HTL - Perg

Höhere Abteilung für Informatik

Diplomarbeit

LogSense: Infrastruktur-Monitoring-Anwendung

Projektteam: Philipp Borbely
Sarah Ettlinger
Thomas Jilek
Emily Stadlbauer

Betreuer: Prof. Maria Inreiter, MSc

In Zusammenarbeit mit: Dynatrace Austria GmbH
Daniel Kreuzer, MSc

Eidesstattliche Erklärung

Hiermit versichern wir, dass die Diplomarbeit in eigenständiger Weise, ohne Hilfe von Außenstehenden und ohne Benutzung von nicht angegebenen Quellen angefertigt wurde. Sollten Teile der vorliegenden Arbeit wörtliche oder sinngemäß den Inhalt aus externen Quellen direkt oder indirekt übernommen worden sein, sind diese Stellen gekennzeichnet und die jeweiligen Quellen angegeben.

Perg, 3. April 2024 Unterschrift 

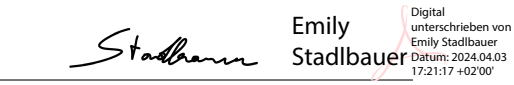
Philipp Borbely

Perg, 3. April 2024 Unterschrift 

Sarah Ettlinger

Perg, 3. April 2024 Unterschrift 

Thomas Jilek

Perg, 3. April 2024 Unterschrift 

Emily Stadlbauer

Gender-Erklärung

In dieser Arbeit werden personenbezogene Bezeichnungen, die sowohl Männer als auch Frauen umfassen, aus Gründen der Lesbarkeit und Kürze generell in der im Deutschen üblichen maskulinen Form verwendet. Dies soll jedoch keine Diskriminierung aufgrund des Geschlechts sein oder den Grundsatz der Gleichheit verletzen.

Perg, 3. April 2024 Unterschrift Philipp Borbely

Philipp Borbely

Perg, 3. April 2024 Unterschrift Ettlinger

Sarah Ettlinger

Perg, 3. April 2024 Unterschrift

Thomas Jilek

Perg, 3. April 2024 Unterschrift Stadlbauer

Emily Stadlbauer

Danksagung

Wir möchten unserer Betreuerin Prof. Maria Inreiter, MSc, für ihre Tipps, Korrekturlesungen und ihre organisatorische Unterstützung während des Verfassens dieser Arbeit danken. Weiters bedanken wir uns herzlich bei unseren Auftraggebern Daniel Kreuzer und Jürgen Riegler für ihre technische Unterstützung und Betreuung im Rahmen unseres Praktikums.

Kurzfassung

Die Grundidee von LogSense ist es, die Bildschirmzeitverwaltung von Smartphones und Teile der Funktionalität des Task-Managers, vom Betriebssystem Windows her bekannt, zu fusionieren. Als erweiterte Funktionalität werden Probleme wie ein hoher Ressourcenverbrauch durch einen bestimmten Prozess erkannt und mitgeteilt. Ein wesentlicher Punkt ist auch die langfristige Speicherung der Daten, um sie rückblickend zu betrachten und zu analysieren. Um dieses Ziel zu erreichen, werden 4 Hauptkomponenten benötigt:

Der **Agent** läuft als Hintergrundprozess auf dem Client Rechner, erfasst Daten über die Hardware-Komponenten des Rechners und sendet sie zur Analyse an den Server.

In der **Datenbank** werden die erfassten Informationen für die spätere Auswertung persistiert. Da es sich dabei um eine große Menge an Zeitreihen-Daten handelt, muss eine leistungsstarke Datenbank eingesetzt werden. Dafür eignet sich am besten die Datenbank TimescaleDB, da sie genau für diesen Anwendungszweck entwickelt worden ist.

Die gemessenen Daten werden serverseitig durch **Machine Learning Algorithmen** und **statistische Verfahren** analysiert und aufbereitet. Dazu zählt die Identifizierung von Anomalien, Ereignissen im Trendverlauf und die Vorhersage von freiem Speicherplatz.

Die Benutzeroberfläche zeigt die erfassten Daten und die Ergebnisse der Analyse in Form von **Diagrammen und Statistiken** übersichtlich an. Zusätzlich kann der Benutzer über die Benutzeroberfläche mit dem System interagieren und eigens definierte Warnungen für ein Gerät erstellen.

Das Ergebnis ist ein System, bestehend aus den oben angeführten Komponenten, mit dem der Ressourcenverbrauch von Rechnern, sowie die Laufzeit und der Ressourcenverbrauch von einzelnen Prozessen überwacht werden kann. Die gesammelten Daten werden analysiert und ausgewertet, um in Form von Statistiken, Ereignissen und Anomalien dargestellt zu werden.

Abstract

The basic idea of LogSense is to merge the screen time management of smartphones and some functionalities from the Windows operating system. As an additional feature, unexpected behaviour such as high resource consumption of particular processes can be detected and reported. Another key feature is the long-term storage of the gathered data so that it can be viewed and analysed retrospectively. To achieve this goal, 4 main components are required:

The **agent** runs as a background process on the client computer, collects data about the hardware components of the PC and sends it to the server for further analysis.

The recorded information is then persisted in the **database** for further analysis. As this involves large amounts of time series data, a powerful database is required. The database most suitable for this is TimescaleDB, since it was developed precisely for this purpose.

The measured data is analysed on the server through the use of **machine learning algorithms** and **statistical methods**. This includes the identification of anomalies, trending events and the prediction of free storage space.

The graphical user interface displays the recorded data and analysis results in form of **diagrams and statistics**. In addition, the user can interact with the system via the user interface and define customized warnings for a device.

The result is a system consisting of the components listed above, which can be used to monitor the resource usage of computers as well as the runtime and resource consumption of individual processes. The collected data is analysed and evaluated in order to be displayed in the form of statistics, events and anomalies.

Inhaltsverzeichnis

1 Einleitung

1.1 Problemstellung

Beim Arbeiten mit einem Rechner treten oftmals vielerlei Probleme in Bezug auf Applikationen und Hardware auf. Allerdings werden Probleme nur erkannt, wenn Benutzer direkt mit dem betroffenen Rechner interagieren, daher sind Systemadministratoren nicht in der Lage, an einer zentralen Stelle Probleme alleine zu identifizieren. Außerdem zeigen bereitgestellten Tools wie der Windows-Taskmanager keine rückblickenden Daten über den Hardware-Verbrauch oder die Verwendungsdauer von Applikationen an.

1.2 Zielsetzung

Das Ziel unserer Diplomarbeit ist es, dem Benutzer eines Rechners und in größeren Rechnernetzen auch dessen Administrator, eine Übersicht über den Ressourcenverbrauch eines PCs und dessen Applikationen zur Verfügung zu stellen. Die Applikation ermöglicht, vergangene Hardware-Metriken über Applikationen einzusehen, um potenzielle Trends oder ungewöhnliches Verhalten zu erkennen. Auch werden Analysemöglichkeiten zum Identifizieren von Anomalien und Ereignissen im Trendverlauf angeboten werden. Im Allgemeinen wird dadurch das Auftreten von Problemen frühzeitig erkannt werden und dessen Behandlung durch das Bereitstellen von relevanten Informationen vereinfacht werden.

1.3 Projektinhalt

Eine übersichtliche Projektstruktur ist eine Notwendigkeit, um die Projektziele zu erreichen. Die Diplomarbeit besteht aus den folgenden 4 Hauptkomponenten, die mit einer Schnittstelle untereinander kommunizieren:

1.3.1 Agent

Der Agent erfasst alle 10 Sekunden Daten über die Hardware-Komponenten eines Rechners, auf dem das Betriebssystem Windows läuft. Nach der Datenerfassung fasst der Agent die Daten der letzten 60 Sekunden zusammen und bereitet sie auf, bevor er sie

zur weiteren Analyse an den Server sendet. Bei der ersten Datenerfassung werden zusätzlich allgemeine Daten über den Rechner erfasst. Die erste Nachricht signalisiert das Ende der zuletzt gestarteten Session und gleichzeitig den Start einer neuen Session.

1.3.2 Analyse

Die gemessenen Daten werden serverseitig durch Machine Learning Algorithmen und statistische Verfahren analysiert. Dazu zählt die Identifizierung von Anomalien, also von Datenpunkten, die enorm von den restlichen Daten abweichen. Außerdem werden sogenannte ”Change-Pointsërkannt, also Datenpunkte, welche Ereignisse im Trendverlauf identifizieren. Zusätzlich gibt es Möglichkeiten Vorhersagen zu erhalten, konkret zukünftigen freien Speicherplatz vorherzusagen.

1.3.3 Schnittstellen

Eine RESTful API wird für die Kommunikation zwischen den einzelnen Komponenten verwendet. Die Schnittstelle nimmt Anfragen sowohl vom Agent als auch von der Benutzeroberfläche entgegen und bearbeitet diese entsprechend. Sendet der Agent neue Daten, werden diese zum Persistieren an die Datenbank übergeben. Wohingegen bei Anfragen von der Benutzeroberfläche die Daten aus der Datenbank geholt und bei Bedarf analysiert und ausgewertet. Sobald dieser Prozess abgeschlossen ist, werden die Grunddaten gemeinsam mit den Ergebnissen der Auswertung an die Webanwendung weitergeleitet.

1.3.4 Datenspeicherung

In der Datenbank werden die erfassten Informationen für die spätere Auswertung persistiert. Da es sich dabei um eine große Menge an Zeitreihen-Daten handelt, muss eine leistungsstarke Datenbank eingesetzt werden. Dafür eignet sich am besten die Datenbank TimescaleDB, da sie genau für diesen Anwendungszweck entwickelt worden ist.

1.3.5 Benutzeroberfläche

Die Benutzeroberfläche zeigt die analysierten Daten in Form von Diagrammen und Statistiken übersichtlich an. Beim erstmaligen Aufrufen der Benutzeroberfläche kann ein Gerät ausgewählt werden, dessen Daten dargestellt werden. Weiters ist der Benutzer in der La-

ge, eigens definierte Warnungen für ein Gerät erstellen, indem er die Art der Diagnose und den jeweiligen Grenzwert angibt.

1.4 Projektumfeld

Diese Diplomarbeit ist in Zusammenarbeit mit dem Unternehmen Dynatrace Austria GmbH entstanden. Weiters sind mehrere Personen der HTL Perg beteiligt, die im nachfolgendem Abschnitt angeführt werden.

1.4.1 Projektteam

Das Projektteam besteht aus 4 Schüler:innen der HTL Perg und die Verantwortlichkeiten haben sich dabei wie folgt aufgeteilt:

- Philipp Borbely - Datenanalyse und -auswertung
- Sarah Ettlinger - Datenerfassung
- Thomas Jilek - Datenspeicherung und Systemintegration
- Emily Stadlbauer - Visualisierung und Anzeige

1.4.2 Auftraggeber

Dynatrace [1] ist ein internationales Software-Unternehmen, das im Bereich Cloud-Computing tätig ist, mit dem Hauptsitz in Waltham, Massachusetts. Das Unternehmen ist im Jahr 2005 unter dem Namen 'dynaTrace Software GmbH' gegründet worden.[49]

Dynatrace bietet als einziges Produkt die Dynatrace Software Intelligence Plattform an, die weltweit von verschiedenen Unternehmen genutzt wird, um ihre Anwendungen und ihre gesamte technische Infrastruktur zu überwachen und zu optimieren. Die Dynatrace Plattform bietet Lösungen für kritische digitale Herausforderungen, die die Sicherheit des Systems und das Nutzererlebnis des Endbenutzers betreffen. In folgender Grafik wird veranschaulicht, für welche Bereiche das Produkt eine Lösung zur Verfügung stellt und mit welchen Technologien diese umgesetzt worden sind.
Abbildung 1.1 zeigt die umfangreiche Architektur der Dynatrace Plattform. Die vorliegende Diplomarbeit ist den Bereichen 'Infrastructure Observability' und 'Application Observability' zuzuordnen. [48]

