**Digital Twin Monitoring System**

Internship Report

Budurovici Cosmin-Andrei

*Date here*

Semester 6 – Internship

Venlo, Limburg, Netherlands

# Internship Report Information

Fontys University of Applied Sciences

School of Technology and Logistics

Post Office Box 141, 5900 AC, Venlo, Netherlands

|  |  |
| --- | --- |
| Type of report:  Student Name:  Student Number:  Study:  Period:  Company name:  Address:  Postal code + City:  Country:  Telephone:  Company supervisor:  Supervising Lecturer:  Company Confidential:  Number of words: | Internship Report  Cosmin Budurovici  4514270  Informatics – Software Engineering  September 2024 – January 2025  Fontys School of Technology and Logistics  Tegelseweg 255  5912 BG Venlo  Netherlands  +31 885089271  Dr. Gregor Schwake  Dr. Gregor Schwake  No  96 words |

# Abstract

*Abstract here*

# Statement of Authenticity

I, the undersigned, hereby certify that I have compiled and written the attached document / piece of work and the underlying work without assistance from anyone except the specifically assigned academic supervisors and examiners. This work is solely my own, and I am solely responsible for the content, organization, and making of this document / piece of work.

I hereby acknowledge that I have read the instructions for preparation and submission of documents / pieces of work provided by my course / my academic institution, and I understand that this document / piece of work will not be accepted for evaluation or for the award of academic credits if it is determined that it has not been prepared in compliance with those instructions and this statement of authenticity.

I further certify that I did not commit plagiarism, did neither take over nor paraphrase (digital or printed, translated or original) material (e.g. ideas, data, pieces of text, figures, diagrams, tables, recordings, videos, code, ...) produced by others without correct and complete citation and correct and complete reference of the source(s). I understand that this document / piece of work and the underlying work will not be accepted for evaluation or for the award of academic credits if it is determined that it embodies plagiarism.

|  |  |
| --- | --- |
| Name:  Student number:  Place/Date:  Signature: | Cosmin Budurovici  4514270  Venlo, *Date here*  …………………………. |

Table of Contents

[Internship Report Information II](#_Toc178249216)

[Abstract III](#_Toc178249217)

[Statement of Authenticity IV](#_Toc178249218)

[Table of Figures V](#_Toc178249219)

[List of Tables VII](#_Toc178249220)

[Glossary and List of Abbreviations VIII](#_Toc178249221)

[1 Introduction 1](#_Toc178249222)

[2 Project Management 1](#_Toc178249223)

[2.1 Problem description 1](#_Toc178249224)

[2.2 Objective 2](#_Toc178249225)

[2.3 Scope 2](#_Toc178249226)

[2.4 Risks 2](#_Toc178249227)

[2.5 Stakeholders 4](#_Toc178249228)

[2.6 Organization 4](#_Toc178249229)

[2.6.1 Approach 4](#_Toc178249230)

[2.6.2 Methodology 4](#_Toc178249231)

[2.6.2 Technology Stack 5](#_Toc178249232)

[2.6.3 Planning 5](#_Toc178249233)

[3 Analysis 6](#_Toc178249234)

[4 Design 6](#_Toc178249235)

[4.1 Selected tools 6](#_Toc178249236)

[4.2 High Level Architecture 6](#_Toc178249237)

[4.3 Environment 6](#_Toc178249238)

[5 Implementation 6](#_Toc178249239)

[6 Rollout 6](#_Toc178249240)

[7 Conclusion 6](#_Toc178249241)

# Table of Figures

[Figure 1: SPoHF structure 1](#_Toc181964694)

# List of Tables

[Table 1: Glossary and list of abbreviations VIII](#_Toc178098718)

[Table 2 Risk/Impact 3](#_Toc178098719)

[Table 3: Stakeholders 4](#_Toc178098720)

# Glossary and List of Abbreviations

|  |  |
| --- | --- |
| Term | Definition |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |

Table 1: Glossary and list of abbreviations

# 1 Introduction

This document is meant to provide a report on the development of the Digital Twin Monitoring System. This chapter will provide background information as well as an introduction to the project.

The SPoHF project focuses on the improvement of fruit and vegetable cultivation. It accomplishes this by creating a digital twin, a program that will precisely monitor plant conditions. The collected data will then be used for making predictions and prescribing courses of action.

