(formattedData);
6      const dtos = formatPd3DataDTO(data);
7      io.emit("sensor_data", dtos);
8    } catch (e) {
9      console.log(e);
10   }
11 });
```

## 3.4 Autenticação

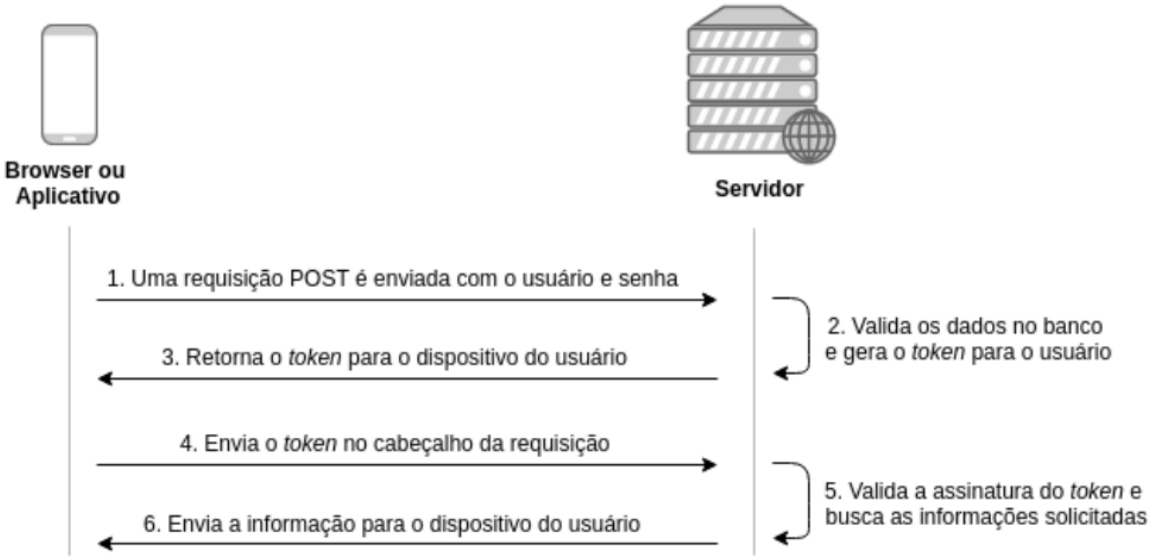
Para a autenticar os usuários necessita-se da criação de uma rota customizada, a tecnologia de *JSON Web Token (JWT)* no *backend* e a criação de um contexto no *frontend*. Uma página dedicada para o login de usuários também foi desenvolvida.

As senhas são criptografadas utilizando o algoritmo de *hash SHA-256*. Permitindo que sejam armazenadas de forma segura no banco de dados.

Vale notar que toda requisição, com exceção da requisição de login, deve conter um token de autenticação no cabeçalho da requisição. Isso se faz necessário para que o servidor

possa identificar o usuário que está realizando a requisição. Na Figura 11 é demonstrado o diagrama de fluxo de autenticação e validação de token.

