INSTITUTO POLITÉCNICO DE BRAGANÇA
Escola Superior de Tecnologia e Gestão

Data lake for aggregation of production data and visualization tools in the stamping industry

Leonardo Leite Meira dos Santos - 54363

Work guided by:

Prof. Paulo Alves

Prof. Kecia Marques

Master in Engineering informatics

2022-2023

Data lake for aggregation of production data and visualization tools in the stamping industry

Final Project Report of the Engineering Informatics Project submitted to the School of Technology and Management at the Polytechnic Institute of Bragança.

Leonardo Leite Meira dos Santos - 54363

2022-2023

I certify that i have read this dissertation and that, in my opinion, is appropriate in content and form as a demonstrator of the developed work.

Leonardo Leite Meira dos Santos - 54363

Abstract

In this project, the main objective is to develop a system for monitoring industrial sensors, specifically for the stamping industry. The work is motivated by the need for real-time monitoring systems to track machine operation in production, thus enabling data-driven management. The system was implemented using Python on the backend with FastAPI for the API, MongoDB for data storage, and NextJs for the dashboard where the information is displayed.

The results indicate that the system is capable of monitoring, analyzing, and presenting sensor data in real-time, with an alert mechanism that triggers notifications based on predefined parameters. The state of the art was reviewed to better understand emerging techniques and technologies in related areas, such as industrial Internet of Things (IoT), big data, and real-time data analysis.

The current implementation, although simpler compared to the solutions found in the literature, served as a valid proof of concept and is highly adaptable for future iterations based on feedback from the real production environment. It is concluded that the developed system meets the initial requirements and offers a certain degree of flexibility to adapt to other contexts and make future enhancements.

The application of Artificial Intelligence (AI) for data analysis and predictive maintenance of machines are likely directions for research and development of future work.

Keywords: Industrial Internet of Things (IIoT), Real-time Monitoring, Sensor Data Analytics, Data Lake

Chapter 1

Introduction

1.1 Framework

In the contemporary industrial scenario, the constant pursuit of efficiency and innovation has become an essential pillar for the competitiveness and financial sustainability of companies [1]. As technology advances, companies face pressure to remain competitive in the market [2]. In this context, monitoring and optimizing machines in production lines become crucial to ensure effective operation and prevent potential downtime or operational failures.

However, the tradition of industrial practices is often characterized by manual inspections and outdated monitoring systems that fail to provide real-time information or in-depth analysis of machine performance. This technological gap can result in significant losses in terms of production, financial resources, and machine maintenance, considering that investment in technology and monitoring can make companies more financially efficient [3].

In addition, with the increasing integration of IoT systems and the proliferation of advanced sensors, there is an immense amount of data being continuously generated, demanding more efficient processing [4]. However, without the proper infrastructure to store and analyze this data, companies may find themselves overwhelmed, unable to extract meaningful insights that could inform strategic and operational decisions.

In this market context, a stamping industry company sought to build projects that enable the sensorization of their machines, storage and processing of data, and visualization of this information, in order to become more competitive, efficient, and profitable. These projects were grouped within the "ATTRACT - Digital Innovation Hub for Artificial Intelligence and High-Performance Computing - Project: 101083770 — ATTRACT — DIGITAL-2021-EDIH-01" context, with funding from the "Digital Europe Programme (DIGITAL) - DIGITAL-2021-EDIH-INITIAL-01 — Initial Network of European Digital Innovation Hubs", referred to as the `Attract Project` in this document.

1.2 Objectives

Given the previous framework, the identified need is to develop a robust system that can receive data from sensors that collect real-time data from machines, store them efficiently in a data lake, and present them through a dashboard, transforming the way the company monitors and optimizes its production line, ensuring efficiency and a proactive approach to maintenance and industrial management. This system would not only provide real-time information about the status and performance of the machines, but it would also allow for historical analyses, helping managers and technicians to identify trends and failures, as well as optimize production, minimizing production losses and maximizing financial gains.

Chapter 2

State of the art

2.1 Data storage and big data

In the context of data storage and big data, the importance of managing and analyzing large data sets has been emphasized in various studies. One of these studies is the work on the FastQ Quality Control (FQC), a software designed to manage information from FASTQ files [5]. Developed in Python and JavaScript, the FQC aggregates data and generates metrics that are displayed on a dashboard. The software is capable of processing single-end or paired data, and can process batch files based on a specified directory.

It is important to note that the software allows customization, users can configure Key Performance Indicator (KPI), charts, and other dashboard elements. The data processing layer of the FQC bears similarities with industrial sensor monitoring systems of this project, mainly in aspects of data access and processing. It operates by accessing a directory, processing the information, and making it available for later use, and also supports the execution of small batch functions.

The dashboard built by the FQC offers a variety of visualizations, including line charts, bar charts, and heat maps, among others. These visualizations are dynamically generated and can be configured using JavaScript Object Notation (JSON) files [6], providing a flexible and user-friendly interface for data management and analysis.

Another contribution to the field of data storage and big data is the work [7], which focuses on predicting product quality through a data-driven approach. The KPI are identified to serve as highly relevant state variables for product quality. Traditionally, these variables are measured through offline laboratory analyses, which introduces latency into the system.

The AI component of the system is layered, where the lower layer deals with both categorized and uncategorized data to train and test the AI models. A Gaussian distribution is applied in the second layer to process the data, which are then fed into a semi-supervised training layer. The final layer provides the results of the predictions, thus closing the cycle.

Case studies presented in the article demonstrate the implementation of this system in industrial mineral processing. Although the method successfully addresses the issue of unmarked records, it requires a high degree of continuity in industrial systems, which is identified as a limitation.

The work in [7] provides insights into the use of data for predictive quality control in industrial environments. The layered AI model and the focus on KPI are especially relevant for the future development of the system, which can use the analyses carried out over time as training data for the AI.

Still within the scope of data storage and big data, Predictive Maintenance (PdM) becomes a very relevant strategy, both in the context of this project and in semiconductor manufacturing, as explained in [8]. The article [8] presents a multiple classifier approach to PdM, aiming to minimize downtime and associated costs. Three main categories for maintenance management are identified: Run to Failure, Preventive Maintenance, and Predictive Maintenance. The latter is emphasized for its ability to leverage historical data, forecasting algorithms, statistics, and engineering methods.