The SPoHF project has various partners including Fontys. Fontys is a large university in the Netherlands that provides education in technology, economics, the arts and social work. Its main locations are in Eindhoven, Tilburg and Venlo.

To make it so that Digital Twin can be utilized, it is necessary for developers to be able to monitor the current state of the application and see problems when they appear. To make this possible Fontys (more precisely the informatics department) has commissioned the Digital Twin Monitoring Application.

# 2 Project Management

This chapter will provide information on the objectives of the project as well as how these objectives will be caried out.

## 2.1 Problem Description

The Digital Twin application receives data from a variety of sensors from various farms and then processes the data, ordering it and putting into a database, before making it available for analysis by AI and people.

For the digital twin project to be used effectively its users need to be able to know if the project is functioning at any given time. The users need to know when any part of the system has problems and what impact these problems have on the larger system so that they can fix them and take measures to mitigate any problems caused by the failure. While this information can be gathered by manually looking at the system this is often slow, error prone and often reveals problems too late.

## 2.2 Objective

Below there is an overview of all the major components of the program:

A diagram of a diagram

Description automatically generated

Figure 1: SPoHF structure

The objective of the Digital Twin Monitoring System is to provide a way to see the current state of Digital Twin as well as give details about performance and notify users of any problems that occur. It accomplishes this by collecting data from Digital Twin (it does not directly collect data from other subprojects such as Insect Monitoring and Climate Monitoring) and displaying it through a visual interface in real time.

## 2.3 Scope

To accomplish our objective the monitoring system will:

* Provide information about what tasks is digital twin performing at any given time.
* Provide information about the current state of the hardware at any given time.
* Show when any part of the system fails and what consequence this has.
* Display this information through a visual interface in real time.

The monitoring system will not:

* Send notifications to users when it detects a failure.
* Have functionality to solve the problems it detects (it’s solely a detection tool)

It’s important to mention that this project will only cover the creation of the application not it’s updates and maintenance.

## 2.4 Potential Risks

There are several risks that need to be considered, prevented and possibly mitigated. This chapter will discuss each risk in detail, how to prevent and mitigate them and how probable or impactful each risk is.

Bellow there is a list of the potential risks of this project as well as their probability and impact. These risks will be discussed in more detail in the rest of the chapter.

|  |  |  |  |
| --- | --- | --- | --- |
| ID | Name | Probability | Impact |
| 1 | Unfinished application | Low | High |
| 2 | Insufficient hardware | Low | Medium |

Table 2: List of risks

Here is the list in graph form:

A diagram of a graph

Description automatically generated with medium confidenceFigure 2: Risks impact/probability

The individual risks will be discussed in more detail below:

**Name:** Unfinished application

**Description:** The role of the Digital Twin Monitoring Application it to monitor Digital Twin. Unfortunately, Digital Twin is a different application that is being worked on by different developer so there is a chance that the digital twin application will not be completed or have significant delays. If this happens The Digital Twin Monitoring System will not have anything to monitor.

**Prevention:** Due to the development of Digital Twin being the responsibility of a different team there aren’t any significant measures to prevent the incompletion of Digital Twin.

**Mitigation:** There are two measures that can be taken to mitigate this risk. First is maintain communications with the development of Digital Twin to know when a problem occours as early as possible. Second is in the case that Digital Twin can’t be completed in time a barebones demo of digital twin will be created to monitor.

**Name:** Insufficient hardware

**Description:** At the current moment both the capabilities of the server the application will be running on as well as how much of those capabilities will be occupied by other applications running on the same server. As a result, there is a chance that the hardware will be unable to support the Digital Twin Monitoring System.

**Prevention:** The only measure we can take is to make sure our application is as efficient as possible and doesn’t occupy to much memory.

**Mitigation:** There are two measures that can be taken to minimize this risk once it occurs. First is to communicate with our customer and request more powerful hardware. Second would be to try to rework the application to take less resources.