Figura 11 – Diagrama de autenticação e validação de token.

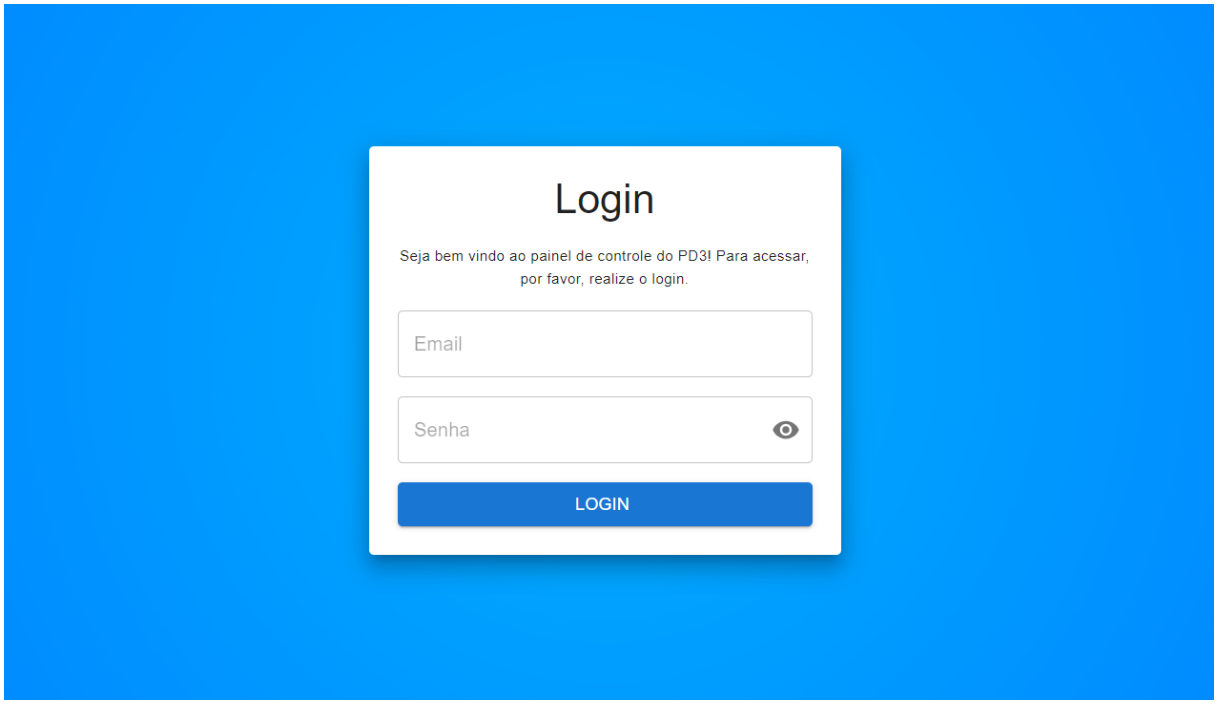



Fonte: (MONTANHEIRO; CARVALHO; RODRIGUES, 2017).

# 4 Resultados e Discussão

A seguir será demonstrado os resultados obtidos com a implementação do sistema descrito na seção anterior. A primeira parte do sistema é a tela de login, por onde o usuário irá acessar o sistema. A Figura 12 mostra a tela de login do sistema.

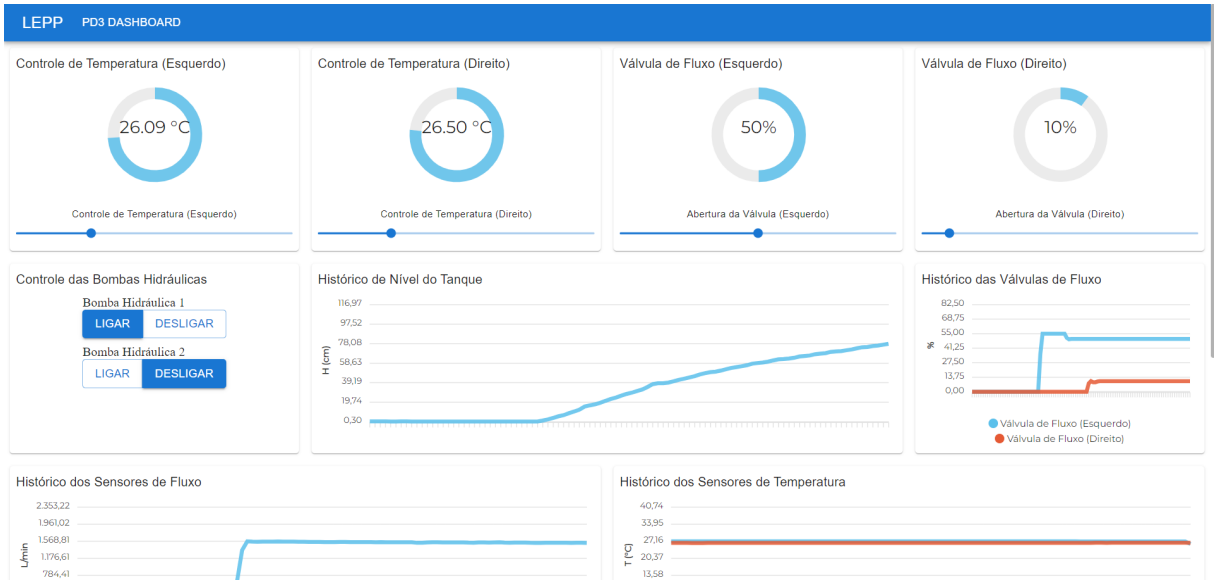
Figura 12 – Tela de login do sistema.




Vale notar que somente usuários previamente cadastrados no sistema podem acessá-lo, sendo uma escolha de segurança para evitar que pessoas não autorizadas o acessem.

Após o login o usuário é redirecionado para a tela inicial do sistema, onde é possível visualizar os dados da Planta Didática PD3, como os sensores de temperatura e vazão, status das bombas hidráulicas, entre outros. A Figura 13 mostra a tela inicial do sistema.

Figura 13 – Tela inicial do sistema.