The paper uses several trained classification modules with different forecasting horizons to offer various performance trade-offs. Two main indicators are identified to reduce total operational costs: the frequency of unexpected breakdowns and the amount of unutilized lifespan. Linear regression is used as a statistical method for forecasting.

The approach is particularly relevant for systems that require real-time data analysis and efficient data storage, such as industrial sensor monitoring systems. It addresses the limitations associated with the lack of continuous data feed in industrial systems and offers a cost-based decision-making system for maintenance management.

Therefore, the article provides insights into the application of machine learning for predictive maintenance, especially in semiconductor manufacturing. The methodology can be particularly beneficial for industrial environments where minimizing downtime and operational costs are essential, and could be a logical evolution of this project, given the storage of historical data received in the system that can be used for predictive maintenance.

2.2 Real-time Monitoring

In the context of industrial sensor monitoring and real-time data analysis, the paper [9] is relevant as it offers a comprehensive exploration of precision agriculture techniques, with a particular focus on IoT-based intelligent irrigation systems. These systems face similar challenges to those in industrial environments, such as latency, bandwidth limitations, and intermittent internet connectivity.

Edge computing (fog computing), as discussed in the article, emerges as a cutting-edge solution to these challenges. It aims to save energy and bandwidth, reducing failure rates and delays. This is particularly relevant for real-time data analysis and alert systems in industrial sensor monitoring. The Fog of Everything architecture, introduced in the article, offers a multi-layered approach that could be adapted for industrial applications aiming to improve service quality and efficiency in data storage. The methodologies and technologies discussed in the article provide ways of system organization for data storage, big data, and sensor monitoring systems.

The article [10] explores the integration of Big Data Analysis (BDA) with Industrial Internet of Things (IIoT), focusing on real-time data analysis, data management and storage, aspects that connect with the aim of the dissertation to develop a robust system for monitoring industrial sensors.

The article's discussion on real-time analysis can guide the development of the Data Reception Module, and the data processing module for historical data analysis in this project. In addition, the article's insights on data management and storage can offer paths to optimize the performance of the MongoDB database if necessary.

Although the article does not specifically discuss alert systems, its categorization of analysis techniques into descriptive, prescriptive, predictive, and preventive procedures can provide a framework for generating alerts based on predefined parameters. Moreover, the article's focus on interoperability and integration in IIoT systems may offer guidelines for the effective design and integration of the various modules in this project, such as the Database, Data Receiving Module, and API.

An important project to highlight in real-time data monitoring is presented in the article [11]. This explores the use of Internet of Things (IoT) technologies to enhance vehicle safety. The article introduces an Obstacle Detection and Alert System (ODAS) System designed to identify obstacles on the road and alert drivers in real-time. The system uses embedded algorithms that detect obstacles based on various vehicle parameters, such as speed and steering angle. Once an obstacle is detected, its location is stored locally and sent to a cloud server periodically. The cloud server processes these data, confirms the presence of a real obstacle, and sends this information back to the vehicle's alert system, which provides audible and visual alerts to the driver.

This article's approach can provide important information for the design and implementation of real-time alert systems in an industrial environment, specifically, the use of IoT for data collection and cloud processing can be adapted to enhance the real-time analysis capabilities of this system. The article also discusses the challenges associated with implementing such a system, including data security and latency, which are important points to consider when transmitting real-time data and generating alerts about machine operation.

In the field of data storage and big data, machine learning algorithms have been widely studied for their ability to analyze and interpret large data sets. A review conducted in

[12], elucidates various statistical and machine learning techniques pertinent to feature selection and data analysis. Methods such as Variance Analysis and Chi-Square tests are highlighted for their utility in identifying statistically significant features in data sets. These techniques are particularly relevant for systems that require real-time data analysis and decision-making, such as industrial sensor monitoring systems.

In addition, the article discusses the application of machine learning algorithms in various areas, potentially including industrial settings and the Internet of Things (IoT). Although the article does not specifically delve into real-time data analysis, the algorithms and methods presented can be adapted for such purposes. For instance, machine learning algorithms can be employed to predict sensor failures or other anomalies based on historical data, thereby enhancing the robustness and reliability of industrial monitoring systems.

In this way, the methodologies and algorithms discussed in [12] provide guidance for the development of systems that require efficient data storage and real-time analysis capabilities.

Chapter 3

Methodology

3.1 Requirements Definition

The precise definition of requirements is crucial to ensure that the developed system meets the project's needs and objectives in an agile way [13]. The requirements of this project were classified into functional and non-functional categories, to ensure a complete understanding of what is expected from the system.

3.2 Data Collection and Storage Method

Within the project context, the way sensor data is collected and stored influences the system's operation, as it is from them that the entire system is structured. Thus, a protocol developed in another project within the same context of the `Attract Project` was used as a basis, which transmits all the necessary information for the context of this project. Within the system in question, the responsibility of implementing the decoder for the given protocol was assigned.

The protocol format is structured to represent the information pertinent to the machine, the type of communication, the sensor, and the meaning of the transmitted data, following the format in table 3.1.

Field	**Description**	**Details**
Machine ID (2 bytes)	Identifies the machine	**High**: Type (e.g., press) **Low**: Machine number
Type (1 byte)	Message type	1. Publish 2. Request
Sensor ID (2 bytes)	Sensor details	**High**: Quantity **Low**: Sensor number
Meaning of Data (2 bytes)	Data nature	**High**: Type **Low**: Meaning
Length (2 bytes)	Bytes count	
Data	Sensor data	Defined by (*)

Table 3.1: Data Fields and Descriptions

3.3 Technologies

- **MongoDB**: A flexible database platform chosen for adaptability in data format changes and its efficiency in handling large data contexts [14].
- **Python**: Utilized for backend development due to its versatility, vast libraries, community support, and suitability for diverse applications [15]. Enhanced with `Pipenv` for workflow optimization [16].
- **FastAPI**: Selected for backend development alongside Python for its efficiency, performance, and asynchronous processing capabilities. Offers strong documentation, data validation, and automatic API documentation through `Swagger UI` [17], [18].
- **NextJs**: Powers the frontend, enhancing React library interaction, supported by TypeScript for static typing. Material UI 5 ensures a modern UI design [19]–[23].
- **Docker**: Employed for containerization, ensuring consistency and reproducibility across platforms [24], [25].
- **NGINX**: Acts as the web server, renowned for high performance, reliability, and efficient content delivery [26].