## 2.5 Stakeholders

The stakeholders of this project, their interest, their influence and impact are described in the table below:

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Name | Company | Influence | Impact | Interest | What is important | How to engage |
| Budurovici Razvan | Fontys | High | High | Integrate insect counting in the Digital Twin application. | His code is what needs to be monitored in this project. As such communication with him is important. | Frequent meetings |
| Gregory Schwake | Fontys | High | High | Check progress and reports | Guides the project gives feedback and must be kept up to date. | Frequent meetings |
| Tim Riebner | Proc evolution | Medium | Medium | Customer | Will use the application after completion | Frequent meetings |
| Cristiane  Salz | Fontys | Medium | Medium | Developed the application that will pass data to insect counting. | Controls the part of the application that generates the first lair. Also needs to be monitored. | Meetings |

Table 3: List of stakeholders

## 2.6 Organization

In the following chapter the approach of this project, the methodology and the timeline will be detailed.

### 2.6.1 Approach

This project will be completed in 5 distinct phases, each with tier own specific deliverables. Below there is a list of stages and their individual deliverables:

A diagram of a project

Description automatically generated

Figure 3: Approach overview

The first phase of the project is the introduction. It consists of the developers of this project being introduced to and familiarizing ourselves with the project. This is considered done when after all the initial meetings are done, all the stakeholders have been met and the technology used for the project is understood on a basic level.

The analysis consists of assessing what the application needs to accomplish and why. The only deliverable for this phase is the product backlog. This consists of interviewing various stakeholders and creating a series of user stories. These are considered done when they are approved by the customer representative (Tim Reibner).

The design phase consists of use taking the information gathered in the analysis phase and using it to determine, in broad strokes, how the application will accomplish these goals.

This phase has tree deliverables the tool selection, the high-level overview and the developer environment.

The tool selection refers to researching and then selecting necessary tools for use in the completion of this project. It is considered done when there is a list of tools that will be used to assist the making of the project. This list should contain the justification for why the tools were chosen and be approved by both the customer representative (Tim Reibner) and the project manager (Gregory Scwake).

The high-level overview consists of making one or more diagrams that will serve as blueprints for our project. It is considered done when we have at least one diagram, the high-level architecture diagram. The project might require other diagrams for particularly complex parts of the project. All diagrams must be approved by the project manager (Gregory Scwake).

The development environment is considered done when we have: set up the project framework, installed and configured all the used tools and tested the basic framework of the project with a demo. This work must be approved by the project manager (Gregory Scwake).

The implementation represents the coding of our application. It is considered done when all the features have been completed and approved by both the project manager (Gregory Scwake) and the customer representative (Tim Reibner).

Rollout consists of configuration. Witch in our case consists of any last-minute changes that are required to make the application usable for our customer. This is considered done once the customer representative (Tim Reibner) agrees that the application fully fulfils all requirements outlined in the user stories and can be utilized. Maintenance and updates after that are outside the scope of the development team.

## 2.6.2 Methodology

This project will be managed using a combination of both waterfall and agile. The analysis, design and rollout will be done using a waterfall stile while the implementation will be done using the sprints of agile. For managing the project will be using Jira. The management program will be used for keeping track of the project board, the timeline, individual tasks, subtasks and the project backlog. Jira was chosen for two reasons. First other stakeholders are also using it for their parts of the project. Second Jira is a good all-around tool that makes it easy to keep track of timelines.

Below there is a timeline for the project:

A diagram of a product selection

Description automatically generatedFigure 4:Timeline

After the introduction, the first task of the project is the product backlog (only deliverable of the analysis phase). The product backlog consists of creating a series of user stories. To create these user stories, we will conduct 3 different interviews: one with Cristiane

Salz, another with the customer representative (Tim Reiber) and the last one an onsite interview at Compass Agro. For each interview there will first be prepared a list of questions for the interviewed to answer.

Next there is the design phases with tree deliverables the tool selection, the high-level overview and the development environment.

The tool selection consists of multiple steps. First, determine different areas that tools are needed to be used for. Second, utilize the existing use cases to create a series of criteria we use for determining which tools to select. Third find a series of tools that could be used for the monitoring application and rate them depending on how well they fulfill our criteria. Only trusted sources will be used to find tools and determine their characteristics. Lastly, select the tools that will be used.

The high-level overview consists of first making the high-level architecture diagram which will include the selected tools and their application. To better understand more complex parts of the project, such as the database, more diagrams besides the first will be made.

The development environment and implementation will be caried out using the Java programing language, the various tools that have been previously selected and the Mac operating system. To ensure the quality of the code we will use two measures: test driven development and frequent code review by Razvan Budurovici (another developer) and the project manager (Gregory Scwake).