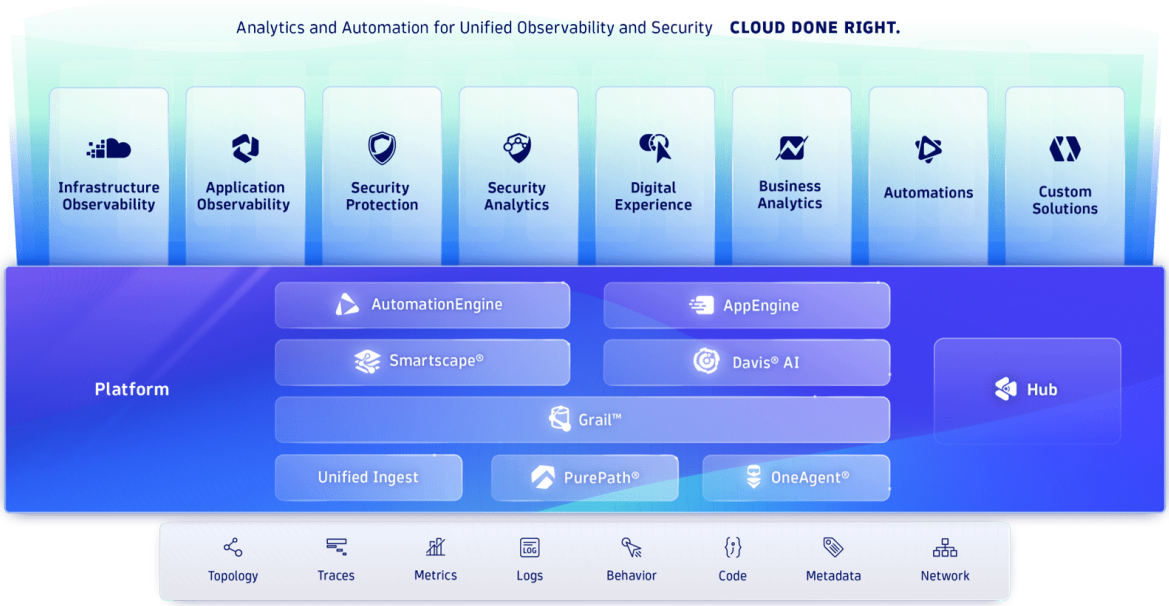


Abbildung 1.1: Architektur der Dynatrace Plattform [48]

[1] 'https://www.dynatrace.com/de/'

2 Theoretische Grundlagen

2.1 Datenerfassung

In den nachfolgenden Abschnitten werden die im Projekt verwendeten Konzepte und Technologien für die Datenerfassung beschrieben.

2.1.1 Agent

Für die Erfassung der Ressourcendaten wird ein sogenannter Agent verwendet. Dieser kann selbstständig, das heißt ohne das Zutun eines Benutzers oder eines anderen Programmes, Tätigkeiten und Prozesse ausführen. Das heißt, der Agent wartet so lange, bis eine gewisse Zeit abgelaufen, ein Programm gestartet worden oder ein Ereignis jeglicher Art festgestellt worden ist und erledigt darauf bestimmte Aufgaben. [39]
Dabei gibt es im Wesentlichen 3 Arten von Agenten:

- **Reaktive Agenten**

  Reaktive Agenten beobachten die Umgebung, auf der sie laufen und treffen Entscheidungen basierend auf den erfassten Daten.

- **Adaptive Agenten**

  Diese Art von Agenten handelt, im Vergleich zu reaktiven Agenten, zusätzlich basierend auf bereits zuvor erfassten Daten und erkannten Zusammenhängen zwischen den Informationspunkten.

- **Kognitive Agenten**

  Kognitive Agenten lernen Muster aus den erfassten Daten und wägen dann selbstständig ab, welche Aufgaben wann und wie durchgeführt werden sollen, um bestmöglich auf die aktuelle Situation der Umgebung zu reagieren.

Für diesen Anwendungsfall wird ein reaktiver Agent verwendet. Nach Ablauf von 60 Sekunden werden die Daten gemessen, je nach Art der Daten zusammengefasst und anschließend in einem geeigneten Format zur Auswertung an den Server gesendet. Es wird

also überprüft, um welche Daten es sich handelt und daraufhin entschieden, wie die Daten für das Senden vorbereitet werden.

2.1.2 Java

Die Programmiersprache Java ist objektorientiert, besitzt eine einfache Syntax mit überschaubarem Sprachumfang und ist durch kontrollierte Speicherzugriffe sicher. Wenn es eine passende JVM (= Java Virtual Machine) für das Betriebssystem gibt, kann das Programm auf diesem Betriebssystem laufen. Diese Plattformunabhängigkeit wird dadurch ermöglicht, dass der Java-Quellcode in Bytecode übersetzt wird, der dann von der JVM für das jeweilige Betriebssystem interpretiert werden kann. Das Logo dieser Programmiersprache ist in der Abbildung 2.1 ersichtlich.

Abbildung 2.1: Java Logo [14]

Der Agent, der lokal auf den Windows Rechnern läuft und die Daten über den Ressourcenverbrauch des Rechners erfasst, ist in der Programmiersprache Java entwickelt. Der Grund dafür ist, dass es für Java viele verschiedene Bibliotheken gibt, die auf die Hardwaretreiber zugreifen, um die benötigten Daten über den Rechner, die darauf laufenden Programme und deren Ressourcenverbrauch auszulesen.

2.1.3 Oshi

Oshi[1] (= Operating System and Hardware Information) ist eine für die Programmiersprache Java entwickelte Bibliothek, die Daten über die Hardware und das Betriebssystem eines Rechners erfassen und bereitstellen kann. Dafür wird im Hintergrund die JNA-Bibliothek (= Java Native Access) verwendet. Diese dient als eine plattformunabhängige Schicht zwischen dem Programmcode der Oshi-Bibliothek und dem nativen Code, der abhängig vom Betriebssystem die Systeminformationen ausliest. [53] Aus diesem Grund kann Oshi Daten über unterschiedliche Betriebssysteme, unter anderem Windows, Linux, macOs und verschiedene Unix Distributionen, erfassen. [59]

[1] `https://www.oshi.ooo/`

2.1.4 java.util.concurrent Package

Dieses Package ist mit der Java Version 5 eingeführt worden und ermöglicht eine gleichzeitige Ausführung von mehreren Prozessen in einer Java-Anwendung. Das ist vor allem bei Webanwendungen von Vorteil, da neuere Interaktionen mit der Benutzeroberfläche weiterhin verarbeitet werden, obwohl ältere Anfragen noch nicht fertig abgearbeitet worden sind. Zusätzlich besteht die Möglichkeit komplexe und ressourcenintensive Aufgaben wie Datei- und Datenbankzugriffe oder Netzwerkanfragen auszuführen, während gleichzeitig andere Aufgaben erledigt werden. Dadurch wird der Programmablauf nicht komplett blockiert, wenn ein Datenbankzugriff durchgeführt wird, der mehrere Sekunden dauert. [38]

Wie aus dem Namen dieses Packages schon hervorgeht, werden Programmteile gleichzeitig und nicht parallel ausgeführt. Eine parallele Ausführung würde bedeuten, dass verschiedene Prozesse zum selben Zeitpunkt von verschiedenen Kernen der CPU bearbeitet werden. Das ist nur möglich, wenn diese Prozesse unabhängig voneinander sind und nicht miteinander kommunizieren. Werden Aufgaben hingegen gleichzeitig ausgeführt, arbeitet die CPU an einem Prozess und wendet sich nach einer gewissen Zeit einem anderen Prozess zu. Das geschieht so schnell, dass es von außen betrachtet so aussieht, als würden die Prozesse parallel laufen. [114]

Genau dieses Prinzip der Gleichzeitigkeit wird als Basis für folgende 5 Hauptkomponenten des java.util.concurrent Packages verwendet:

- **Executors**

  Die Executer Schnittstelle definiert eine einzige Methode mit dem Namen execute, der als Parameter eine Implementierung der Runnable Schnittstelle übergeben wird. Dadurch kann die Übermittlung einer Aufgabe und deren Ausführung zeitlich voneinander getrennt werden. Je nach Implementierung der Executor Schnittstelle kann die übergebene Aufgabe sofort im aktuellen oder in einem anderen Thread ausgeführt werden. Wobei es natürlich auch möglich ist, bei mehreren übergebenen Aufgaben die Reihenfolge der Ausführung nach bestimmten Kriterien zu planen. [13]

- **Queues**

  Wenn mehrere Threads auf die gleiche Queue zugreifen und Elemente hinzufügen oder herausnehmen, kann es zu Synchronisations-Problemen im Programmablauf kommen. Um dies zu verhindern, stellt dieses Package verschiedene Implementierungen von blockierenden Queues zur Verfügung, die die Zugriffe auf die Elemente der Queue steuern. [88]

- **Timing**

  Die TimeUnit Enumeration definiert 7 zeitliche Größen, mit denen eine Zeitdauer angegeben werden kann. Dazu gehören: Tage, Stunden, Minuten, Sekunden, Millisekunden, Mikrosekunden und Nanosekunden. Zusätzlich werden Funktionen zum Konvertieren in eine andere Zeitgröße und zum Festlegen der Wartezeit eines Threads unterstützt. [13]

- **Synchronizers**

  Das java.util.concurrent Package stellt 4 verschiedene Implementierungen von Synchronisationsobjekten zur Verfügung. Diese werden verwendet, um den Zugriff von mehreren Threads auf bestimmte Ressourcen zu steuern. Das wird umgesetzt, indem das Synchronisationsobjekt den Zustand der Ressource enthält. Abhängig von dem Wert dieses Zustands wird Threads der Zugriff gewährt oder verweigert. [34]

- **Concurrent Collections**

  ConcurrentHashMap, ConcurrentSkipListMap, ConcurrentSkipListSet, CopyOnWriteArrayList und CopyOnWriteArraySet sind Implementierungen von Collections in diesem Package. Sie unterstützen alle eine Steuerung des Zugriffs bei gleichzeitiger Nutzung von mehreren Threads. Synchronisierte Collections besitzen die gleiche Funktionalität, wobei Concurrent Collections bevorzugt verwendet werden sollen. Sie verwalten den Zugriff effizienter, indem unendlich viele Threads auf die Collection lesend zugreifen dürfen und nur bei schreibenden Zugriffen anderen Threads der Zugriff verweigert wird. [13]

2.2 Datenverarbeitung

In den nachfolgenden Abschnitten werden die im Projekt verwendeten Konzepte und Technologien für die Datenverarbeitung beschrieben.

2.2.1 Python

Für das gesamte Backend ist die Programmiersprache Python zur Verwendung gekommen. Python ist eine interpretierte und höhere Programmiersprache, welche mehrere Programmierparadigmen aufweist, unter anderem die objektorientierte und funktionale Programmierung. Besonders bekannt ist Python für seine kompakte, lesbare und einfache Syntax. Beispielsweise werden bei Python Einrückungen anstatt von Klammern verwendet. Außerdem erfordert Python keine explizite Deklaration von Variablentypen, welche zusätzliche Flexibilität mit sich bringt. In dieser Diplomarbeit ist Python hauptsächlich aufgrund der großen und breiten Anzahl an Data-Science und Maschine-Learning-Bibliotheken zur Verwendung gekommen. [82]

2.2.2 Pandas

Pandas ist eine Python-Bibliothek zur Datenverarbeitung und -analyse. Sie stellt verschiedene Datenstrukturen und Funktionen für das Manipulieren von tabellarischen und strukturierten Daten zur Verfügung. Zu den wichtigsten Strukturen gehören Dataframes und Series.

- **Series**
  Series sind eindimensionale, Array-ähnliche Objekte, die wie eine einzelne Spalte in einer Tabelle angeordnet sind.

- **Dataframes**

  Bei Dataframes handelt es sich um zweidimensionale Tabellen mit Zeilen und Spalten. Einzelne Spalten eines Dataframes weisen dabei verschiedene Datentypen auf. Zu den wichtigsten Funktionen gehören Aggregationen, Gruppierungen und statistische Operationen.