Chapter 4

System Architecture

In this chapter, the architecture and structure adopted for the construction of the system components are discussed in detail. It is important to understand that the system was conceived as a set of modules, with each one performing specific functions, and when operated together, these modules result in the achievement of the purposes intended for the system.

The system is fundamentally structured in distinct layers, the backend, the frontend, and the database.

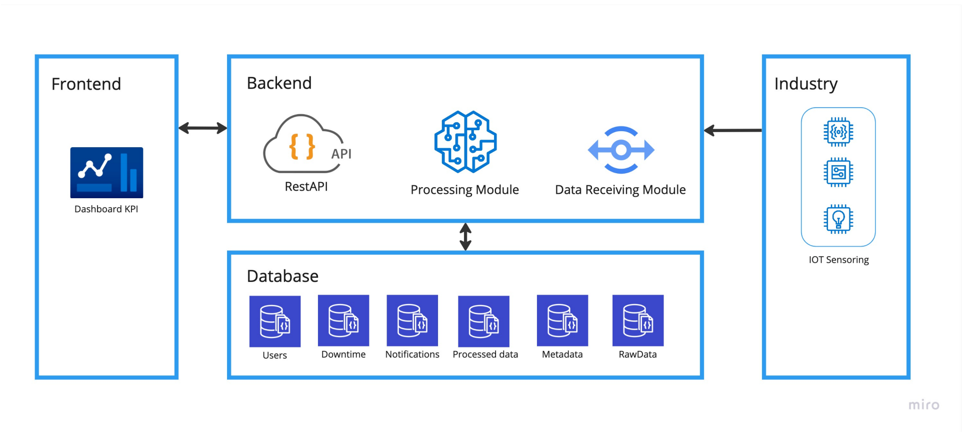


Figure 4.1: System architecture.

The backend functions as the core of the system. Its main role is to receive the data, process it according to the rules established in the requirements and user stories, and store it securely in the database. In addition to storage and processing functions, the backend is also assigned the responsibility of making these data available through an API, which can be accessed using HyperText Transfer Protocol (HTTP) methods. This API acts as an intermediary between the central logic of the system and the interfaces with which the end user interacts, the frontend. On the other hand, the frontend is characterized as the visual interface that the end user accesses. It serves as a means by which users interact with the system, sending and receiving information. This layer is designed to access, retrieve, and present the data processed and stored by the backend according to design principles and data visualization [27].

As can be seen in figure 4.1, in the industrial plant, where the machines with the sensors are located, the sensors send the data to the system, which receives them through the Data Receiving Module and stores them in the Database. The processing module accesses the stored data to perform aggregation, and the API manages access to the database, making the information available to users on the frontend.

- **Data Receive Module**: The data receiving module utilizes the 'SensorConnection' class to manage the connection with the sensor network and forward received data for proper handling. Adaptability is ensured through the 'IotSensorConnectionInterface', from which the 'IotSensorConnection' class originates, facilitating integration of various data receipts. This class, when instantiated, establishes a connection and monitors new data, forwarding it to the 'SensorsRepository' class. This class evaluates sensor data based on set parameters, deciding on alert triggers, making data available via API in real-time, and saving it in the 'Raw Data' collection of the data lake. The provision of data is managed by the 'SensorValue' class, which updates in-memory data accessed by users. This structure effectively handles the receipt, evaluation, and storage of raw sensor data.

- **Data Process Module**: The data processing module processes the raw data to deliver statistical analysis for users, executing a specific function that first establishes lists of database collections and machines sending information. Iteratively, data from each machine undergoes statistical analysis using the Box Plot method, a graphical representation showcasing data variation through quartiles. This method offers insights into data centrality, dispersion, and potential outliers, providing a comprehensive understanding of data trends and anomalies.

- **API**: The API is modularized into specific sub-modules, ensuring each segment has a distinct responsibility. Within these modules, there's a three-tiered structure: the *controller* layer manages HTTP requests, the service layer processes information and applies business rules, and the repository layer connects with the database for data access.

- **Frontend**: The frontend, built using *Next.js* [19], adheres to the framework's predefined structure. Routes are contained within the `pages` folder, while base layouts are in the `layouts` folder. Built upon *React* components [20], these layouts ensure reusability throughout the application. The use of *Typescript* [21] streamlines data structuring through models in the `types` folder, improving development efficiency. To manage component data, the React *Context API* centralizes information sharing, leading to a more organized architectural approach.

Chapter 5

System Characteristics from a Functional Point of View

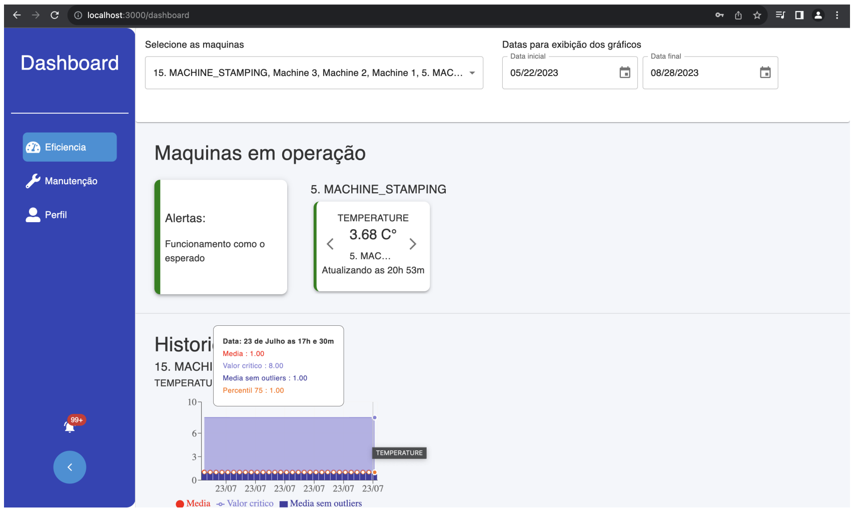
After detailing the chosen architecture for the system and the technologies, this chapter is focused on the functionalities that make up the application. The project encompasses several components, including a frontend layer, a backend layer, a data receiving module, a data processing module, and the database. Therefore, this chapter aims to detail the operation of each system functionality, providing a comprehensive view of how each component interacts and contributes to the operation of the system as a whole.