The configuration (part of the rollout) will be caried out by first showcasing the monitoring application to the customer and then discussing with them on what changes need to be made for the application to be deployed.

# 3 Analysis

The chapter covers the analysis part of the project. More precisely creation of the product backlog through a series of user stories.

To create the user stories there has been tree interviews with various stakeholders. Before each interview a list of questions was made to ensure that all relevant points were brought up to interviewed. A full list of all prepared interview questions can be found in Appendix A: Interview Questions.

The first interview was with Cristiane Salz the developer of one of the applications that collects data for digital twin (Insect Counting). This interview offered us detailed information on how the digital twin will receive information from Insect Counting, but not significant information on how our application needs to perform.

The second interview was with the customer representative (Tim Reibner). As the customer this interview provided the bulk of the information we have used for our user stories.

The third interview was conducted significantly later due to delays and involved the developers going directly to Compass Agro. This interview provided useful insights into what the Digital Twin (the application that will be monitored) was used for.

Using the information gathered throughout the interviews we here able to make the following list of user stories:

1. As a developer I want to be able to see the trace of each individual batch of information so that I can assess the efficiency of each process.
2. As a developer or manager, I want to be able to see data from the last 4 weeks so that I can keep track of the systems functionality over time.
3. As a developer I want to be able to see the traces of currently processing baches in real time so that I can assess the current state of the system.
4. As a developer I want to see the current state of the server the main application is running on so that I can see the state of the system as a hole.
5. As a developer I want to be able to see the error message produced by the application when there is a problem so that I can have a better idea of how to solve the problem.
6. As a manager I want to see the consequences of any failure in an understandable way so that I can understand what went wrong and what consequences that failure has.
7. As a manager and developer, I want to be able to access the monitoring application with any device that can connect to the internet so that I can more easily access the data that I need.

Table 4: User stories

This list of uses was then approved by both the customer (Tim Reinber) and the company supervisor (Gregory Schake) making this phase of the project complete.

# 4 Design

This chapter will go over the development phase of this project. This phase involves tree major deliverables the selection of tools, the creation of high-level architecture and the environment set up.

## 4.1 Tool Selection

While it is possible to code the functionality of this program from scratch using already preexisting tools for the same purpose is easier and les error prone.

The first step of the process was figuring out in what areas we can use a tool instead of custom code. After analyzing our user stories and our knowledge of the program well be monitoring, we have determined tree areas where we can use tools:

* **Monitoring data storage:** To display the collected metrics of the application for 4 weeks there needs to be some way to store for the visualization to have data to pull from after the fact.
* **Hardware monitoring:** Some of the important metrics of the application are the remaining storage, ram and CPU usage. There are many tools that can be used to get the hardware metrics of multiple nodes easily.
* **Monitoring Visualization:** Using an already preexisting tool to display the information in a web interface will be significantly easier

The second step is determining what sources we will be using for researching different existing tools. Using only trusted sources is important in to make sure the selection is not affected by misinformation and as objective as possible. Below there are listed the sources well be using:

* **Stack share** is the most popular website that specifically deals with finding different tools for applications. It is used by many companies and developers and shows what companies utilize which tools which is a useful metric to determine if a tool is widely used professionally.
* **The websites of the tools themselves** will be the primary source of information regarding the features and capabilities of each tool
* **Slant** a website designed for comparing different software tools and their upsides and downsides.
* **LibHunt** a website that provides curated lists, alternatives and comparisons for different tools. Is reliable doe to its size. Is a good indicator of a tools ecosystem since it includes GitHub metrics.

The third step is using knowledge of the project and user stories to create a series of criteria that can be used to select the best tool for the project. There are two types of criteria normal and **knockout criteria**. The normal criteria are each given a score that they can have. The higher the score the better the tool in that aspect. The **knockout criteria** can either be true or false and immediately disqualifies a tool from being chosen if it’s false. Below we have all the criteria that will be used.

Below there is a list of all criteria used for this project.

* General Criteria (Applies to all areas)
  + Ecosystem
  + Free of cost
  + Compatibility
  + Wildcard
* Monitoring data storage
  + 4-week storage
  + Time series
* Hardware monitoring
  + Basic hardware monitoring
* Monitoring visualization
  + Web interface
  + Real time display
  + Display features

Of these criteria Speed is notably absent as a criteria from the data storage area. Because the database will not be handling particularly large amounts of data and the fact that measuring the exact performance of a database for this project will be very difficult it was decided to not use speed as a criteria for this project.