Os primeiros componentes são o que é definido como *Gauge* e *Switch*. O *Gauge* é um componente que mostra o valor de um sensor em tempo real. O *Switch* é um componente que mostra o status de um atuador como a bomba hidráulica

Esse tipo de componente proporciona uma visualização concisa e de fácil entendimento para o usuário, pois ele consegue visualizar o valor do sensor e o status do atuador em um único lugar. Permitindo também modificar o valor do atuador.

No sistema desenvolvido, temos as seguintes *Gauges* e *Switches*, além da demonstrada acima:

Figura 14 – Controle de Temperatura (Esquerdo).

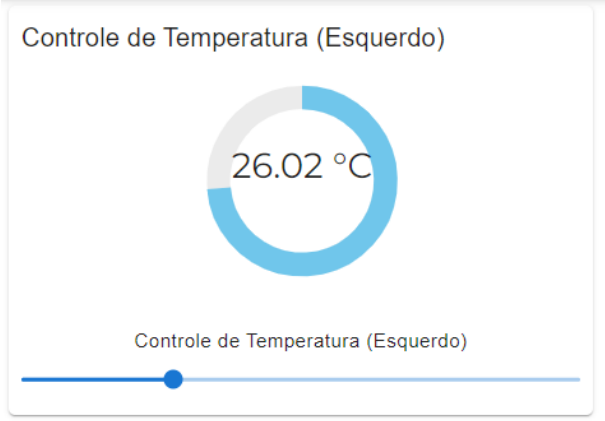


Figura 15 – Controle de Temperatura (Direito).

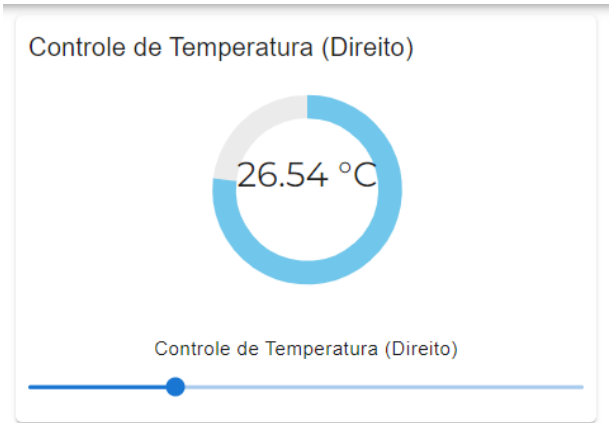



Figura 16 – Válvula de Fluxo (Esquerda).

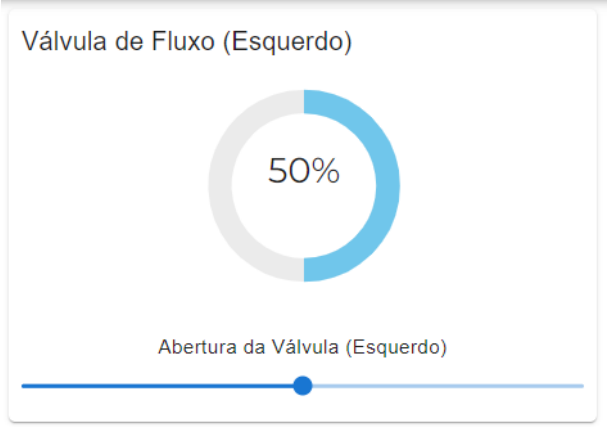



Figura 17 – Válvula de Fluxo (Direita).