Each section of this chapter will be dedicated to a specific functionality, examining its role and operation in depth, as well as the interaction between different system components for its realization.

5.1 Real-time Monitoring

On the user interface dashboard, individual cards corresponding to each monitored machine are presented. In figure 5.1, this page can be seen with a card for the machine `5.MACHINE_STAMPING`. On each card, sensor information is displayed with the possibility of navigating between them with a directional arrow. The card can be seen in figure 5.2.

The acquisition of this real-time data is carried out through a continuous data stream.



Selecione as maquinas

15. MACHINE_STAMPING, Machine 3, Machine 2, Machine 1, 5. MAC...

Datas para exibição dos gráficos

Data inicial: 05/22/2023

Data final: 08/28/2023

Maquinas em operação

Alertas:

Funcionamento como o esperado

5. MACHINE_STAMPING

TEMPERATURE
3.68 C°
5. MAC...
Atualizando as 20h 53m

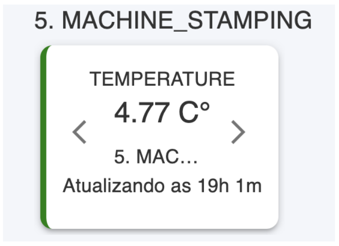
Histori

15. MACHI

TEMPERATU

Data: 23 de Julho as 17h e 30m
Media : 1.00
Valor critico : 8.00
Media sem outliers : 1.00
Percentil 75 : 1.00

Figure 5.1: Dashboard with real time and graph data.



5. MACHINE_STAMPING

TEMPERATURE
4.77 C°
5. MAC...
Atualizando as 19h 1m

Figure 5.2: Card with machine data.

The frontend of the application makes a request to the endpoint `iot/realtime`, which returns this real-time data stream.

In the backend layer, the data stream is constantly fed by the `SensorValue` class, which in turn, is updated by the data receiving module. Upon receiving new sensor readings, the module performs appropriate validations before updating the values in the `SensorValue` class. Once updated, the API accesses these new values and inserts them into the data stream transmitted to the connected user.

5.2 Alerts and notifications

The main objective of this feature is to monitor the performance of the machines in real time and issue alerts and notifications to users if inappropriate operating conditions are detected. This allows corrective actions to be taken immediately.

Whenever a new sensor reading is received by the data receiving module, a validation is performed to check if the machine is operating within acceptable parameters. These parameters are obtained through the system metadata, loaded at the initialization of the API.

If a value outside the acceptable range is identified during validation, the data receiving module inserts a special marking on this reading within the `SensorValue` class. This marking is later transmitted to the frontend during the data streaming process by the **IOT Analytics** module of the API.

Upon receiving a marked reading, the frontend updates the corresponding information card to reflect the anomalous state. Specifically, within the dashboard shown in figure 5.1, the color of the card is changed to red or yellow, both on individual cards, figure 5.2, and on the general card that is used to show the overall status of the machines, figure 5.3, thus serving as an immediate visual alert for the user.

The data receiving module, keeps track of the operational state of each machine that is sending information. When a machine exits an alert state, the occurrence is recorded in the database, with the start and end date of the alert state, along with information related

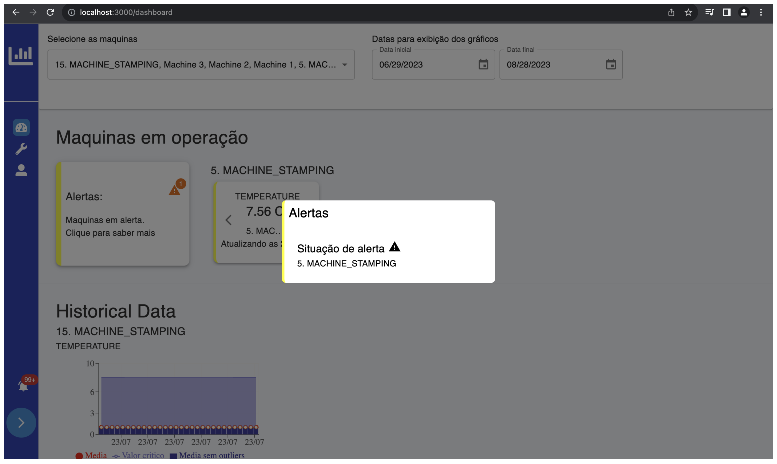


Figure 5.3: General Notifications.

to the sensor and the involved machine. Subsequently, a message is sent to the connected users using the *WebSocket* interface, informing them of the machine's malfunction during the period recorded by the system.

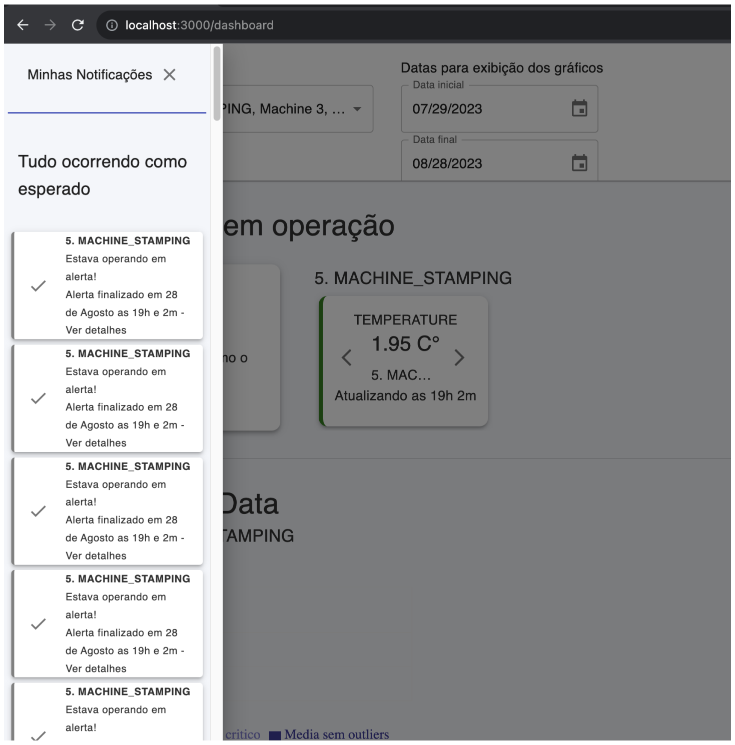
Information about alert completion can be accessed by users through notifications in the system interface, as can be seen in figure 5.4. By clicking on the notification icon in the dashboard's side menu, the user can view these notifications.