Below we have an example of one of our criteria in more detail:

**Name:** Ecosystem

**Explanation:** The more widely used a tool is the easier it is to find information on how to use it such as tutorial or blogs and the easier it is to solve problems since there is a higher chance that other people have had a similar problem and solved it. For this criteria we will prioritize projects that have been popular recently since this reduces the chance that the tool is outdated.

**How this criteria is measured:** This criteria’s score will be generated using two metrics the number of stars on GitHub and the activity score on LibHunt.

The number of stars on GitHub is a good indicator of how widely popular a project is. For this criteria the points will be added in the following way:

* 5 points 40.000+ stars
* 4 points 30.000-40.000 stars
* 3 points 20.000-30.000 stars
* 2 points 10.000-20.000 stars
* 1 point 5.000-10.000 stars
* 0 points 0-5.000 stars

The lib hunt activity score is based on recent metrics such as commit frequency, issue and pull request activity and community engagement (number of discussion and responses to issues). This makes the LibHunt activity score a good indicator of the how currently popular and up to date the tool is. The points for this criteria from this metric will be added in the following way:

* 5 points 9-10 score
* 4 points 7-9 score
* 3 points 5-7 score
* 2 points 3-5 score
* 1 point 1-3 score
* 0 points 0-1 score

The points of this metrics are added together to form the final score of the criteria which will be between 0-10.

The full list of criteria in more detail can be found in Appendix B: Tool Selection Criteria.

The fourth step is searching for tools and then rating them according to the criteria.

Below there are tables of the tools that have been found that pass all the **knockout criteria.** A full list of all tools that have failed to meet our knockout criteria can be found in Appendix C: Unchosen tools.

**Possible tools for data storage:**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Name | Good ecosystem | Works with other tools | Wildcard | Total |
| Prometheus | 10 | 6 | 2 | 18 |
| Timescale DB | 7 | 3 | 1 | 11 |
| Quest DB | 7 | 0 | 0 | 7 |
| Influx DB | 8 | 3 | 0 | 11 |

**Possible tools for hardware monitoring:**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Name | Good ecosystem | Works with other tools | Wildcard | Total |
| Prometheus Node Exporter | 6 | 3 | 0 | 9 |
| Telegraph | 7 | 3 | 0 | 10 |
| Zabbix | 6 | 3 | 0 | 9 |

**Possible tools for visualization:**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Name | Good ecosystem | Compatibility | Display features | Wildcard | Total |
| Grafana | 10 | 3 | 10 | 2 | 25 |
| Apache Superset | 10 | 3 | 7 | 0 | 21 |
| Metabase | 9 | 3 | 6 | 0 | 18 |

A full explanation of each score a tool received can be found in Appendix D: Score Explanations.

The fifth step is choosing which tools will be used.

For each category the highest score that is held by Prometheus, Telegraph and Grafana. Prometheus and Grafana work well with each other and are easy to use, but Telgraf does not work with Prometheus. Because of this the actual option with the highest score is the one that works with Prometheus, the Prometheus node exporter.

The final selection is Prometheus node exporter, Prometheus and Grafana.

## 4.2 High-Level Overview

The high-level overview was made by studding our use cases and our used tools to make a design for the architecture of the monitoring application. The diagram can be seen below:

A diagram of a computer system

Description automatically generated

Figure 5: Architectural diagram 1.0

A more in depth explanation of the above diagram is the following. Data is first collected by an application such as insect counting (other data collecting applications are Lightning Monitoring, Temperature Monitoring Etc). The data then goes trough Digital Twin. As the data passes through Digital Twin the Monitoring Data Collectors collect data about the process as well as data about the condition of the hardware. The monitoring data is then stored and later visualized.

## 4.3 Development environment

To complete the development environment the environment described in the high-level overview was set up so that the user can see some basic metrics regarding both Apache spark and the hardware. A full description on how to set up the environment was set up in Appendix E: Configuration Environment

# 5 Implementation

This section will cover in detail all the sprints of the implementation their results. As well as how the sprints where organized.

For keeping track of sprints and tasks Jira is used. First a list of all necessary sprints is planed based on the user stories. Then each sprint is split into a series of smaller tasks and executed.