In der vorliegenden Diplomarbeit wird Pandas bei der Datenverarbeitung eingesetzt, insbesondere in Kombination mit Scikit-learn, NumPy und Ruptures. Features, bei denen Pandas eingesetzt wird:

- Anomalien
- Change-Points
- Bereinigung von Datensätzen
- Aggregation von Datensätzen

[81]

2.2.3 Numpy

NumPy ist eine Programmierbibliothek für Python, welche grundlegende Datenstrukturen und Funktionen für numerische Berechnungen zur Verfügung stellt. Zu den wichtigsten Datenstrukturen gehören ndarrays, mehrdimensionale Arrays, welche effizienter als herkömmliche Python-Listen agieren und für das Arbeiten von großen Datenmengen geeignet sind. Häufig benötigt wird NumPy bei Verwendung von Machine Learning Algorithmen der Scikit-learn Bibliothek und oftmals auch bei bestimmten Datenmanipulationen wie das Umformen von Daten. [80] -learnScikit-learn Scikit-learn ist eine Library für Python, die zahlreiche Machine Learning Algorithmen implementiert hat. Der Hauptvorteil von Scikit-learn ist eine benutzerfreundliche und konsistente API zum einfachen Trainieren und Evaluieren von Modellen. Zu den verwendeten Algorithmen in diesem Projekt zählt der Supervised-Learning-Algorithmus der linearen Regression und der Ausreißererkennungsalgorithmus Isolation Forest. Außerdem wird Scikit-learn zur Bewertung der implementierten Algorithmen benötigt. [84]

2.2.4 Ruptures

Ruptures ist eine Python-Bibliothek zur Erkennung von Change Points, genauer gesagt strukturellen Brüchen in Zeitreihendaten. Sie bietet verschiedene Algorithmen zur automatischen Erkennung dieser Change-Points an, wie den in dieser Diplomarbeit verwendeten Pelt-Algorithmus zur Erkennung von Ereignissen. [83]

2.2.5 Matplotlib

Matplotlib ist eine Python-Bibliothek zur Visualisierung von Daten mithilfe von Grafiken und Diagrammen wie Linienplots, Streudiagrammen und Histogrammen. Verwendet worden ist Matplotlib in dieser Diplomarbeit in Verbindung mit Juypter zur Datenanalyse. Hauptsächlich ist dabei auf das Pyplot-Modul zugegriffen worden, welches eine Schnittstelle zur Erstellung von den besagten Plots und Diagrammen bereitstellt. Die Visualisierung der Daten ist notwendig, um Muster und Trends zu erkennen und unterstützt bei der Gewinnung von Erkenntnissen. [79]

2.3 Schnittstellen

In den nachfolgenden Abschnitten werden die im Projekt verwendeten Konzepte und Technologien für die Schnittstellen beschrieben. Fast Api, ist das im Projekt verwendete Web Framework, welches auf Pydantic, Starlette und Uvicorn basiert. Flask ist das Web Framework, welches für den Prototypen verwendet wurde und REST ist das im Projekt verwendete Architekturkonzept.

2.3.1 Fast API

FastAPI [37] (siehe Logo in Abbildung 2.2) ist ein Webframework zur Erstellung von RESTful APIs in der Programmiersprache Python, das Framework selbst ist ebenfalls in Python geschrieben. FastAPI verfügt über eine Vielzahl von Funktionen, wie beispielsweise automatische OpenAPI-Dokumentation, Datenvalidierung und verschiedene Sicherheitsfunktionen. FastAPI basiert auf und ist vollständig konform mit OpenAPI für

Abbildung 2.2: Fast API Logo [9]

die API-Erstellung, einschließlich Deklarationen von Pfadoperationen, Parametern, Body-Anfragen, Sicherheit, usw., und dem JSON-Schema.

FastAPI wurde von Sebastián Ramírez erstmals 2018 veröffentlicht und obwohl die Zahl der Nutzer schnell ansteigt, ist die Dokumentation möglicherweise nicht so ausführlich wie bei einigen anderen Frameworks.

Pydantic

Pydantic ist eine Datenvalidierungsbibliothek für Python. FastAPI enthält alle Funktionen von Pydantic, da dessen gesamtes Daten Handling auf Pydantic basiert. Beim Schreiben von Code in einer IDE stellt Pydantic Typhinweise für die Schemenvalidierung und Serialisierung durch Typ-Anmerkungen bereit.

Starlette

FastAPI ist vollständig kompatibel mit dem ASGI framework/toolkit Starlette. Tatsächlich ist FastAPI eine Unterklasse von Starlette und es sind auch alle Funktionen von Starlette enthalten („as FastAPI is just Starlette on steroids“). Starlette, und damit auch FastAPI, ist eines der schnellsten verfügbaren Python-Frameworks.

Uvicorn

Uvicorn[112] ist eine ASGI-Webserver-Implementierung für Python gemäß der die ASGI[110] (Asynchronous Server Gateway Interface) –Spezifikation für minimale Low-Level-Server- /Anwendungsschnittstelle für asynchrone Frameworks in Python. Uvicorn unterstützt derzeit HTTP/1.1 und WebSockets.

2.3.2 Flask

Flask[10] (siehe Logo in Abbildung 2.3) ist ein in Python geschriebenes Web-Micro-Framework und benötigt als solches keine besonderen Tools oder Bibliotheken. Flask wurde unter einer BSD-Lizenz von Armin Ronacher und anderen Mitgliedern der Pocoo-Gruppe entwickelt.

Abbildung 2.3: Flask Logo [10]

Durch die minimalistische Struktur gibt es in Flask keine Datenbankabstraktionsschicht, keine Formularvalidierung oder andere Komponenten, für die bereits vorhandene Bibliotheken von Drittanbietern gemeinsame Funktionen bereitstellen. Allerdings unterstützt Flask Erweiterungen, die Anwendungsfunktionen hinzufügen, als wären sie in Flask selbst implementiert. Es gibt Erweiterungen für objektrelationale Mapper, Formularvalidierung, Upload-Handhabung, verschiedene offene Authentifizierungstechnologien und mehrere gängige Framework-bezogene Tools.
Die Anwendungserstellung wird durch Entwicklungsserver, Debugger und integrierte Unterstützung für Unit-Tests sowie eine vollständige Dokumentation erleichtert.

2.3.3 REST

Der Begriff REST[62] (Representational State Transfer) wurde im Jahr 2000 von Roy Fielding eingeführt und stellt eine Abstraktion der Struktur und des Verhaltens des World Wide Web dar. Der sogenannte REST-Architekturstil definiert eine Software-Architektur, die als Richtlinie für netzwerkbasierte Anwendungen, insbesondere Client-Server-Anwendungen, dienen soll.

REST definiert eine Reihe von Einschränkungen, wie sich die Architektur eines verteilten Hypermedia-Systems im Internet-Maßstab wie dem Web verhalten soll. Der REST-Architekturstil legt Wert auf einheitliche Schnittstellen, die unabhängige Bereitstellung von Komponenten, die Skalierbarkeit der Interaktionen und die Schaffung einer mehrschichtigen Architektur. RESTful eine Anwendung, die die Beschränkungen der REST-Architektur einhält.

2.4 Server

In den nachfolgenden Abschnitten werden die im Projekt verwendeten Konzepte und Technologien für den Server beschrieben. Wobei Ubuntu das Betriebssystem des Servers und Nginx der verwendete Webserver sind.

2.4.1 Ubuntu

Abbildung 2.4: Ubuntu Logo [23]

Ubuntu (siehe Logo in Abbildung 2.4) ist ein Linux-Dialekt (eine sogenannte Linux-Distribution, d.□h. ein Betriebssystem), der auf Debian basiert und hauptsächlich aus freier und Open-Source-Software besteht. Debian ist eines der ältesten Betriebssysteme (das Debian-Projekt wurde 1993 von Ian Murdock gegründet), das auf dem Linux-Kernel basiert, der wiederum ursprünglich 1991 von Linus Torvalds für seinen i386-basierten PC geschrieben und bald als Kernel für ein GNU-Betriebssystem übernommen wurde, das als kostenloser (libre) Ersatz für das legendäre UNIX geplant war. Linux ist ein monolithischer Kernel mit modularem Aufbau (d. h. er kann zur Laufzeit ladbare Kernelmodule einfügen und entfernen), und unterstützt die meisten UNIX-Funktionen.

Ubuntu wird von der britischen Firma Canonical und einer Gemeinschaft anderer Entwickler entwickelt und offiziell in mehreren Editionen veröffentlicht: Desktop, Server und Core für Internet-of-Things-Geräte und Roboter.

In der Arbeit ist es das Betriebssystem des Servers, auf welchem die API und Angular Applikation gehostet wird.

2.4.2 nginx

Nginx[56] (siehe Logo in Abbildung 2.5) kann auch als Reverse-Proxy, Load Balancer, Mail-Proxy und HTTP-Cache verwendet werden. Nginx wurde 2004 vom russischen Entwickler Igor Sysoev veröffentlicht.
Nginx kann einfach konfiguriert werden, um statische Webinhalte bereitzustellen oder als Proxyserver zu fungieren, und auch zur Bereitstellung dynamischer Inhalte im Netzwerk eingesetzt werden.

Abbildung 2.5: Nginx Logo [16]

Nginx verwendet einen asynchronen, ereignisgesteuerten Ansatz anstelle von Threads, um Anfragen zu verarbeiten. Die modulare, ereignisgesteuerte Architektur von Nginx kann unter hoher Last eine vorhersehbare Leistung bieten.

Nginx Open Source ist kostenlose Open-Source-Software, daneben gibt es die kommerzielle Version Nginx Plus.
Nginx wird in dem Projekt als Webserver zum hosten der API und Webseite verwendet.

2.5 Datenhaltung

In den nachfolgenden Abschnitten werden die im Projekt verwendeten Konzepte und Technologien der Datenhaltung beschrieben. Die Datenbank läuft innerhalb eines Docker Containers, wobei es sich um eine Postgres Datenbank mit der Timescale Erweiterung handelt. Um auf die Daten zuzugreifen, fungiert Psycopg als Adapter und SQL als Abfragesprache.

2.5.1 PostgreSQL

PostgreSQL[2] (siehe Logo in Abbildung 2.6), auch als 'Postgres' bekannt, ist eine relationales Datenbank Management System (RDBMS) [61], welches weitgehend mit dem SQL-Standard konform ist und diesen auch mit Funktionen erweitert, welche das sichere Speichern und das Arbeiten mit großen Datenmengen ermöglichen.

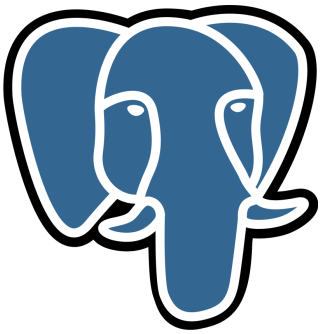


Abbildung 2.6: PostgreSQL Logo [17]

Postgres steht auf allen häufig verwendeten Betriebssystemen, wie Linux, Windows und Mac-OS zur Verfügung. Die bewältigbaren Arbeitsaufgaben reichen von Einzelplatzmaschinen bis hin zu Datawarehouses oder Webdiensten mit vielen gleichzeitigen Benutzern.

Die Datenbank wurde 1996 [60] unter diesem Namen veröffentlicht, das POSTGRES Projekt wurde jedoch schon in den 80-er Jahren von Michael Stonebraker begonnen, der dafür mit dem Turing Award ausgezeichnet wurde. Mittlerweile konzentriert sich die PostgreSQL Global Development Group nur auf die Entwicklung der Datenbank-Engine und eng verwandter Komponenten. Da PostgreSQL Open-source ist, bietet es eine weitreichende Unterstützung für Erweiterungen durch eine umfangreiche Entwicklergemeinschaft. Zu den so bereitgestellten Funktionserweiterungen gehören zum Beispiel spezielle Datenbank-Funktionen zur Unterstützung von Geodaten und andere diverse tool-sets.

[2] `https://www.postgresql.org/about/`

2.5.2 TimescaleDB

TimescaleDB [70] (siehe Logo in Abbildung 2.7) erweitert und optimiert PostgreSQL für das Speichern und die Analyse von Zeitreihen-Datensätzen, dass heißt von Datensätzen, die aus Paaren einer Zeitangabe [69], dem sogenannten Zeitstempel, und einem dazugehörenden Wert bestehen.

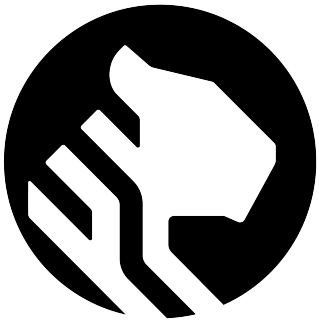


Abbildung 2.7: TimescaleDB Logo [22]

TimescaleDB wurde von Timescale Inc. als open-source Software entwickelt und in C [28] geschrieben.
Als Add-On von PostgeSQL wird auch bei TimescaleDB SQL als Abfragesprache verwendet. Die erweiternden Funktionen und Datenstrukturen sind auf schnelles Einfügen von großen Datenmengen und komplexe Abfragen ausgerichtet. Durch eine automatische Aufteilung der Daten nach der Zeit (space-time partitioning) wird horizontale Skalierung ermöglicht und so die Einfügegeschwindigkeit weiter erhöht. Daten werden unter anderem in sogenannten Hypertables gespeichert.