Therefore, when starting a session in the system, previous notifications are loaded for the user. In addition, any new notification generated during the user's session is transmitted in real time through a WebSocket connection established between the frontend and the backend.

5.3 Statistical Analysis of Historical Data

The data processing module is responsible for the statistical analysis of historical data. Following the system's standard parameter, every day at midnight, the raw unprocessed data is read and analyzed. The result of the analysis is then stored in the database for future queries.

The possibility to access these analytical data is provided by the API, specifically by


localhost:3000/dashboard

Minhas Notificações

Tudo ocorrendo como esperado

5. MACHINE_STAMPING
Estava operando em alerta!
Alerta finalizado em 28 de Agosto as 19h e 2m - Ver detalhes

5. MACHINE_STAMPING
Estava operando em alerta!
Alerta finalizado em 28 de Agosto as 19h e 2m - Ver detalhes

5. MACHINE_STAMPING
Estava operando em alerta!
Alerta finalizado em 28 de Agosto as 19h e 2m - Ver detalhes

5. MACHINE_STAMPING
Estava operando em alerta!
Alerta finalizado em 28 de Agosto as 19h e 2m - Ver detalhes

5. MACHINE_STAMPING
Estava operando em alerta!

Datas para exibição dos gráficos
Data inicial: 07/29/2023
Data final: 08/28/2023

PING, Machine 3, ...

em operação

5. MACHINE_STAMPING
TEMPERATURE
1.95 C°
5. MAC...
Atualizando as 19h 2m

Data
TAMPING

critico ■ Media sem outliers

Figure 5.4: Notification drawer.

the "IOT Analytics" module. The request to obtain this information must be authenticated.

The visual representation of these analyses is carried out through graphs that aggregate four main metrics resulting from the processing. Thus, the graph displays:

- *Ideal Value*: Represented in an area chart, it serves as an ideal operating parameter to provide a perspective relative to the other data.
- *75th Percentile*: Displayed in a line chart, this metric offers a view on the distribution of values during the aggregation period and its evolution over time.
- *Aggregation Average*: Represented in a scatter chart, this metric provides the average value of the aggregated data.
- *Average with Outlier Removal*: Illustrated in a bar chart, this metric is calculated after the removal of outlier values, as determined by the boxplot construction method.

Therefore, image 5.5 shows how these information are displayed within the dashboard.

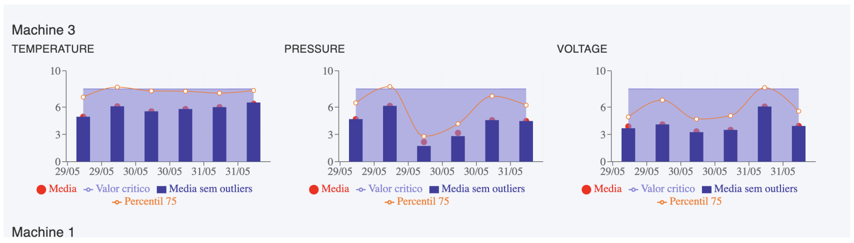
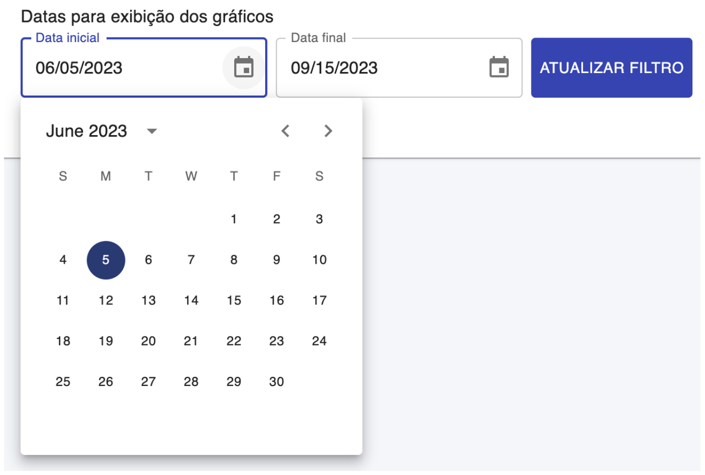


Figure 5.5: Graph example.

The visualization of these graphs can be filtered by date, allowing the selection of the start date and the end date for the data to be displayed. When these fields are changed and the "apply filter" button is clicked, a new request is sent and the data is loaded according to the specified period. By default, when the page is first loaded, the system sends a request seeking data from the last 30 days, and it is up to the user to use the filter to view a different period. The date filter can be seen in figure 5.6.