In order to properly organize the sprints, we have meet with both the customer representative and project manager to determine the importance of each individual user story and used that to organize a series of sprints. The following sprints have been organized in order of execution:

|  |  |  |
| --- | --- | --- |
| ID | Name | Description |
| 1 | Hardware Monitoring | The objective of this sprint is the creation of a visual interface that can give the customer data about the current state of the servers the application is running on. |
| 2 | Error Monitoring | The objective of this sprint is the creation of a user interface that can be used to alert the user of any errors and or problems the system encounters when executing jobs. |
| 3 | Trace Monitoring | The objective of this sprint is creating an interface that shows the user traces of all data that is entered in Digital twin. |
| 4 | Management interface | The objective of this sprint is taking data presented in the other interfaces and using it to create an interface that can utilized by users that aren’t developers. |

Table 5: Sprint List

## 5.1 Sprint 1: Hardware monitoring

The objective of this sprint is the creation of a visual interface that can give the customer data about the current state of the servers the application is running on. More precisely we have to give our customers information about the CPU, RAM and memory current usage of each device as well as these metrics over a period selected by the user.

To achieve the objective for this sprint the work was split the work into the following tasks:

1. Desing an interface for the hardware monitoring data.
2. Document the process in this document.

For this specific sprint setting up the database and the monitoring of the hardware was unnecessary as these steps where completed during the development environment.

To design the interface the following mockup was first created:

A screenshot of a computer

Description automatically generated

Table 6: Hardware monitoring mock up

The tree or more traffic lights at the top represent different servers. They are red if the server is unavailable or has CPU, RAM or Memory usage over 90%. They are yellow if the server is has over 80% CPU, RAM or Memory usage and green otherwise. Below we have a dropdown that selects witch server the user wants to see in more detail. Below that we have the current RAM, CPU and Memory usage of the server as well as graphs showing these statistics over a period of time selected by the user in another dropdown.

Below we have the final version of our hardware monitoring interface:

A screenshot of a computer

Description automatically generated

Table 7: Hardware monitoring interface

Grafana features have been used to replace the traffic lights with server images and the loading bars with speedometers to make the image clearer. This version of the interface was then approved by the customer representative and the project manager.

## 5.2 Sprint 2: Error monitoring

The objective of this sprint is the creation of a user interface that can be used to alert the user of any errors and or problems the system encounters when executing jobs. For now this is done by only monitoring the errors and warnings of Apace Spark ass it is the only application that is ready to be monitored.

When in the initial planning for this sprint it was discovered that Prometheus the database, we have been using for the collection of hardware monitoring data is extremely inadequate for storing error and warning information from logs. As a result it was decided to utilize Loki a databases designed by the Grafana team specifically for log retention. This specific tool was chosen for the following reasons:

* It’s relation with Grafana makes it very easy to use with it guaranteeing that we can display the information quickly and effectively.
* It’s molded after Prometheus making it very easy to use and not requiring the developers to learn much new information to learn how to use it.
* It comes with Promtail an application designed to extract log data from a file or folder a submit it to Loki. This makes it very easy to extract data from Apache Spark.

Because of the use of this new tool the Architectural diagram was updated:

A diagram of data processing

Description automatically generated

Figure 6: Archsitectural diagram updated

To complete this sprint we have divided the work into multiple smaller steps:

1. Configure Apache Spark to write the logs to a rolling file so it can be read by Promtail.
2. Set up Promtail to collect data from Apache Spark.
3. Set up Prometheus to collect data.
4. Connect Grafana with Prometheus.
5. Design a dashboard showcasing both the collected errors and warnings.

Before making the visual interface the following mock up:A screenshot of a computer

Description automatically generated

Figure 7: Error monitoring mockup

The interface shows a button used to select the time to in which the errors and warning where produced. Besides that, there are two fields one showing the warnings Produced by Apache spark and the other showing the errors both with timestamps for each message.

Bellow there is a screen show with the final result:

A screenshot of a computer

Description automatically generated

Figure 8: Error monitoring interface

This version of the interface was then approved by the customer representative and the project manager.

## Sprint 3: Trace monitoring

The objective of this sprint is creating a user interface capable of showing the user distributed traces.