- **Hypertable**

  Hyptertables dienen als Abstraktion für einen einzelne, kontinuierliche Tabellen. Intern unterteilt TimescaleDB die Hypertables in Chunks, die einem bestimmten Zeitintervall und Partitionsschlüsseln entsprechen. Chunks werden durch die Verwendung regulärer PostgreSQL-Tabellen implementiert.

2.5.3 Psycopg

Psycopg [29] (siehe Logo in Abbildung 2.8) ist ein prominenter PostgreSQL Adapter für die Programmiersprache Python. Psycopg wurde hauptsächlich in C als ein libpq wrapper geschrieben. Psycopg zeichnet sich durch die vollständige Implementierung des Python DB API 2.0-Standards und die Sicherheit der Threads, auch wenn diese dieselbe Verbindung teilen, aus. Weiters bietet Psycopg robuste und schnelle Handhabung von Verbindungen, Transaktionen und Cursoren sowie Unterstützung für

Abbildung 2.8: Psycopg Logo [18]

die Ausführung von SQL-Befehlen und den effizienten Abruf von Ergebnissen. Darüber hinaus werden parametrisierte Abfragen erleichtert, SQL-Injektion-Schwachstellen verhindert und asynchrone Operationen unterstützt.

2.5.4 SQL

SQL, oder ausgeschrieben Structured Query Language [64] ist eine Programmiersprache, welche für den Zugriff auf Datenbanken, insbesondere relationale Datenbank Management Systeme (RDBMS) [76], konzipiert wurde. SQL ist eine deklarative Programmiersprache, im Kontrast zu den meisten Programmiersprachen, welche prozessorientiert sind. Deklarativ bedeutet, dass die Anweisungen beschreiben, was zu tun ist, und nicht, wie etwas zu tun ist.

SQL war eine der ersten kommerziellen Sprachen, die das relationale Modell von Edgar F. Codd nutzte. SQL wurde 1974 bei IBM von Donald D. Chamberlin und Raymond F. Boyce entwickelt und ist seit 1986 (American National Standards Institute (ANSI)) bzw. 1987 (International Organization for Standardization (ISO)) bis heute Standard für den Zugriff auf strukturierte Daten (jedoch nicht unverändert, sondern oftmals revidiert; darüber hinaus hält sich scheinbar keine Implementierung vollständig an die Standards).

2.5.5 Docker

Docker (siehe Logo in Abbildung 2.9) ist eine Plattform [47] zum Erstellen, Bereitstellen, Ausführen, Aktualisieren und Verwalten von Containern.

Abbildung 2.9: Docker Logo [8]

Container sind standardisierte, ausführbare Komponenten, die den Anwendungsquellcode mit den Bibliotheken des Betriebssystems in einer funktionellen Weise kombinieren, sodass dieser auf jedem Linux-, Windows- oder macOS-Computer ausgeführt werden kann. Ein Container [46] ist daher eine Form der Virtualisierung, welche den Anwendungen die erforderlichen Bibliotheken, Binärdateien, Konfigurationsdateien und Ressourcen in einer isolierten Umgebung bietet.

Bei der Ausführung unter Linux nutzt Docker Funktionen des Linux-Kernels zur Isolation von Ressourcen (z. B. cgroups und Kernel-Namespaces) zusammen mit einem Union-fähigen Dateisystem (z. B. OverlayFS), sodass die Container innerhalb einer einzelnen Linux-Instanz ausgeführt werden können. Die Verwendung von Namespaces, die vom Linux-Kernel unterstützt wird, isoliert größtenteils die Sicht einer Anwendung auf die Betriebsumgebung, einschließlich Prozessbäume, Netzwerk, Benutzer-IDs und gemountete Dateisysteme. Die cgroups des Kernels sorgen für eine Begrenzung der verfügbaren Ressourcen (Speicher und CPU).

Seit Version 0.9 enthält Docker eine eigene Komponente (libcontainer genannt) zur Nutzung der direkt vom Linux-Kernel bereitgestellten Virtualisierungsfunktionen, zusätzlich zur Verwendung abstrahierter/abgeleiteter Virtualisierungsschnittstellen mittels libvirt, LXC und systemd-nspawn.

Docker ist eine Plattform [47] zum Erstellen, Bereitstellen, Ausführen, Aktualisieren und Verwalten von Containern. Ein Container [46] ist eine Form der Virtualisierung, welche Anwendungen und die erforderlichen Bibliotheken, Binärdateien, Konfigurationsdateien und Ressourcen in einer isolierten Umgebung bietet. Für den Anwendungsfall wird eine Docker Instanz der TimescaleDB Datenbank verwendet.

2.6 Visualisierung

In den nachfolgenden Abschnitten werden die im Projekt verwendeten Konzepte und Technologien für die Visualisierung beschrieben. Für die Visualisierung der Ergebnisse wird das Angular Framework verwendet, das auf den Technologien HTML, CSS und TypeScript basiert. Für die Darstellung der erfassten Zeitseriendaten wird die JavaScript-Bibliothek ChartJS verwendet. Um die Implementierung des optischen Designs zu erleichtern wird Bootstrap, Angular Material und Flexbox eingesetzt. Für das Einblenden einer Warnung sollte kein Rechner ausgewählt sein, wird CoreUI benutzt.

2.6.1 Angular

Angular[3] (siehe Logo in Abbildung 2.10) ist ein Framework zur Entwicklung von Single Page Applications. Die Architektur des Frameworks ist in der Abbildung 2.11 zusehen. Es verwendet eine komponentenbasierte Architektur. Diese Komponenten sind eigenständige und wiederverwendbare Bestandteile der Applikation, die Struktur in die Anwendung bringen. Eine Komponente besteht aus einer HTML-, einer CSS- und einer TypeScript-Datei. Mithilfe von Dependency Injection teilen Dienste, Daten oder Funktionen zwischen den einzelnen Komponenten. Weiters wird ein Angular Projekt mit Modulen organisiert. Ein Modul besteht aus beliebig vielen Komponenten und Diensten. Angular bietet zudem ein leistungsfähiges Routing-Modul für die Navigation zwischen verschiedenen Ansichten. Um Daten aus einer REST API zu laden, ist in Angular ein HTTP-Client enthalten, der es ermöglicht es, mit dem Server zu kommunizieren. Um das Verhalten der einzelnen DOM-Elemente zu steuern, werden Direktiven verwendet. Einweg- und Zweirichtungsdatenbindung ermöglicht es, dynamische Daten im HTML zu verwenden. [75]

Abbildung 2.10: Angular Logo [2]

[3] https://angular.io/

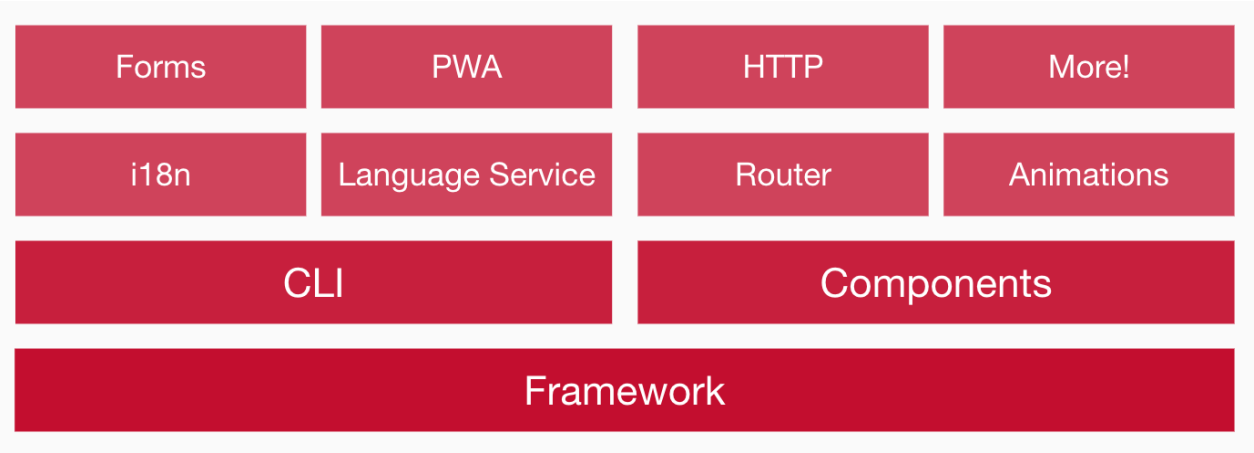


Abbildung 2.11: Architektur der Angular Plattform

HTML

HTML steht für Hypertext Markup Language und ist der Standard für die Strukturierung von Inhalten auf Webseiten. Browser interpretieren die verschiedenen Attribute und Tags, die Webseiten für Benutzer anzeigen. [51]

CSS & SCSS

CSS steht für Cascading Style Sheets und ist eine Sprache zur optischen Gestaltung von Webseiten. Es ermöglicht, das Aussehen und das Layout von HTML-Dokumenten zu kontrollieren. Es gibt verschiedene Möglichkeiten, um die Darstellung von Elementen festzulegen. Über Selektoren werden HTML-Elemente ausgewählt und mit einem bestimmten Design versehen werden. Weiteres gibt es die Option, eigene CSS-Klassen anzulegen. Diese können jedem beliebigen HTML-Element zugeordnet werden. [35]
SCSS steht für Sassy CSS und ist eine erweiterte Syntax für CSS, die die Arbeit mit CSS erleichtert. Es unterstützt zusätzliche Funktionen wie Variablen, verschachtelte Regeln, Mixins, Vererbung und Importe. [98]

TypeScript

TypeScript ist eine Programmiersprache, die auf JavaScript basiert. Der Hauptunterschied ist, dass TypeScript eine statische Typisierung unterstützt. Das ermöglicht es Entwicklern, Variablen und Funktionen mit Datentypen anzulegen. TypeScript stellt weiteres auch Klassen, Vererbung, Generics und Module zur Verfügung. Um sicherzustellen, dass der Code auf jedem gängigen Gerät ausgeführt werden kann, wird er in JavaScript kompiliert. [93]

Komponenten

Angular-Komponenten sind wie Bausteine in der Angular-Anwendung. Sie sind Teile der Benutzeroberfläche, die beliebig oft wiederverwendet werden. Eine Komponente wird erstellt, um eine bestimmte Funktion umzusetzen. Jede Komponente besteht aus einer HTML-Datei, einer CSS-Datei und einer TypeScript-Datei. Mithilfe von Datenbindung werden die Variablen und Funktionen aus der TypeScript-Datei in der HTML-Datei verwendet. Somit werden dynamische Daten angezeigt und auf Benutzeraktionen reagiert. Angular bietet verschiedene Methoden, die abhängig vom Lebenszyklus der Komponenten aufgerufen werden. [106]

Direktiven

In Angular sind Direktiven Funktionen, die auf DOM-Elemente oder Komponenten angewendet werden, um ihr Verhalten zu adaptieren. Es gibt verschiedene Arten von Direktiven in Angular, die unterschiedliche Zwecke erfüllen: [101]

- Attributdirektiven

  Attributdirektiven ändern das Aussehen oder das Verhalten von DOM-Elementen. Ein Beispiel hierfür ist die ’ngIf’-Direktive, die Elemente aufgrund einer Bedingung anzeigt oder verbirgt. [100]

- Strukturdirektiven

  Strukturdirektiven ändern die DOM-Struktur, indem sie Elemente hinzufügen oder entfernen. Ein Beispiel hierfür ist die ’ngFor’-Direktive, die das Wiederholen von HTML-Elementen mithilfe einer Datenquelle ermöglicht. [104]

Dependency Injection

Das grundlegende Konzept von Dependency Injection besteht darin, dass eine Klasse oder eine Komponente ihre Abhängigkeiten nicht selbst erstellt oder kennt, sondern diese von außen in die Klasse oder Komponente injiziert werden. Dadurch wird die Klasse oder Komponente von ihren Abhängigkeiten entkoppelt und somit flexibler und leichter testbar. In Angular wird Dependency Injection verwendet, um Komponenten, Dienste und andere Teile der Anwendung zu verwalten und miteinander zu verbinden. Wird ein Dienst in einer Komponente benötigt, wird dieser im Konstruktor deklariert und von einem, von Angular bereitgestellten, Injector in der Anwendung gesucht. [107]