Datas para exibição dos gráficos

Data inicial: 06/05/2023

Data final: 09/15/2023

S	M	T	W	T	F	S
				1	2	3
4	5	6	7	8	9	10
11	12	13	14	15	16	17
18	19	20	21	22	23	24
25	26	27	28	29	30	

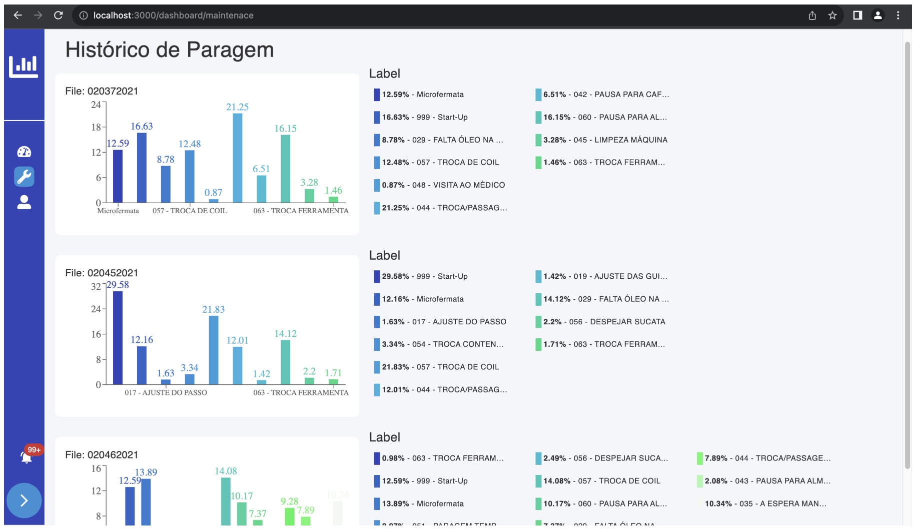
Figure 5.6: Date filter.

5.4 Display of data related to machine downtime

This feature is dedicated to the display of data related to machine downtime, previously stored in the database. The functionality aims to demonstrate how this data would be presented if it were received by the system in a manner similar to sensor data. The source of the data for the preparation of these graphs comes from three spreadsheets received at the beginning of the project development, where the information was saved in the database. This data is accessed by the frontend through the `downtime_analytics` module of the API.

The graphs generated to represent these data are presented in the form of column charts. Each chart displays a title corresponding to the spreadsheet from which the data were extracted. The chart legend indicates the percentage that each specific stop represents in relation to the total stops.

Column charts provide an effective way to compare the different machine stops, as can be seen in figure 5.7. This visualization offers a means to analyze operational efficiency, identify possible areas for improvement, and track the outcome of actions taken in relation to machine downtime and predictive maintenance.


localhost:3000/dashboard/maintenace
Histórico de Paragem
File: 020372021
Microfermata
057 - TROCA DE COIL
063 - TROCA FERRAMENTA
12.59
16.63
8.78
12.48
0.87
21.25
6.51
16.15
3.28
1.46
Label
12.59% - Microfermata
16.63% - 999 - Start-Up
8.78% - 029 - FALTA ÓLEO NA ...
12.48% - 057 - TROCA DE COIL
0.87% - 048 - VISITA AO MÉDICO
21.25% - 044 - TROCA/PASSAG...
6.51% - 042 - PAUSA PARA CAF...
16.15% - 060 - PAUSA PARA AL...
3.28% - 045 - LIMPEZA MÁQUINA
1.46% - 063 - TROCA FERRAM...
File: 020452021
017 - AJUSTE DO PASSO
063 - TROCA FERRAMENTA
29.58
12.16
1.63
3.34
21.83
12.01
1.42
14.12
2.2
1.71
Label
29.58% - 999 - Start-Up
12.16% - Microfermata
1.63% - 017 - AJUSTE DO PASSO
3.34% - 054 - TROCA CONTEN...
21.83% - 057 - TROCA DE COIL
12.01% - 044 - TROCA/PASSAG...
1.42% - 019 - AJUSTE DAS GUI...
14.12% - 029 - FALTA ÓLEO NA ...
2.2% - 056 - DESPEJAR SUCATA
1.71% - 063 - TROCA FERRAM...
File: 020462021
12.59
13.89
14.08
10.17
7.37
9.28
7.89
10.34
Label
0.98% - 063 - TROCA FERRAM...
12.59% - 999 - Start-Up
13.89% - Microfermata
2.49% - 056 - DESPEJAR SUCA...
14.08% - 057 - TROCA DE COIL
10.17% - 060 - PAUSA PARA AL...
7.89% - 044 - TROCA/PASSAGE...
2.08% - 043 - PAUSA PARA ALM...
10.34% - 035 - A ESPERA MAN...
Figure 5.7: Downtime graph.

5.5 User Profile

The user profile functionality was developed with the aim of providing users with access to their personal data stored in the system. In addition to viewing this information, this screen also allows for modifications, including password changes, email, and other personal data such as name, surname, and description.

To read and modify the displayed data, the *User* module of the API is used. This module provides endpoints that allow access and modification of the stored data, ensuring that the information is updated according to the user's interactions, as can be seen in figure 5.8.

An additional feature provided by this screen is the logout option. By selecting this option, all client-side stored information is erased, thus ensuring the user's secure exit from the system.

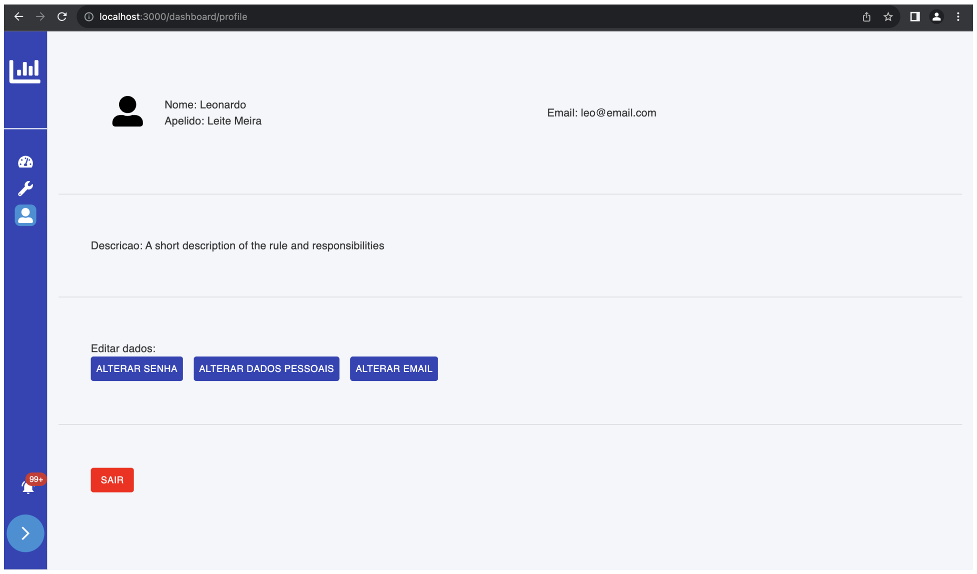


Figure 5.8: Profile page.