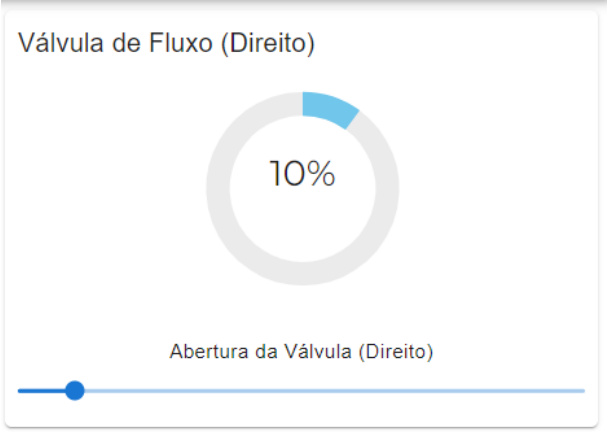
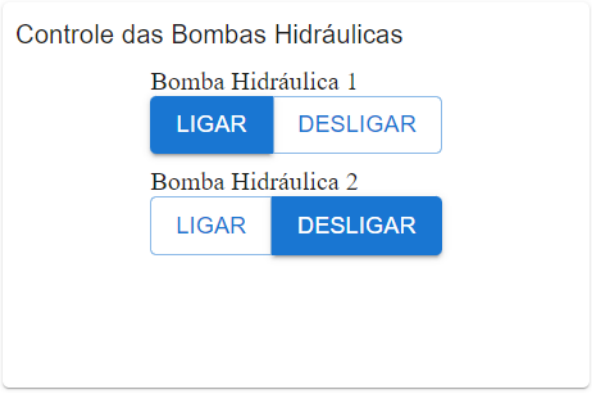
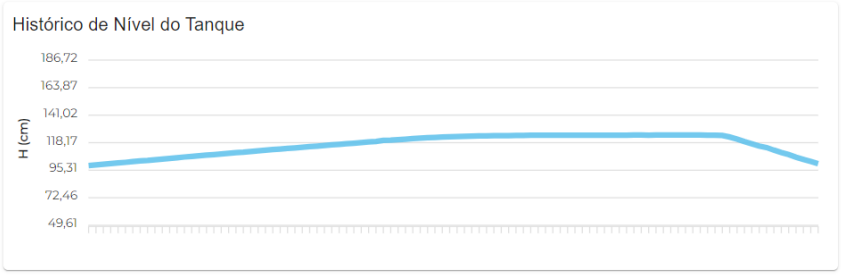


Figura 18 – Controle das Bombas Hidráulicas.




A seguir, é mostrado o componente *Chart*, que é um componente que mostra o valor de um ou mais sensores em um gráfico, como demonstra a Figura 19.

Figura 19 – Componente *Chart*.




As linhas vermelhas são os sensores do lado direito e as azuis os sensores do lado esquerdo. Abaixo, temos mais algumas Figuras que demonstram os outros gráficos do sistema.

Figura 20 – Gráfico Histórico de Abertura das Válvulas de Fluxo.

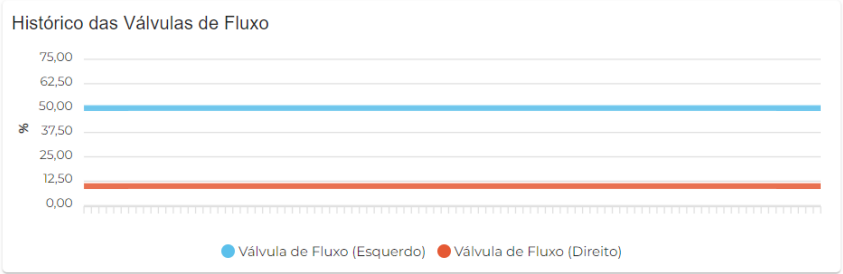


Figura 21 – Gráfico Histórico dos Sensores de Vazão.

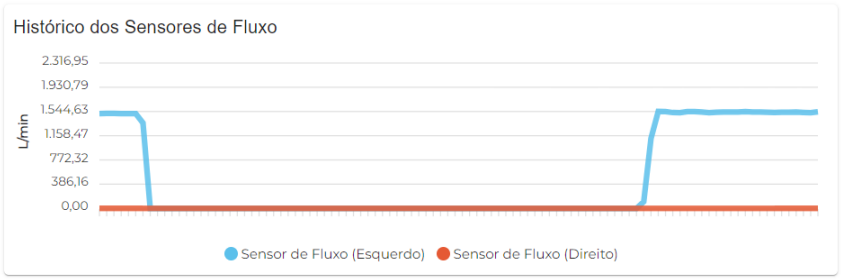
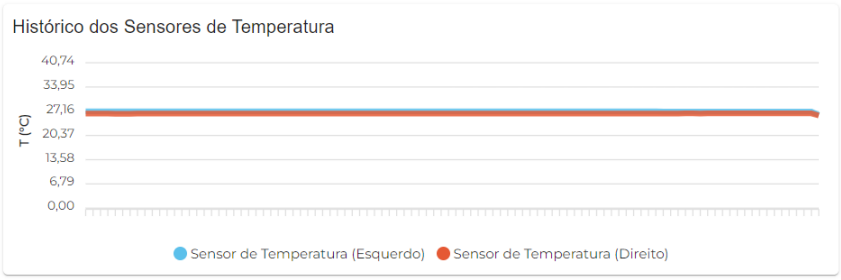


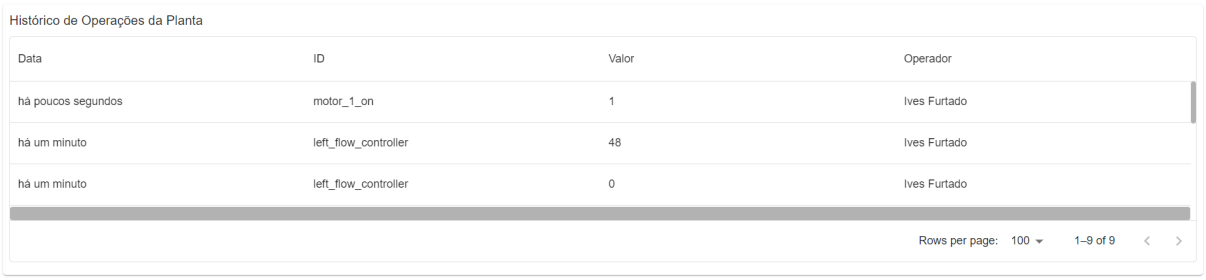

Figura 22 – Grafico Histórico dos Sensores de Temperatura.