Routing

Routing ermöglicht die Navigation zwischen verschiedenen Ansichten in einer Single-Page-Anwendung. Alle Ressourcen werden beim ersten Laden der Anwendung geladen, anstatt für jede Ansicht eine eigene Seite zu laden. Jede Ansicht wird durch eine eindeutige URL repräsentiert. Die Routen werden im Router-Outlet definiert und in den einzelnen Komponenten über die ’routerLink’-Direktiven aufgerufen. Es gibt auch die Möglichkeit, dynamische Parameter in den URLs zu verwenden. [103]

Services

In Angular ist ein Service eine Klasse, die eine spezifische Funktionalität bereitstellt, die in verschiedenen Komponenten mehrmals verwendet werden. Dienste eignen sich ideal für Datenzugriffe, Authentifizierung und andere Funktionen, die unabhängig von Komponenten ausgeführt werden. Um den Service in den Komponenten zu verwenden, wird dieser injiziert. [105]

HttpClient

Der HttpClient wird von Angular für die Kommunikation mit RESTful APIs zur Verfügung gestellt. Es werden Anfragen, mit den standardisierten HTTP-Methoden wie GET, POST, PUT und DELETE, an einen Server gesendet. Dies ermöglicht den Austausch von Daten zwischen dem Angular-Projekt und einem externen Server. Für die Rückgabe von Daten

verwendet der HttpClient Observables. Dadurch werden reaktive Aktualisierungen in der Benutzeroberfläche ermöglicht, wenn Daten geladen werden. [108]

CLI

Die Angular CLI (Command Line Interface) ist ein Entwicklungswerkzeug, das Entwicklern dabei hilft, Angular-Anwendungen effizient zu erstellen und zu verwalten. Sie bietet Befehle für das Generieren von Code, den Start eines Entwicklungsservers und vieles mehr. [102]

In dieser Diplomarbeit wird Angular verwendet, um die gesammelten und analysierten Daten in einer Webapplikation zu visualisieren.

2.6.2 Bootstrap

Bootstrap[4] (siehe Logo in Abbildung 2.12) ist ein CSS Framework, um schnell und einfach responsive Webseiten zu erstellen. Es bietet unterschiedliche, vordefinierte Komponenten, die angepasst und eingebunden werden. Ein responsives Rastersystem, das in Bootstrap inkludiert ist, hilft flexible Layouts zu erstellen. [32] Mithilfe von Sass werden Komponenten importiert und globale Variablen, wie Schatten und Farben, definiert. Weiteres bietet SASS Funktionen zur Durchführung von Berechnungen und Mixins zur Wiederverwendung von CSS-Regeln. Bei bereits vorhandenen Komponenten gibt es die Möglichkeit, diese an eigene Bedürfnisse anzupassen. [31]

Abbildung 2.12: Bootstrap Logo [4]

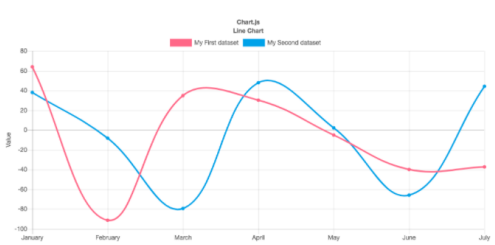
In dieser Diplomarbeit wird Bootstrap 5 verwendet, um die Webapplikation responsiv zu gestalten und mithilfe von SASS-Variablen die CSS-Klassen übersichtlicher zu strukturieren.

[4] https://getbootstrap.com/

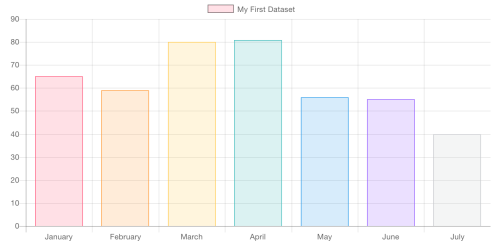
2.6.3 Chart.js

Chart.js[5] (siehe Logo in Abbildung 2.13) ist eine JavaScript-Bibliothek, mit deren Hilfe unterschiedliche Diagrammtypen in Webanwendungen dargestellt werden. Unter anderem sind folgende Diagrammarten, wie in den Abbildungen in Tabelle 2.1 zu sehen, definiert:

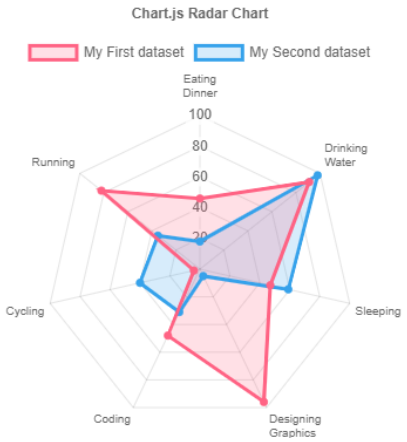
Abbildung 2.13: Chart.js Logo [5]



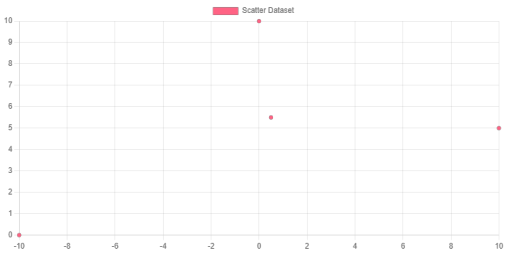
(a) Lininendiagramm



(b) Balkendiagramm



(c) Bereichsdiagramm



(d) Streudiagramm

Tabelle 2.1: ChartJS Beispiele [77]

Die Bibliothek bietet unterschiedliche, interaktive Funktionen wie Tooltips und Animationen. Darüber hinaus gibt es viele Möglichkeiten, mit verschiedenen Optionen und Konfigurationen, das Diagramm nach eigenen Wünschen anzupassen. Chart.js unterstützt responsives Design und kann sich an unterschiedliche Bildschirmgrößen anpassen.[33]

[5] https://www.chartjs.org/

In dieser Diplomarbeit ist Chart.js im Angular Projekt eingebunden, um unterschiedliche Messwerte grafisch darzustellen.

2.6.4 Angular Material

Angular Material[6] ist eine Bibliothek mit UI-Komponenten für Angular Projekte. Sie bietet verschiedene vorgefertigte UI-Komponenten, die innerhalb der Design-Spezifikationen angepasst werden. Einige dieser Komponenten sind:

- Icons
- Inputs
- Radio Buttons
- Slider

Die UI-Komponenten passen sich unterschiedlichen Bildschirmgrößen an. Angular Material ist für Angluar-Projekte ausgelegt und verwendet Angular-spezifische Funktionen und lässt sich einfach in Angular-Projekte integrieren. [109]

In dieser Diplomarbeit werden Angular Material Komponenten verwendet, um die Benutzeroberfläche ansprechender zu gestalten.

2.6.5 CoreUI

CoreUI[7] ist ein Open-Source-Framework für das Front-End-Design von Webanwendungen. Es bietet eine Reihe von vorgefertigten Komponenten, Layouts und Stilen, um die Entwicklung von modernen, ansprechenden und benutzerfreundlichen Benutzeroberflächen zu erleichtern. Es baut auf Bootstrap auf, verwendet dessen Struktur und Komponenten und passt diese an oder erweitert sie. CoreUI ist nicht auf ein bestimmtes JavaScript-Framework beschränkt. Es bietet spezielle Versionen und Integrationen für Angular, React und Vue.js. [6]
In dieser Diplomarbeit werden einzelnen Komponenten verwendet, um die Webanwendung anschaulicher zu gestalten.

[6] `https://material.angular.io/`
[7] `https://coreui.io/angular/`

2.6.6 Flexbox

Flexbox ist ein Konzept, genauer gesagt ein Layout-Modell, in CSS, das dazu verwendet wird, um die Anordnung und Positionierung von Elementen auf einer Webseite zu steuern. Es ermöglicht, komplexe Layout-Herausforderungen einfacher zu meistern. Es wird speziell bei der Gestaltung von responsiven Designs eingesetzt. Flexbox arbeitet mit einem Container, der die Elemente umschließt. Für die Anordnung innerhalb des Containers ist eine Hauptachse und eine Kreuzachse zuständig. Die Hauptachse gibt die horizontale Richtung an und die Kreuzachse steuert die senkrechte Ausrichtung der Elemente. Flexbox ermöglicht es Elemente auf einer Webseite anzuordnen ohne aufwendigen HTML-Code oder umständlichen Berechnungen. [30]

2.7 Entwicklungssysteme

In den folgenden Kapiteln werden die verwendeten Entwicklungssysteme in dieser Diplomarbeit beschrieben.

2.7.1 PyCharm

Abbildung 2.14: PyCharm Logo [19]

PyCharm[8] (siehe Logo in Abbildung 2.14) ist eine integrierte Entwicklungsumgebung (IDE), die hauptsächlich für die Python-Programmierung konzipiert ist. Sie wurde von JetBrains entwickelt und bietet eine Reihe von Funktionen zur Verbesserung der Python-Entwicklung, einschließlich Code-Analyse, Debugging und Integration der Versionskontrolle. PyCharm bietet intelligente Code-Vervollständigung, Syntax-Hervorhebung und Refactoring-Tools. Es unterstützt verschiedene Frameworks und Bibliotheken, die in der Python-Entwicklung häufig verwendet werden. Darüber hinaus lässt sich PyCharm in Versionskontrollsysteme wie Git integrieren, sodass die Benutzer in der Lage sind, den Projektverlauf effektiv zu verwalten. PyCharm Community Edition ist Open Source, während die kommerzielle Variante, Professional Edition, Closed-Source-Module enthält. [90]

2.7.2 IntelliJ IDEA

Abbildung 2.15: IntelliJ IDEA Logo [12]

IntelliJ IDEA[9] ist eine vom Unternehmen JetBrains entwickelte IDE (= Integrated Development Environment). Diese funktioniert als Code-Editor vor allem für jene Programmiersprachen, die auf der JVM (= Java Virtual Machine) basieren, wie Java, Kotlin, Scala und Groovy. Wobei auch andere Programmiersprachen, Frameworks und Dateitypen unterstützt werden. Zum Beispiel Python, SQL, HTML, Angular, XML und Markdown. Zusätzlich bietet IntelliJ diverse Funktionen, die den Entwicklungsprozess von Anwendungen wesentlich vereinfachen. Dazu zählen Versionsverwaltung, Code-Vervollständigung, Refactoring und die Unterstützung von verschiedensten Build-Tools. [52] Das Logo dieses Entwicklungssystems ist in Abbildung 2.15 dargestellt.

[8] https://www.jetbrains.com/de-de/pycharm/
[9] https://www.jetbrains.com/idea/

Für die Implementierung des Agents in der Programmiersprache Java wurde IntelliJ IDEA Ultimate in der Version 2023.2 verwendet.

2.7.3 WebStorm

WebStorm[10] (siehe Logo in Abbildung 2.16) ist eine Entwicklungsumgebung speziell für Webentwicklung von JetBrains. Die IDE bietet Unterstützung für unterschiedliche Webtechnologien wie HTML, CSS, JavaScript, TypeScript und Node.js. Sie enthält Funktionen wie Code-Vervollständigung, Refaktorisierung, und Syntax-Hervorhebung ebenso wie ein integriertes Versionskontrollsystem. Dieses ermöglicht es Entwicklern direkt in der IDE Änderungen zu verfolgen und Commits durchzuführen. Durch eine Live-Edit-Funktion werden Änderungen, die am Code vorgenommen werden, direkt im Browser angezeigt. Weiters sind in WebStorm Code-Standards enthalten. Mit integrierten Tools wird der Code automatisch auf seine Qualität geprüft. [91]

Abbildung 2.16: WebStorm Logo [24]

In dieser Diplomarbeit wird das Angular-Projekt in WebStorm entwickelt.