Chapter 6

Results and Evaluation

A series of benefits is offered by the developed system. Firstly, comprehensive monitoring of machines throughout the industrial plant is facilitated. Through this monitoring, crucial information about the operational status of each machine is displayed to employees in real time.

Additionally, the system provides the ability to take quicker actions in case of operational issues, and as a result, the downtime of machines can be reduced, mitigating losses associated with production. Consequently, it allows for timely oversight and understanding of the machinery's current state. The use of such a system not only ensures that the machinery is functioning optimally, but it also aids in the early detection of potential issues, making it an essential tool in modern industrial operations.

Another advantageous aspect lies in the evaluation of the effectiveness of the implemented maintenance measures. Through historical tracking, the system allows for the visualization of the effects of actions taken for machine maintenance, thus, more informed and effective decisions can be made quickly, extending the benefits to other machines in the plant.

The developed system features a modular characteristic, which allows the integration of new modules to expand its functionalities. A practical example of this is the possibility of adding a module intended for working with data from computer vision. This module would be capable of evaluating and monitoring the quality of production in real time, identifying defects with greater precision and efficacy. Additionally, another module that can be integrated refers to the access to historical data from the machines, providing a retrospective view and facilitating access to this information. Finally, considering the importance of maintenance, a module for predictive maintenance can be implemented. This module, by using the received data, could develop a model that understands the operation pattern of the machines, allowing more assertive and optimized interventions.

Finally, the system's historical data also contributes to the generation of insights from its analysis. This analysis can provide a new perspective for monitoring machine performance, and operational anomalies can be identified and specific corrective measures can be implemented to improve certain indicators.

Chapter 7

Conclusion and Future Work

In this project, a multifunctional system was developed for the collection, storage, processing, and visualization of data generated by sensors. Python was used for the creation of the API and the processing module, MongoDB for database management, and NextJs for the construction of the control panel, known as *dashboard*. The system architecture can be seen in 4.

The data were received through a multicast connection established by the data receiving module. Once received, the data were immediately subjected to a preliminary analysis to identify any condition that could trigger an alert. If an alert was generated, users were notified, allowing for quick and effective interventions.

For the analysis of historical data, the BoxPlot method was employed in the processing module. This analysis aimed at identifying patterns and anomalies in the data collected over time, providing valuable insights for the operation and maintenance of the monitored machines.

The API, played a crucial role in the system, managing access to the data. Security was ensured through the implementation of JSON Web Token (JWT) authentication, and several *endpoints* were developed to allow effective access to the data.

The *frontend* was responsible for displaying real-time information, generated alerts, and processed data in the form of charts.

7.1 Suggestions for Future Work

7.1.1 Use of artificial intelligence for prediction

Given that the data read by the sensors are stored in the system, they have the potential to reveal significant information about the operation of the machines. The application of AI techniques to the collected data was identified as the main functionality for the evolution of the system, which can become a robust tool for predictive maintenance. By applying machine learning algorithms to the stored data, predictive models can be generated that anticipate failures or inefficiencies in industrial equipment.

The implementation of a predictive maintenance system based on AI could confer a significant competitive advantage to the company, not only would it improve operational efficiency, but it would also optimize the allocation of resources for maintenance, resulting in cost reduction and increased productivity. Therefore, the future exploration of AI for the analysis of stored data is strongly recommended to enhance the effectiveness of the system under study.

7.1.2 Monitoring and Logs

Comprehensive system monitoring and activity log maintenance are crucial aspects for system sustainability and scalability. The absence of a well-structured log system can result in difficulties in identifying and resolving issues that may arise during real-time system operation. In this context, three main areas are identified where monitoring and logs could provide valuable insights for continuous improvement.

Moreover, it would be advantageous to maintain a comprehensive record of data transactions between the sensors and the data receiving module. These logs could include information such as the date and time of the transaction, sensor identification, and any anomaly or failure during the receiving process. This would facilitate the verification of the integrity of the received data and assist in early detection of potential hardware or network connectivity issues.

Log for Statistical Analysis

The data processing module, responsible for statistical analysis, would also benefit from a log system. Details about the execution of the BoxPlot or any other statistical analysis could be recorded. This includes information such as the number of data points analyzed, any detected outliers, and the time taken for the analysis execution. With this information, it would be possible to further refine the analysis algorithm and identify areas for optimization.

7.1.3 Dynamic Parameters for Alerts

In the current version of the system, the parameters responsible for triggering alerts are defined statically, embedded directly in the source code. This approach, although functional, presents limitations in terms of flexibility and adaptability to different operational scenarios.

In the current configuration, any change in alert parameters requires a direct intervention in the code, followed by a testing and deployment process, which can be both time-consuming and prone to errors. Moreover, the lack of flexibility limits the company's ability to quickly adapt to new operational conditions.

It would be advantageous to allow alert parameters to be configured dynamically, through the system's user interface. A feature that allows users to adjust alert thresholds and other related parameters could be implemented. The possibility of making these adjustments in real time, without the need to interrupt the system operation, would represent a significant advancement in system usability and adaptability.

With the implementation of dynamic parameters, users could respond more quickly to changes in operational conditions, such as variations in machine workload or updates in security policies. In addition, this flexibility would increase the system's portability, facilitating its deployment in various industrial environments with distinct requirements.

Bibliography

[1] S. Bonilla, H. Silva, M. T. da Silva, R. F. Gonçalves, and J. Sacomano, "Industry 4.0 and sustainability implications: A scenario-based analysis of the impacts and challenges", *Sustainability*, vol. 10, no. 10, p. 3740, 2018. DOI: `10.3390/SU10103740`. [Online]. Available: `https://www.mdpi.com/2071-1050/10/10/3740/pdf?version=1539767696`.