Esse tipo de componente é muito útil para o usuário, pois ele consegue visualizar o valor de um sensor em um determinado período de tempo, podendo assim analisar o comportamento do sistema.

A seguir, é mostrado o componente *Table*, que é um componente que exibe dados tabulares. Na aplicação são demonstradas operações realizadas na planta em uma tabela, como mostra a Figura 23.

Figura 23 – Componente *Table*.



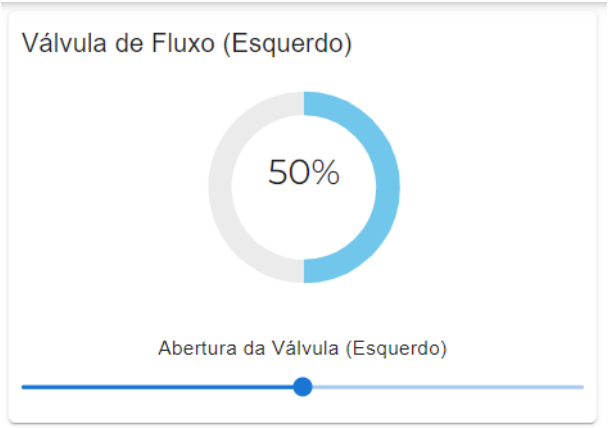

As colunas exibidas no componente acima são: *Data*, *ID*, *Valor* e *Operador*. Sendo a coluna *Data* o tempo em que ocorreu a interação e a coluna *ID* o identificador do sensor ou atuador. A coluna de *Valor* é que foi enviado para a planta didática utilizando o seu padrão de dados e a coluna de *Operador* o usuário que realizou a operação.

O histórico de operações permite ao usuário distinguir sua operação e a operação de outros.

Demonstrado os componentes que integram o sistema. A seguir será mostrado o sistema em funcionamento.

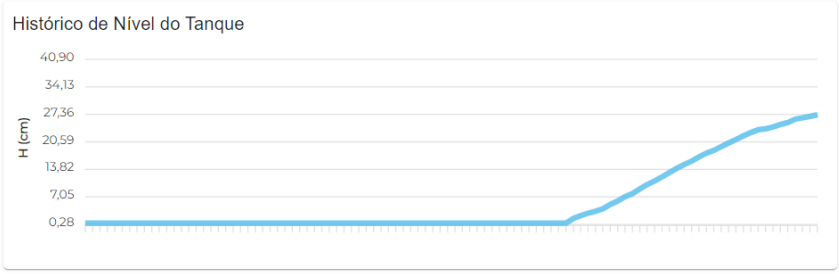
Primeiro é alterada a abertura da válvula de fluxo da esquerda para 50%, como mostra a Figura 24.

Figura 24 – Válvula de Fluxo (Esquerda) em 50%.



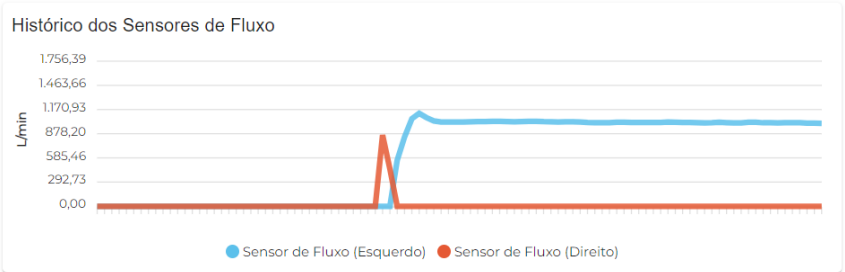

Com a bomba hidráulica 1 ligada é possível observar o sensor de nível de água reagindo a essa alteração, como mostra a Figura 25.