2.7.4 Datagrip

DataGrip[11] (siehe Logo in Abbildung 2.17) ist ein von JetBrains entwickeltes Datenbank-Management-Tool. Es ermöglicht Benutzern, sich mit verschiedenen Arten von Datenbanken zu verbinden, SQL-Abfragen auszuführen und Datenbankstrukturen zu verwalten. DataGrip bietet Funktionen wie Code-Vervollständigung, Syntax-Hervorhebung und Integration der Versionskontrolle für eine effiziente Datenbankentwicklung. Darüber hinaus bietet es Werkzeuge für die Datenexploration und -manipulation. Die Funktion zur Integration der Historie verfolgt die in der Datenbank durchgeführten Aktionen, sodass die Benutzer Änderungen bei Bedarf überprüfen und rückgängig machen können. Insgesamt erleichtert DataGrip die Datenbankverwaltung und

Abbildung 2.17: Datagrip Logo [7]

[10] https://www.jetbrains.com/de-de/webstorm/
[11] https://www.jetbrains.com/de-de/datagrip/

Entwicklungsaufgaben in einer benutzerfreundlichen Umgebung. [89]

In der Diplomarbeit wurde es zum Verwalten der Datenbank und Daten genutzt.

2.7.5 Jupyter

Jupyter[12] (siehe Logo in Abbildung 2.18) ist eine interaktive Open-Source-Plattform, welche es ermöglicht, Datenanalyse, Modellierung und Visualisierung in Python durchzuführen. Die Kernkomponente davon ist das Jupyter Notebook, welches eine webbasierte Benutzeroberfläche zur Verfügung stellt, um Code, Text und visuelle Ergebnisse in einem einzigen Dokument zu kombinieren. Dadurch ermöglicht es eine interaktive Programmierung vor allem in Bezug auf Data Science, da Code in einzelnen Zellen ausgeführt werden kann und die Ergebnisse dazu sofort angezeigt werden. Praktisch ist dieses Feature vor allem bei der Visualisierung von Daten in Form von Tabellen oder Diagrammen, da diese direkt im Dokument angezeigt und gespeichert werden. [78]

Abbildung 2.18: Jupyter Logo [7]

2.8 Sonstige Software

2.8.1 GitLab

GitLab[13] (siehe Logo in Abbildung 2.19) ist eine Open-Source-Software für die Versionsverwaltung von Quellcodedateien und bietet Funktionen für die Zusammenarbeit in Softwareprojekten. GitLab ermöglicht es, Änderungen im Code zu verfolgen und nachzuvollziehen. In Repositories wird der Code hochgeladen und geteilt, somit ist es mehrere Personen möglich an einem Projekt zuarbeiten. [111]

Abbildung 2.19: GitLab Logo [11]

In dieser Diplomarbeit wird das, von der HTL Perg gehostete, GitLab verwendet auf dem der Code, der im Rahmen dieser Arbeit entwickelt wurde liegt. Die Verwendung von Git-

[12] https://jupyter.org/
[13] https://about.gitlab.com/

Lab erleichtert die Zusammenarbeit der Projektmitglieder erheblich, da es eine zentrale Plattform für die Versionskontrolle bietet. Durch die Nutzung von GitLab haben alle Teammitglieder die Möglichkeit gleichzeitig am Code zuarbeiten, Änderungen zu verfolgen und auf frühere Versionen des Codes zurückzugreifen, was eine effiziente Koordination und Zusammenarbeit während des gesamten Entwicklungsprozesses ermöglicht.

2.8.2 Adobe XD

Adobe XD[14] (siehe Logo in Abbildung 2.20) ist ein Design- und Prototyping-Tool, das von Adobe Inc. entwickelt wurde. Das einfache Erstellen von Prototypen für Benutzeroberflächen für Websites und mobile Apps ermöglicht dem Nutzer erste Erfahrungen mit der Schnittstelle. Von der Prototypenerstellung bis zur Simulation von Interaktionen bietet Adobe XD alles, um eine Webapplikation oder eine mobile App zu entwerfen. [41]

Abbildung 2.20: Adobe XD Logo [1]

Während der Planung dieser Diplomarbeit haben wir Adobe XD genutzt, um die Benutzeroberfläche der Webapplikation zu entwerfen.

2.8.3 LucidChart

Lucidchart [55] (siehe Logo in Abbildung 2.21) ist eine vielseitige webbasierte Anwendung zur Erstellung verschiedener Diagrammtypen, von Flussdiagrammen bis hin zu Organigrammen. Es wird von Lucid Software Inc. produziert. Lucidchart ist vollständig browserbasiert und unterstützt die Zusammenarbeit in Echtzeit, sodass alle Benutzer gleichzeitig an Projekten arbeiten und die Ergänzungen jedes Benutzers in Echtzeit sehen. Seine umfassende Funktionalität erleichtert die Darstellung und Optimierung von Prozessen, Systemen und Organisationsstrukturen. Die Plattform bietet eine Kollaborationsfunktion, welche das Zusammenarbeiten vereinfacht. Lucidchart unterstützt mehrere Diagrammtypen, darunter Flussdiagramme, Mindmaps und Organigramme.


Abbildung 2.21: LucidChart Logo [15]

[14] `https://helpx.adobe.com/at/support/xd.html`

Bei dieser Diplomarbeit wurde LucidChart für das Erstellen des Datenbankschemas verwendet.

2.8.4 StarUML

StarUML[15] (siehe Logo in Abbildung 2.22) ist eine Software, die zur Modellierung von Software und Systemen verwendet wird. Es bietet eine Vielzahl an Funktionen zur Erstellung von UML-Diagrammen. Es unterstützt Entwickler bei der Darstellung von Struktur und Verhalten von Software. Fast alle UML-Diagrammarten werden von StarUML unterstützt, unter anderem Use-Case-Diagramme, Sequenz-Diagramme und Klassendiagramme. [66]

Abbildung 2.22: Star UML Logo [21]

Während der Planung dieser Diplomarbeit haben wir StarUML verwendet, um Use-Case-Diagramme zu erstellen.

[15] https://staruml.io/

3 Planung und Realisierung

3.1 Funktionalität

LogSense teilt sich in drei Kern-Funktionalitäten auf, welche in den nachfolgenden Use-Case-Diagrammen dargestellt werden.

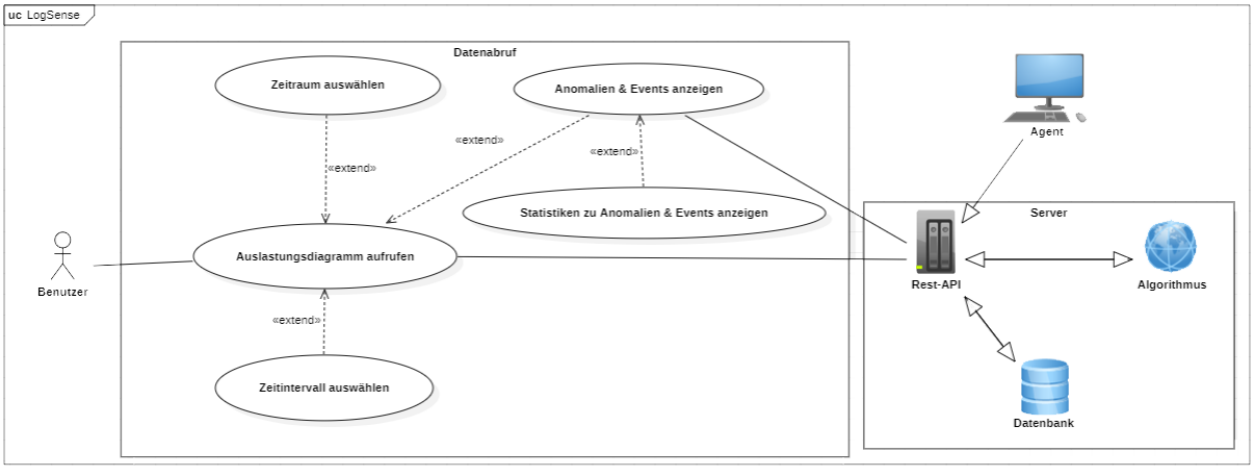


Abbildung 3.1: Use-Case-Diagramm für den Abruf der Daten

Abbildung 3.1 visualisiert den Use Case ”Datenabruf”. Der Benutzer kann Auslastungsdiagramme von RAM, CPU oder freien Speicher aufrufen, indem er die dazugehörige Ansicht auswählt. Danach gibt es die Möglichkeit, einen Zeitraum für das Laden der Daten auszuwählen sowie die Möglichkeit, ein Zeitintervall anzugeben, in dem die Daten gruppiert werden. Zusätzlich kann der Benutzer alle aufgetretenen Anomalien und Ereignissen im Graphen einblenden. Wenn der Benutzer seinen Zeiger über eine Anomalie oder Ereignis bewegt, werden Statistiken zu diesem Ereignis beziehungsweise der Anomalie angezeigt, unter anderem die Applikationen, die zu diesem Zeitpunkt die größte Auslastung hatten. Der Agent liest den Hardware-Verbrauch vom Client-Rechner ein und sendet die gemessenen Daten zur Rest-API, welche die Daten in der Datenbank persistiert. Der Benutzer kann diese Daten über die Benutzeroberfläche, über die Rest-API anfordern. Die Schnittstelle zum Server wird über eine Rest-API realisiert. Die Rest-Schnittstelle wiederum kommuniziert mit der Datenbank, um die benötigten Daten zu laden und analysiert diese darauffolgend mit den implementierten Algorithmen.

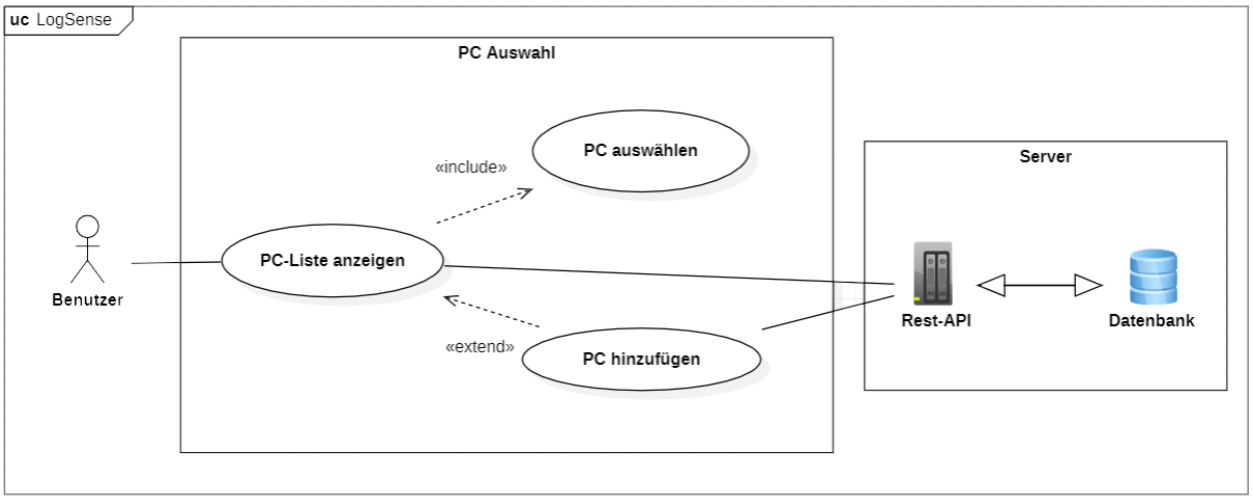


Abbildung 3.2: Use-Case-Diagramm für die Auswahl eines Rechners

Der Use Case in Abbildung 3.2 stellt den Prozess der PC-Auswahl dar. Zuerst wird dem Benutzer eine Liste von PCs angezeigt. Von dieser Liste hat dieser dann die Möglichkeit, einen bestimmten Rechner auszuwählen. Es gibt außerdem die Möglichkeit, einen neuen PC hinzuzufügen. Die Liste der PCs wird über eine API-Anfrage angefordert, wobei die Rest-API die benötigten Daten aus der Datenbank lädt. Beim Hinzufügen eines neuen Rechners werden die Rechner-Informationen per API-Anfrage übermittelt und in der Datenbank gespeichert.