[2] W. M. Ashraf, G. M. Uddin, S. M. Arafat, S. Afghan, A. H. Kamal, M. Asim, M. H. Khan, M. W. Rafique, U. Naumann, S. G. Niazi, H. Jamil, A. Jamil, N. Hayat, A. Ahmad, C. Shao, L. B. Xiang, I. A. Chaudhary, and J. Krzywański, "Optimization of a 660 mwe supercritical power plant performance—a case of industry 4.0 in the data-driven operational management part 1. thermal efficiency", *Energies*, vol. 13, no. 21, p. 5592, 2020. DOI: `10.3390/en13215592`. [Online]. Available: `https://www.mdpi.com/1996-1073/13/21/5592/pdf?version=1603956781`.

[3] T. Mabad, O. Ali, M. A. Ally, S. Wamba, and K. C. Chan, "Making investment decisions on rfid technology: An evaluation of key adoption factors in construction firms", *IEEE Access*, DOI: `10.1109/ACCESS.2021.3063301`. [Online]. Available: `https://ieeexplore.ieee.org/ielx7/6287639/9312710/09366873.pdf`.

[4] K. Bajaj, B. Sharma, and R. Singh, "Implementation analysis of iot-based offloading frameworks on cloud/edge computing for sensor generated big data", *International Journal of Information Management Data Insights*, 2021. DOI: `10.1007/s40747-021-00434-6`. [Online]. Available: `https://link.springer.com/content/pdf/10.1007/s40747-021-00434-6.pdf`.

[5] J. Brown, “Fqc: A software for managing information from fastq files”, *Bioinformatics*, 2017. DOI: 10.1093/bioinformatics/btx373.

[6] M. D. Network. (2023). Mdn web docs: Working with json. Accessed on: 2023-09-13, [Online]. Available: https://developer.mozilla.org/en-US/docs/Learn/JavaScript/Objects/JSON.

[7] L. Ren, Z. Meng, X. Wang, L. Zhang, and L. T. Yang, “A data-driven approach of product quality prediction for complex production systems”, *IEEE Transactions on Industrial Informatics*, vol. Volume Number,

[8] G. A. Susto, A. Schirru, S. Pampuri, S. McLoone, and A. Beghi, “Machine learning for predictive maintenance: A multiple classifier approach”, *IEEE Transactions on Industrial Informatics*, vol. 11, no. 3, pp. 820–830, 2015.

[9] U. Shafi, R. Mumtaz, J. García-Nieto, S. A. Hassan, S. A. R. Zaidi, and N. Iqbal, “Precision agriculture techniques and practices: From considerations to applications”, vol. 19, no. 3, p. 3796, 2019.

[10] M. H. Rehman, “The role of big data analytics in industrial internet of things”, *Future Generation Computer Systems*, 2019. DOI: 10.1016/J.FUTURE.2019.04.020. [Online]. Available: https://dx.doi.org/10.1016/J.FUTURE.2019.04.020.

[11] D. Umakirthika, “Internet of things in vehicle safety - obstacle detection and alert system”, *International Journal of Engineering and Computer Science*, 2018. DOI: 10.18535/IJECS/V7I2.05. [Online]. Available: https://dx.doi.org/10.18535/IJECS/V7I2.05.

[12] I. H. Sarker, “Machine learning: Algorithms, real-world applications and research directions”, *SN COMPUT. SCI.*, vol. 2, no. 160, 2021, Received: January 27, 2021; Accepted: March 12, 2021; Published: March 22, 2021. DOI: https://doi.org/10.1007/s42979-021-00592-x.

[13] A. R. Asghar, S. N. Bhatti, A. Tabassum, Z. Sultan, and R. Abbas, “Role of requirements elicitation & prioritization to optimize quality in scrum agile development”, *International Journal of Advanced Computer Science and Applications*, 2016. DOI: `10.14569/IJACSA.2016.071239`. [Online]. Available: `http://thesai.org/Downloads/Volume7No12/Paper_39-Role_of_Requirements_Elicitation_Prioritization_to_Optimize_Quality.pdf`.

[14] M. Inc. (2023). Mongodb documentation. Accessed on: 2023-09-13, [Online]. Available: `https://www.mongodb.com/docs/`.

[15] P. S. Foundation. (2023). Python official documentation. Accessed on: 2023-09-13, [Online]. Available: `https://www.python.org/`.

[16] (2023). Pipenv: Python development workflow for humans. Accessed on: 2023-09-13, PyPA, [Online]. Available: `https://pipenv.pypa.io`.

[17] S. Ramírez. (2023). Fastapi documentation. Accessed on: 2023-09-13, [Online]. Available: `https://fastapi.tiangolo.com/`.

[18] Swagger. (2023). Swagger ui. Accessed on: 2023-09-13, [Online]. Available: `https://swagger.io/tools/swagger-ui/`.

[19] V. Inc. (2023). Next.js documentation. Accessed on: 2023-09-13, [Online]. Available: `https://nextjs.org/docs`.

[20] F. Inc. (2023). React documentation. Accessed on: 2023-09-13, [Online]. Available: `https://react.dev/`.

[21] M. Corporation. (2023). Typescript language documentation. Accessed on: 2023-09-13, [Online]. Available: `https://www.typescriptlang.org/`.

[22] M.-U. Team. (2023). Material-ui documentation. Accessed on: 2023-09-13, [Online]. Available: `https://mui.com/`.

[23] G. Inc. (2023). Material design 3. Accessed on: 2023-09-13, [Online]. Available: `https://m3.material.io/`.

[24] D. Inc. (2023). Docker documentation. Accessed on: 2023-09-13, [Online]. Available: `https://docs.docker.com/`.

[25] ——, (2023). Overview of docker. Accessed: yyyy-mm-dd, [Online]. Available: `https://docs.docker.com/get-started/overview/`.

[26] N. Inc. (2023). Nginx documentation. Accessed on: 2023-09-13, [Online]. Available: `https://nginx.org/en/docs/`.

[27] S. Barbosa and G. Barbosa, "Introduction to data visualization", in *17th IFIP Conference on Human-Computer Interaction (INTERACT)*, Paphos, Cyprus, 2019, pp. 527–529. DOI: `10.1007/978-3-030-29390-1_30`. [Online]. Available: `https://hal.inria.fr/hal-02877680/file/488595_1_En_30_Chapter.pdf`.