The first step of this sprint is to determine if monitoring distributed traces is possible for Apache Spark without modifying application. To answer the question the following where researched: Apache spark and its monitoring features and other Apache Spark monitoring systems and how they monitor distributed traces. The result of this research is the fact that without instrumenting the application (which will modify it substantially) it is not possible to monitor the distributed traces of the application.

The closest thing that we can do is show the duration of each spark job trough monitoring logs. We did this by modifying our error monitoring interface to also monitor the completion of jobs and their duration. Below we have an image of the updated UI:

A screenshot of a computer

Description automatically generated

Figure 9:Updated Error Monitoring with Trace Monitoring

This implementation was significantly easier than the expected as a result to take advantage of the remaining time of the sprint deletion of records older than 4 was also implemented. This required the restarting and modifying of configuration of both Prometheus and Loki.

This work was approved by both the customer and the project manager.

# 6. Rollout

# Appendix

## Appendix A: Interview Questions

### First interview (Cristiane Salz)

* How does Insect Counting transfer data to Digital Twin?
* What data does Insect Counting transfer to Digital Twin?
* How often does Insect Counting transfer data to Digital Twin?

### Second interview (Tim Riebner)

* What kinds of information does the monitoring system need to collect about the operation of digital twin.
* Does the monitoring system also need to monitor the state of the hardware Digital Twin is running on.
* Should the monitoring system update the information shown in real time or only periodically.
* What types of users access the monitoring system and what can each of them see.
* How will the monitoring system display information to users?
* How long should the monitoring information be saved for?

### Third interview (Compass Agro)

* What does Compass Agro do?
* How will digital twin be used to help Compass Agro?
* Will Compass Agro directly use the monitoring system?

## Appendix B: Tool Selection Criteria

### General Criteria (Applies to all areas)

**Name:** Ecosystem

**Explanation:** The more widely used a tool is the easier it is to find information on how to use it such as tutorial or blogs and the easier it is to solve problems since there is a higher chance that other people have had a similar problem and solved it. For this criteria we will prioritize projects that have been popular recently since this reduces the chance that the tool is outdated.

**How this criteria is measured:** This criteria’s score will be generated using two metrics the number of stars on GitHub and the activity score on LibHunt.

The number of stars on GitHub is a good indicator of how widely popular a project is. For this criteria the points will be added in the following way:

* 5 points 40.000+ stars
* 4 points 30.000-40.000 stars
* 3 points 20.000-30.000 stars
* 2 points 10.000-20.000 stars
* 1 point 5.000-10.000 stars
* 0 points 0-5.000 stars

The lib hunt activity score is based on recent metrics such as commit frequency, issue and pull request activity and community engagement (number of discussion and responses to issues). This makes the LibHunt activity score a good indicator of the how currently popular and up to date the tool is. The points for this criteria from this metric will be added in the following way:

* 5 points 9-10 score
* 4 points 7-9 score
* 3 points 5-7 score
* 2 points 3-5 score
* 1 point 1-3 score
* 0 points 0-1 score

The points of this metrics are added together to form the final score of the criteria which will be between 0-10.

**Name:** Free of cost

**Explanation:** The company does not want to pay for any additional tools that will be using in the creation of the monitoring application as a result we will only be using free and open-source options. This is a **knockout criteria**.

**How this criteria is measured:** True or false. False if it cost money to use.

**Name:** Compatibility

**Explanation:** This criteria is meant to represent if the tool selected works well with other tools that will be used. We define “works well” as having features or plugins that make it easier to use alongside other tools in the creation of the application.

**How this criteria is measured:** 3 points will be assigned for each other tool the selected tool “works well” with. These points will only be added if the tool it works well with another tool that is utilized.

**Name:** Wildcard

**Explanation:** This is meant to represent any unique features or problems a tool has that are not covered by other criteria.

**How this criteria is measured:** A score from -3 to +3 point depending on the specific tool.

### Monitoring data storage

**Name:** 4-week storage

**Explanation:** The data storage must be capable of storing only the last 4 weeks of data and automatically deleting older data as to not occupy too much space. Is a **knockout criteria**.

**How this criteria is measured:** True or false. False if it can’t automatically delete data older than 4 weeks.

**Name:** Time series

**Explanation:** For this project we require any database used to be a time series database. This is because time series databases make it easier to keep track of the timing of different metrics making the excellent for a monitoring solution. This is a **knockout criteria**.