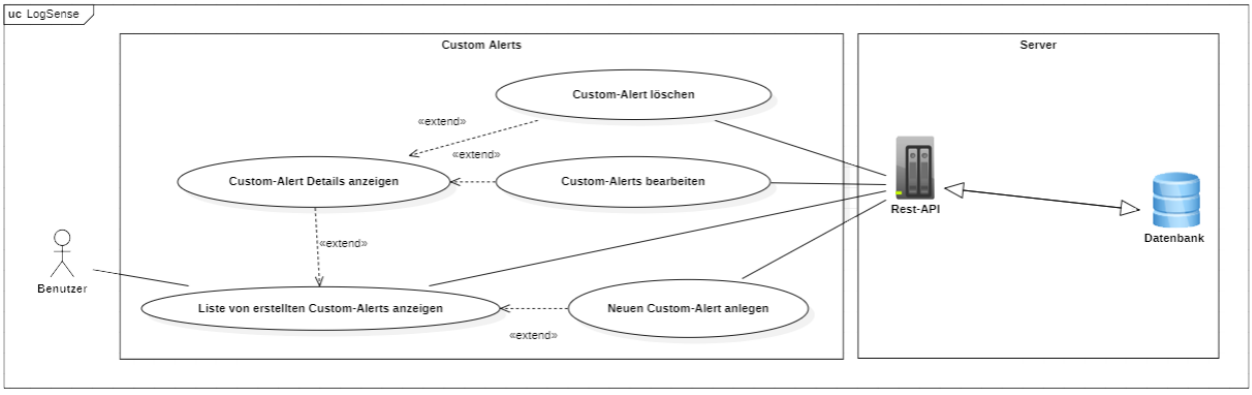


Abbildung 3.3: Use-Case-Diagramm für benutzerdefinierte Benachrichtigungen

Der Use Case in Abbildung 3.3 stellt die Verwendung von benutzerdefinierte Benachrichtigungen dar. Nach dem Aufrufen des Benachrichtigung-Screens wird dem Benutzer eine Liste von bestehenden benutzerdefinierte Benachrichtigungen angezeigt. Bei der Auswahl

einer Benachrichtigung werden die Details von diesem angezeigt. Danach kann dieser auch bearbeitet oder gelöscht werden. Der Benutzer kann auch eine neue benutzerdefinierte Benachrichtigung anlegen. Bei Anzeige der benutzerdefinierten Benachrichtigungen wird eine API-Anfrage per Rest gesendet und die angeforderten Daten werden aus der Datenbank geladen. Beim Anlegen oder Bearbeiten einer Benachrichtigung werden die Informationen über die Rest-API verschickt und in der Datenbank gespeichert.

3.2 Entwurf

In den nachfolgenden Abschnitten wird der Prototyp der Benutzeroberfläche mit seinen Funktionalitäten dargestellt.

3.2.1 Startseite

Die finale Version des Entwurfs unterscheidet sich deutlich von der ersten Idee. In der Abbildung 3.4 kann vor allem erkannt werden, dass alle relevanten Details in der finalen Version bereits auf der Startseite angezeigt werden. Beim Hovern über das Info-Icon werden zum Beispiel sämtliche Informationen über den Rechner angeführt. Außerdem werden die aktuellen Statistiken zum Ressourcenverbrauch angezeigt. Die Gesamtdauer der Prozesse, die in einem auswählbaren Zeitbereich gestartet worden sind, werden nur auf der Startseite in einem Balkendiagramm dargestellt.

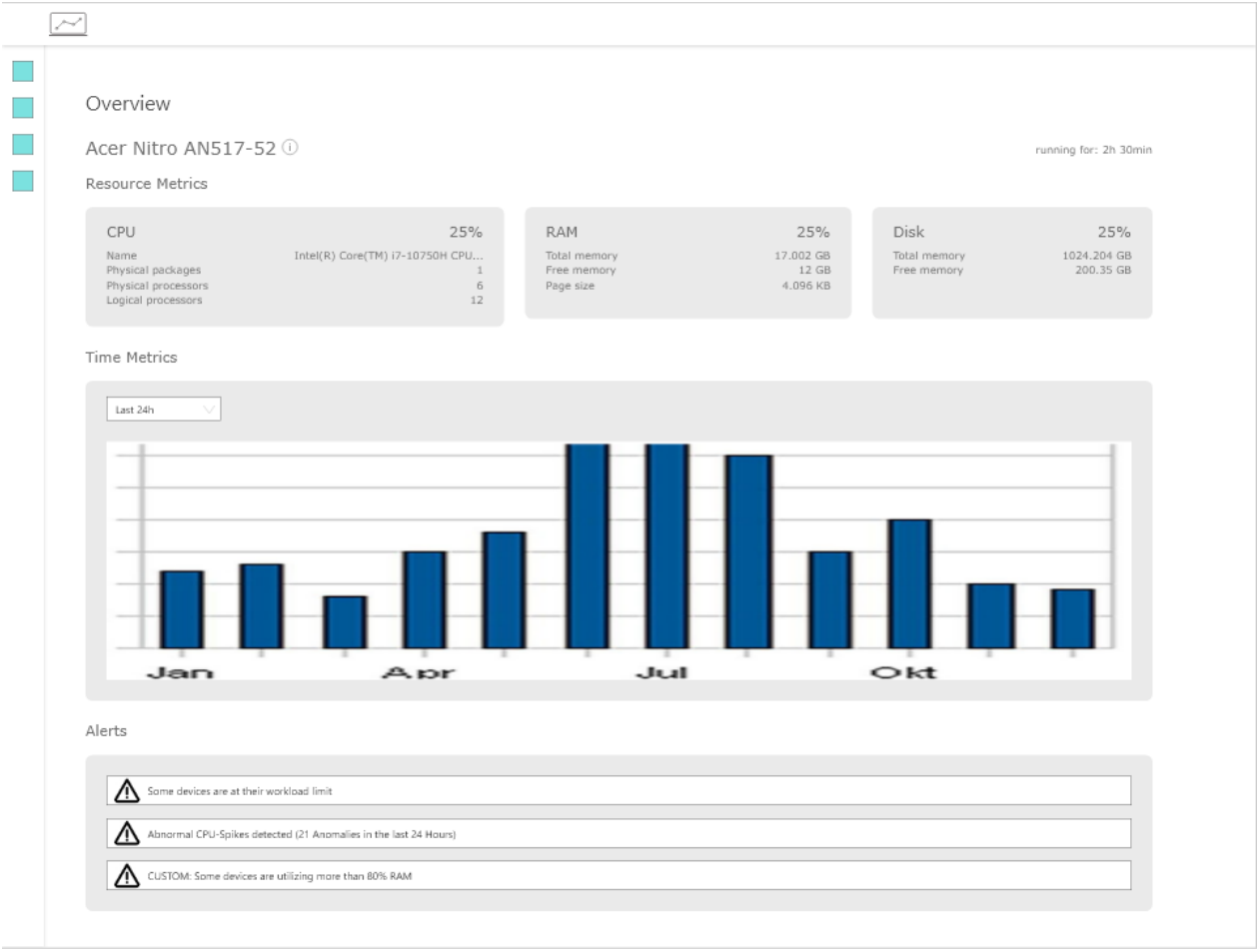


Abbildung 3.4: Finaler Entwurf der Startseite

In der Abbildung 3.5 wird die 1. Version des Entwurfs dargestellt. Diese ist überarbeitet worden, weil das Balkendiagramm mit den Zeitaufzeichnungen der Prozesse zu klein angezeigt wird und deshalb die Daten kaum ablesbar sind. Zusätzlich ist es sehr unübersichtlich, wenn alle Informationen so kompakt dargestellt werden. Deshalb sind die Daten auf unterschiedliche Ansichten im finalen Entwurf aufgeteilt worden, um die Webanwendung benutzerfreundlicher zu gestalten.

Abbildung 3.5: 1. Version des Entwurfs der Startseite

3.2.2 Ressourcenverbrauch

Für jede einzelne Ressource gibt eine eigene Ansicht in der Webanwendung. Die Abbildung 3.6 zeigt, zum Beispiel, die Ansicht mit den Informationen über die CPU. Der wichtigste Punkt ist hierbei die CPU-Auslastung in einem auszuwählenden Zeitraum, die in einem Liniendiagramm dargestellt wird. Dabei gibt es die Möglichkeit, den Zeitabstand zwischen den Datenpunkten festzulegen. Außerdem kann der Benutzer auswählen, ob im Diagramm auch Ereignisse und Anomalien markiert werden sollen. Weiters werden Statistiken der analysierten Daten wie die Stabilität und der Durchschnittsverbrauch angezeigt. CPU-spezifische Benachrichtigungen und vom Benutzer angelegte Benachrichtigung, die sich auch die CPU beziehen, werden ebenfalls in dieser Ansicht aufgelistet. Auch die generellen Informationen über die CPU sowie die CPU-Auslastung der einzelnen Prozesse werden hier angezeigt.

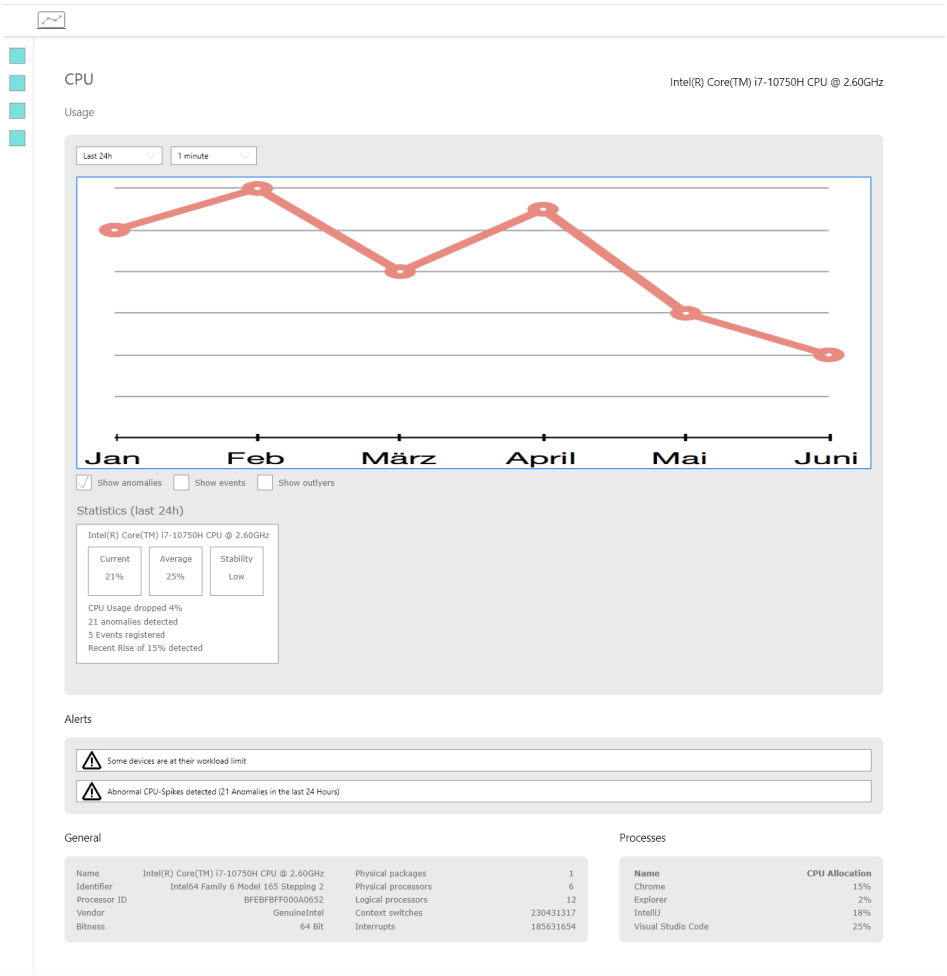


Abbildung 3.6: Finaler Entwurf der CPU-Ansicht

Fast wie bei der CPU-Ansicht werden auf der RAM-Ansicht, die in der Abbildung 3.7 dargestellt wird, der RAM-Verbrauch, die Statistiken der analysierten Daten, Benachrichtigungen und die RAM-Auslastung der einzelnen Prozesse abgebildet. Statt den generellen Informationen werden hier die Hauptspeichergröße, der freie Speicherplatz und die Größe einer Speicherseite angegeben.