Figura 25 – Gráfico de Nível do Tanque com Válvula de Fluxo da Esquerda em 50%.




Outro gráfico que também é alterado é o gráfico de vazão, como mostra a Figura 26.

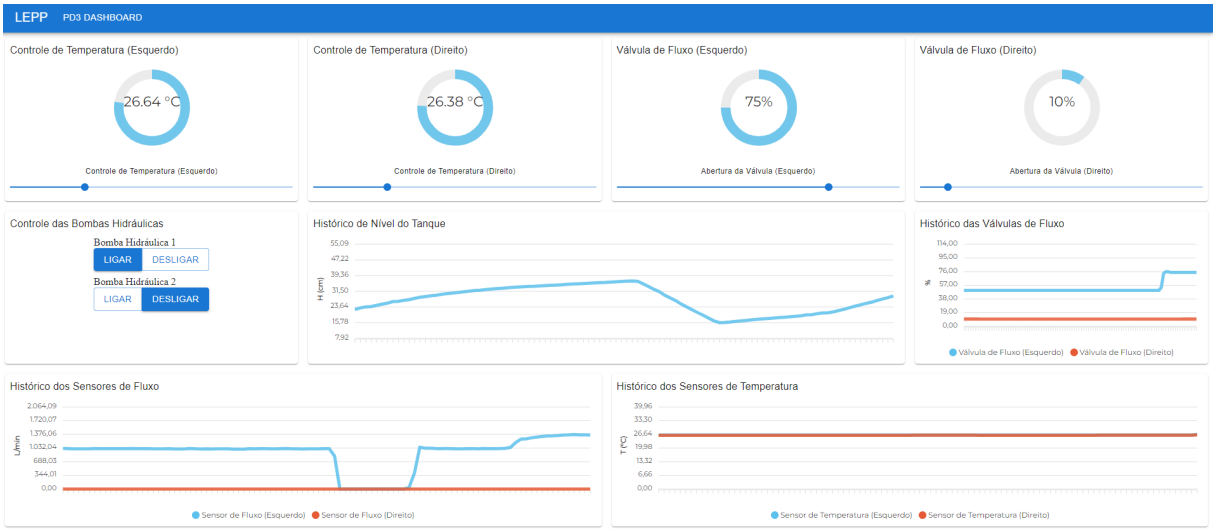
Figura 26 – Gráfico de Vazão com Válvula de Fluxo da Esquerda em 50%.



Isso demonstra o sistema funcionando em tempo real, com as mensurações e operações atualizando conforme o usuário realiza suas ações.

Abaixo, na Figura 27, tem-se a visão geral do sistema para uma abertura de válvula em 75%.

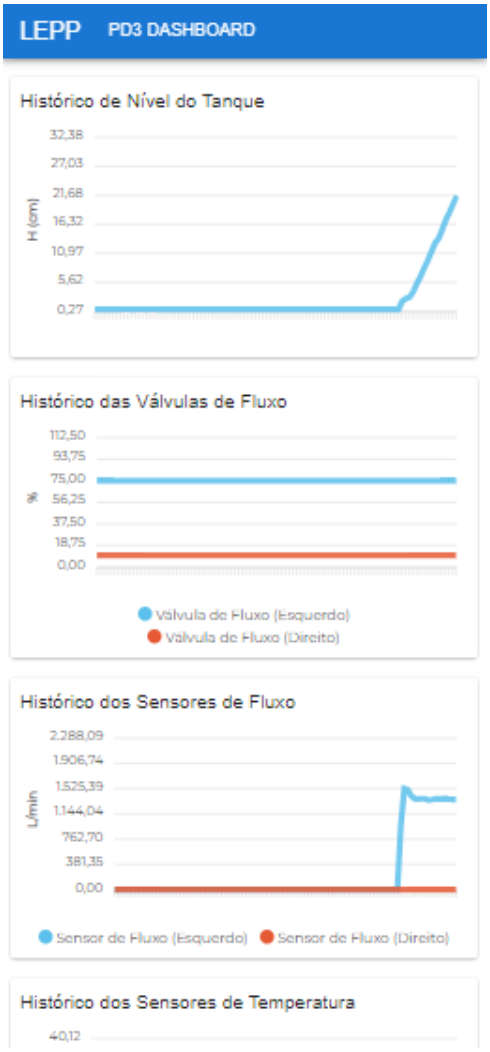
Figura 27 – Visão Geral do Sistema com Válvula de Fluxo (Esquerda) em 75%.




Por fim, uma parte muito importante do sistema é a responsividade, permitindo que o usuário de qualquer dispositivo possa acessar o sistema e obter uma visualização adequada.