**How this criteria is measured:** True or false. True if the tool is a time series database.

**Name:**  Speed

**Explanation:** Because the database will not be handling particularly large amounts of data and the fact that measuring the exact performance for this project will be very difficult it was decided to not use speed as a criteria for this project.

**How this criteria is measured:** Not applicable

### Hardware monitoring

**Name:**  Basic Hardware Monitoring

**Explanation:** The tool used must be able to monitor the remaining ram, memory and the cpu usage. This is a **knockout criteria**.

**How this criteria is measured:** True or false. True if it can monitor all tree metrics.

### Monitoring visualization

**Name:**  Web Display

**Explanation:** For ease of use the tool must be able to display the information through a web interface to make it easy to access for the customers. This is a knockout criteria.

**How this criteria is measured:** True or false. True if it can display the information rough a web interface.

**Name:**  Real Time Display

**Explanation:** In order to quickly address problems the visual interface must be able to change in real time to reflect changes in the monitored application. This is a **knockout criteria**.

**How this criteria is measured:** True or false. True if it can modify itself in real time.

**Name:**  Display Features

**Explanation:** The tool should have the features necessary to display the data in concise and easy to understand way.

**How this criteria is measured:** This criteria is measured based on the available feature of the tools. Each helpful feature will have a number of points assigned to it and the final score will be made of the sum of the features the tool has:

* Can display pop ups - 1 point
* Can display text messages – 3 points
* Can switch between multiple pages – 1 point
* Can display traces – 3 points
* Can display graphs – 2 points

## Appendix C: Unchosen Tools

….

## Appendix D: Score Explanations

### Monitoring Data Storage

**Tool Name:** Prometheus

**Good ecosystem:** 5 points for GitHub stars and 5 points for activity score

**Works with other tools:** 3 points for working with Grafana and 3 additional for working with Prometheus node explorer for a total of 6 points if both tools are picked.

**Wildcard:** 2 points thanks to the fact that Prometheus is considered easier to use than its alternatives

**Tool Name:** Timescale DB

**Good ecosystem:** 2 points for GitHub stars and 5 points for activity score

**Works with other tools:** 3 points for working with Telegraph and 3 points for working with Zabbix because Telegraph and Zabbix cannot be both selected Timescale DB receives the only tree points.

**Wildcard:** 1 point for the fact that Timescale DB uses the Postgres SQL as it’s query language that the developer of this project is already familiar with.

**Tool Name:** Quest DB

**Good ecosystem:** 2 points for GitHub stars and 5 points for activity score

**Works with other tools:** does not work particularly well with other tools

**Wildcard:** Nothing notable

**Tool Name:** Influx DB

**Good ecosystem:** 3 points for GitHub stars and 5 points for activity score

**Works with other tools:** 3 points for telegraph

**Wildcard:** Nothing notable

### Hardware Monitoring

**Name:** Prometheus Node Exporter

**Good ecosystem:** 2 points for GitHub stars and 4 points for activity score

**Works with other tools:** 3 points for working with Prometheus

**Wildcard:** Nothing notable

**Name:** Telegraph

**Good ecosystem:** 2 points for GitHub stars and 5 points for activity score

**Works with other tools:** 3 points for working with Timescale DB

**Wildcard:** Nothing notable

**Name:** Zabbix

**Good ecosystem:** 1 point for GitHub stars and 5 points for activity score

**Works with other tools:** 3 points for working with Timescale DB

**Wildcard:** Nothing notable

### Monitoring Visualization

**Name:** Grafana

**Good ecosystem:** 5 points for GitHub stars and 5 points for activity score

**Works with other tools:** 3 points for working with Prometheus

**Display features:** 10 points for having all display features

**Wildcard:** 2 points thanks to the fact that Grafana is considered easier to use than its alternatives

**Name:** Apache Superset

**Good ecosystem:**5 points for GitHub stars and 5 points for activity score

**Works with other tools:** 3 points for working with Timescale DB

**Display features:** 7 points for all display features except trace display

**Wildcard:** Nothing notable

**Name:** Metabase

**Good ecosystem:** 4 points for GitHub stars and 5 points for activity score

**Works with other tools:** 3 points for working with Timescale DB

**Display features:** 6 points for all display features except trace display and pop-ups

**Wildcard:** Nothing notable

## Appendix E: Configuration Environment