A Figura 28 mostra o sistema em funcionamento em um dispositivo móvel.

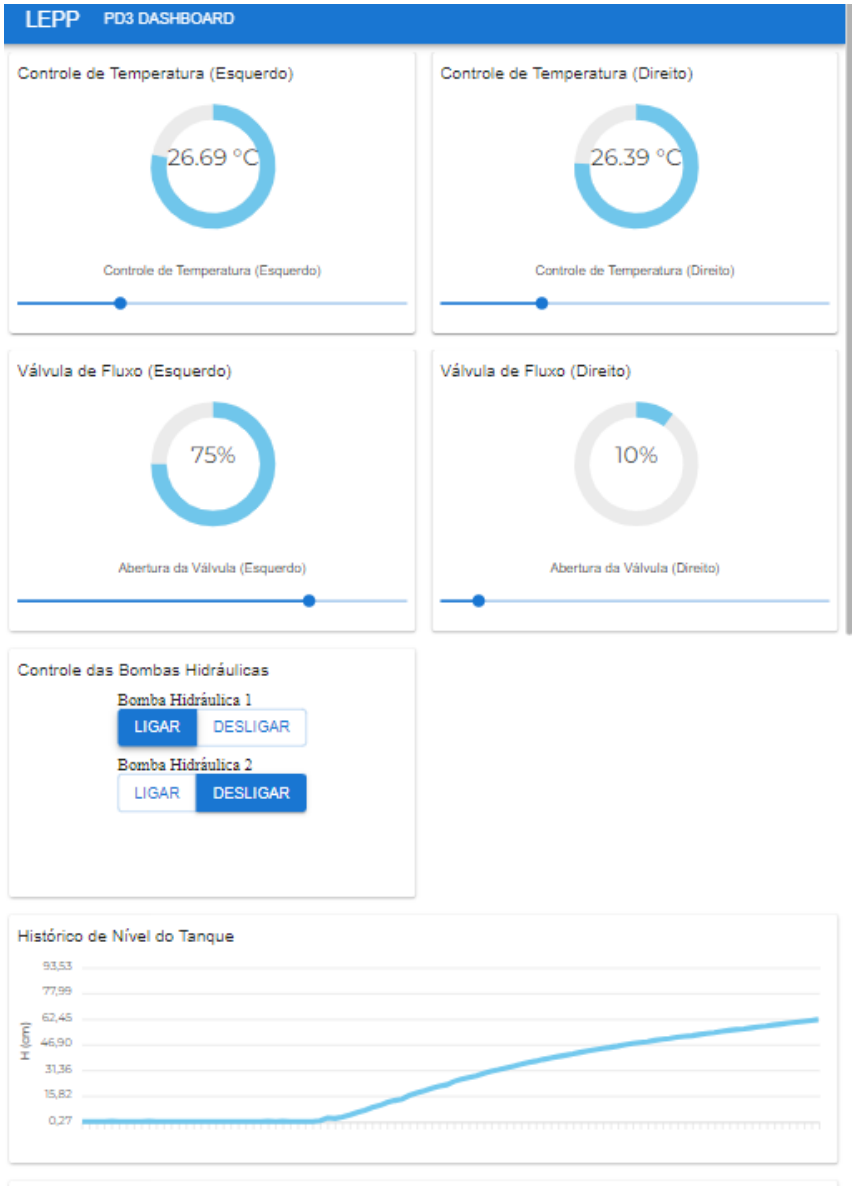
Figura 28 – Sistema em funcionamento em um dispositivo móvel.




O sistema se adapta ao tamanho da tela do dispositivo, permitindo que o usuário possa acessar o sistema de qualquer lugar e a qualquer momento.

A Figura 29 mostra o sistema em funcionamento em um dispositivo *tablet*.

Figura 29 – Sistema em funcionamento em um dispositivo *tablet*.



# 5 Considerações Finais

O sistema de controle, desenvolvido baseado em *Javascript* e *Typescript*, demonstrou a viabilidade de utilizar essas tecnologias modernas em aplicações de automação industrial. Trazendo maior flexibilidade e facilidade de integração entre o servidor *backend* e o *frontend* com os sistemas de monitoramento e controle.

Com o emprego de um *framework* como o Express, é possível obter vantagens de sua simplicidade e ampla aceitação no mercado, favorecendo o processo de desenvolvimento e a futura manutenção do código. A escolha do MongoDB, como sistema de armazenamento de dados, se alinhou perfeitamente com os requisitos do sistema, proporcionando o armazenamento adequado dos dados da planta e dos usuários de maneira simplificada e eficiente.

A aplicação da CronJob para a leitura periódica dos dados da planta provou ser uma solução completa para a aquisição dos dados necessários para o sistema de monitoramento e controle, garantindo atualizações constantes e confiáveis do estado operacional da planta.

A interface de usuário desenvolvida com ReactJS e Material-UI entregou uma experiência de usuário agradável e intuitiva, favorecendo a interação eficiente do operador com a aplicação.

Finalizando, o projeto foi cuidadosamente elaborado para servir como uma sólida fundação para iniciativas futuras, pavimentando o caminho para pesquisas e desenvolvimentos subsequentes com a planta didática PD3. A plataforma criada oferece uma estrutura de dados bem definida e um framework de trabalho expansível, incentivando a exploração de novas funcionalidades e a implementação de melhorias contínuas. Desta forma, contribui-se para a formação de um corpo de conhecimento capaz de impulsionar a inovação e a otimização contínuas na área de automação e controle de processos.

# Referências

ACKERMAN, W. R. B. W. J. Understanding supervisory systems. *IEEE Computer Applications in Power*, IEEE, p. 37–40, 1992. Citado na página 11.

ALTUS. *Como o sistema supervisório ajuda na produtividade de uma empresa?* 2017. <https://www.altus.com.br/post/528/sistema-supervisorio-produtividade-empresa/> (05/11/2023). Citado na página 11.

DUBEY, A. Enhancing real time communication and efficiency with websocket. *International Research Journal of Engineering and Technology (IRJET)*, 2023. Citado na página 16.

EXPRESS. 2010. <https://expressjs.com/> (10/11/2023). Citado na página 15.

FIELDING, R. T. *Architectural Styles and the Design of Network-based Software Architectures*. Tese (Doutorado) — University of California, Irvine, 2000. Citado na página 16.

FONSECA, M. de O. Comunicação opc – uma abordagem prática. In: ASSOCIAçãO BRASILEIRA DE METALURGIA E MATERIAIS. *VI Seminário de Automação de Processos*. [S.l.], 2002. Citado na página 14.

MICROSOFT. *DCOM Technical Overview*. 2014. <https://learn.microsoft.com/en-us/previous-versions//cc722925(v=technet.10)> (10/11/2023). Citado na página 15.

MONGOOSE. 2013. <https://mongoosejs.com/> (10/11/2023). Citado na página 15.

MONTANHEIRO, L. S.; CARVALHO, A. M. M.; RODRIGUES, J. A. Utilização de json web token na autenticação de usuários em apis rest. *XIII Encontro Anual de Computação*, 2017. Citado na página 26.

REACT. 2013. <https://reactjs.org/> (10/11/2023). Citado na página 15.

RIBEIRO, F. P. et al. Desenvolvimento de um sistema de controle e automação para aplicação didática. *XIX Congresso Brasileiro de Automática*, 2012. Citado na página 14.

ROUTER, O. *What is MongoDB? The NoSQL database explained easily*. 2021. <https://www.opc-router.com/what-is-mongodb/> (15/11/2023). Citado na página 16.

SAGIROGLU, S.; SINANC, D. Big data: A review. 2013. Citado na página 10.

SAMPAIO, A. C. S. *Supervisão De Malha De Controle De Temperatura E Estudo De Estratégia De Controle Feedfoward Em Uma Planta Didática Da Smar*. Dissertação (Monografia) — Universidade Federal de Viçosa, 2013. Citado 2 vezes nas páginas 13 e 14.

SCHWAB, K. *A quarta revolução industrial*. São Paulo: Edipro, 2016. Citado na página 10.

SMAR. 2023. <https://www.smar.com/pt/sobre-a-smar> (10/11/2023). Citado na página 11.

SOCKET.IO. 2014. <https://socket.io/> (11/11/2023). Citado na página 19.

TECHTARGET. *REST API (RESTful API)*. 2020. <https://www.techtarget.com/searchapparchitecture/definition/RESTful-API> (16/11/2023). Citado na página 16.

THOMESSE, J.-P. Fieldbus technology in industrial automation. *Proceedings of the IEEE*, p. 1073–1101, 2005. Citado na página 12.

USANDOPY. *O que é MVC? Entenda arquitetura de padrão MVC*. 2023. <https://www.usandopy.com/pt/artigo/o-que-e-mvc-entenda-arquitetura-de-padrao-mvc/> (09/11/2023). Citado na página 18.

ZEILMANN, R. P. *Uma Estratégia para Controle e Supervisão de Processos Industriais via Internet*. Dissertação (Dissertação de Mestrado) — Universidade Federal de Santa Catarina, 2002. Citado na página 13.