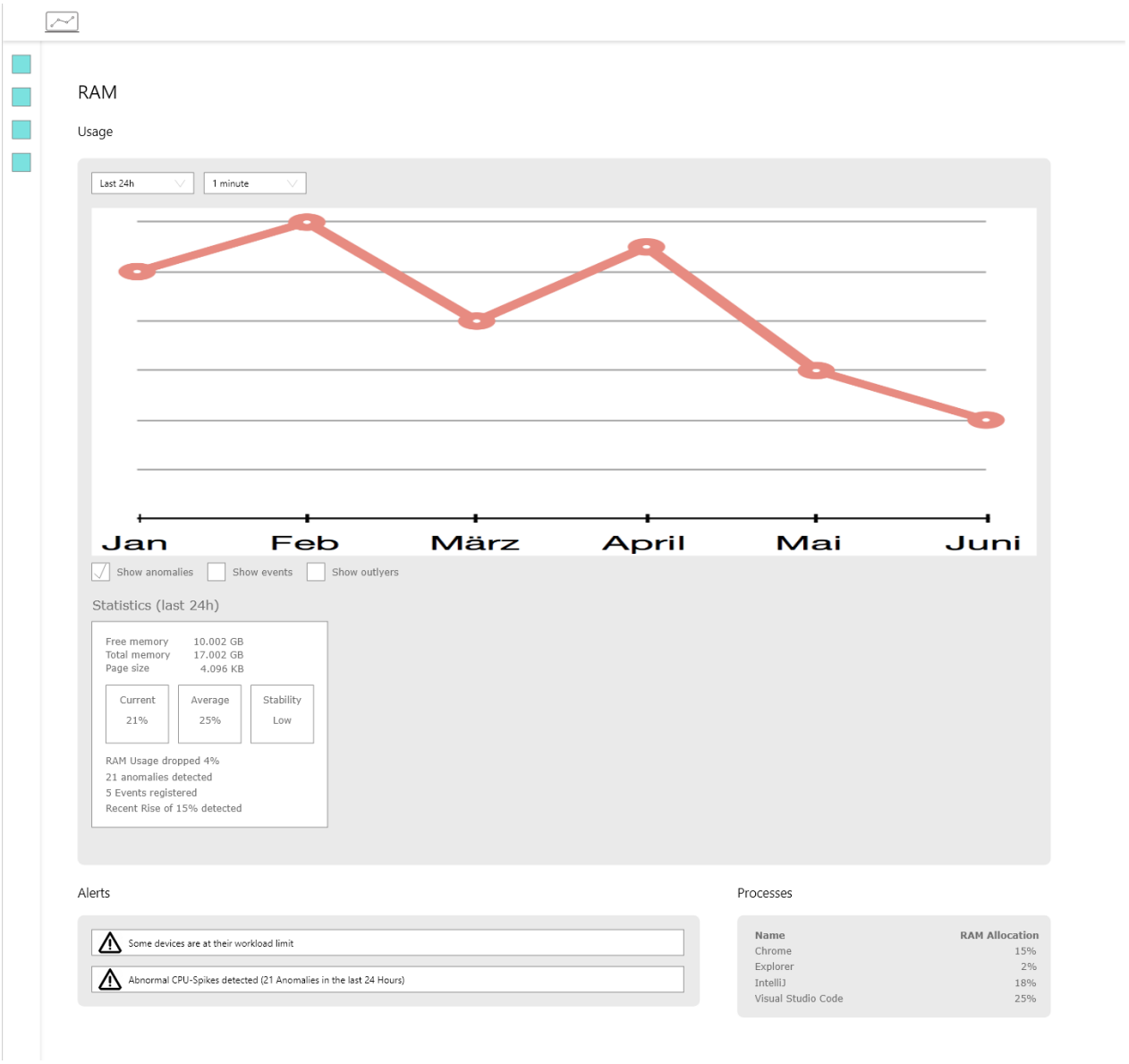


Abbildung 3.7: Finaler Entwurf der RAM-Ansicht

Auf der Festplatten-Ansicht, die in der Abbildung 3.8 angeführt ist, sieht man den Verlauf des freien Speichers in einem auszuwählenden Zeitraum in einem Liniendiagramm. Wie auch bei den oben angeführten Ressourcen kann man den Zeitabstand zwischen den Datenpunkten festlegen. Zusätzlich kann im Liniendiagramm eine Vorhersage des freien Speichers in den nächsten Tagen eingeblendet werden. Auch in dieser Ansicht werden die Festplatten-spezifischen Benachrichtigungen aufgelistet. Generelle Informationen zu allen erkannten Speichermedien wie zum Beispiel das Modell und der Gesamtspeicherplatz werden hier eingesehen. Zu jedem dieser Speichermedien werden weitere Informationen über deren Partitionen eingeholt und aufgelistet.

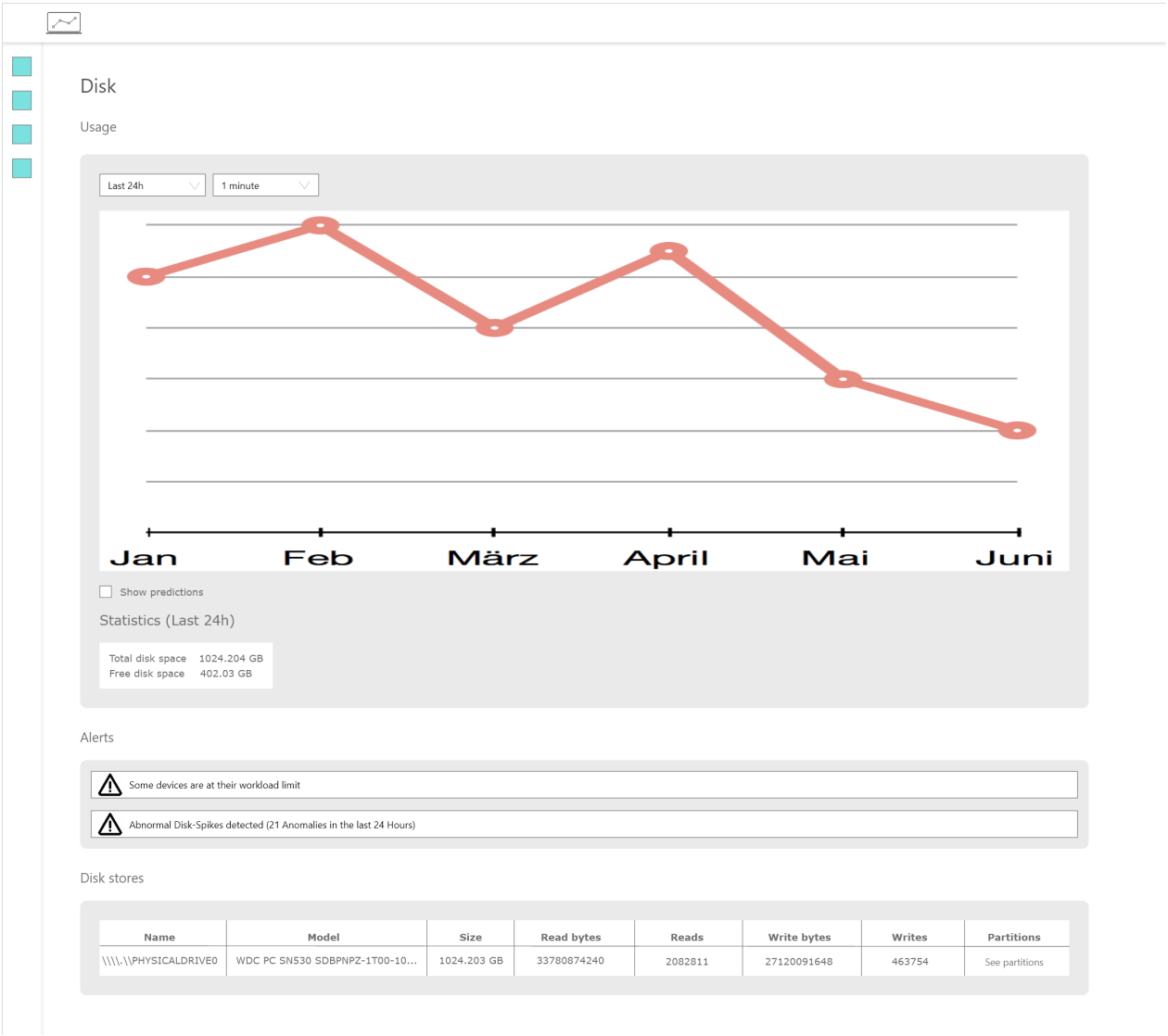


Abbildung 3.8: Finaler Entwurf der Festplatten-Ansicht

Exemplarisch ist der 1. Entwurf der CPU-Ansicht in der Abbildung 3.9 dargestellt. Dieser ist für die CPU-, RAM- und Festplatten-Ansichten überarbeitet worden, um dem Liniendiagramm mehr Raum zu geben. Dadurch können die Informationen in den Graphen besser entnommen werden. Außerdem ist m ersten Entwurf, die Auswahl der Zeit zwischen den Datenpunkten, noch nicht berücksichtigt worden.

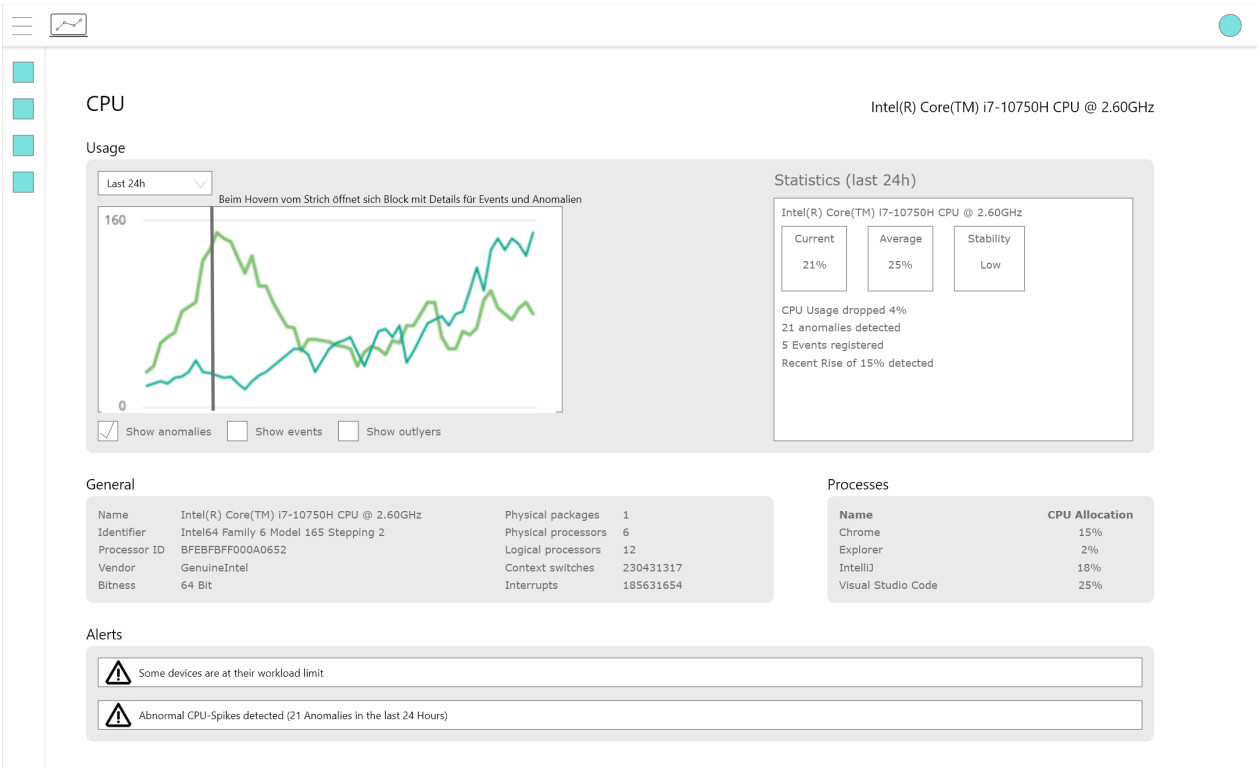


Abbildung 3.9: 1. Version des Entwurfs der Ressourcen-Ansicht

3.2.3 Prozesse

Alle Prozesse, die im ausgewählten Zeitraum erfasst worden sind, werden in der Prozess-Ansicht aufgelistet, die in der Abbildung 3.10 ersichtlich ist. In der Detail-Ansicht eines konkreten Prozesses werden die CPU- und RAM-Auslastungen in dem ausgewählten Zeitraum in Liniendiagrammen dargestellt. Dabei gibt es die Möglichkeit, den Zeitabstand zwischen den Datenpunkten festzulegen. Außerdem besteht die Möglichkeit, im Diagramm erkannte Ereignissen und Anomalien zu markieren. Sämtliche Benachrichtigungen, die den ausgewählten Prozess betreffen werden, hier aufgelistet. Zusätzlich werden generelle Informationen, zum Beispiel der Status und die Anzahl der Threads, und Statistiken der analysierten Daten angezeigt.

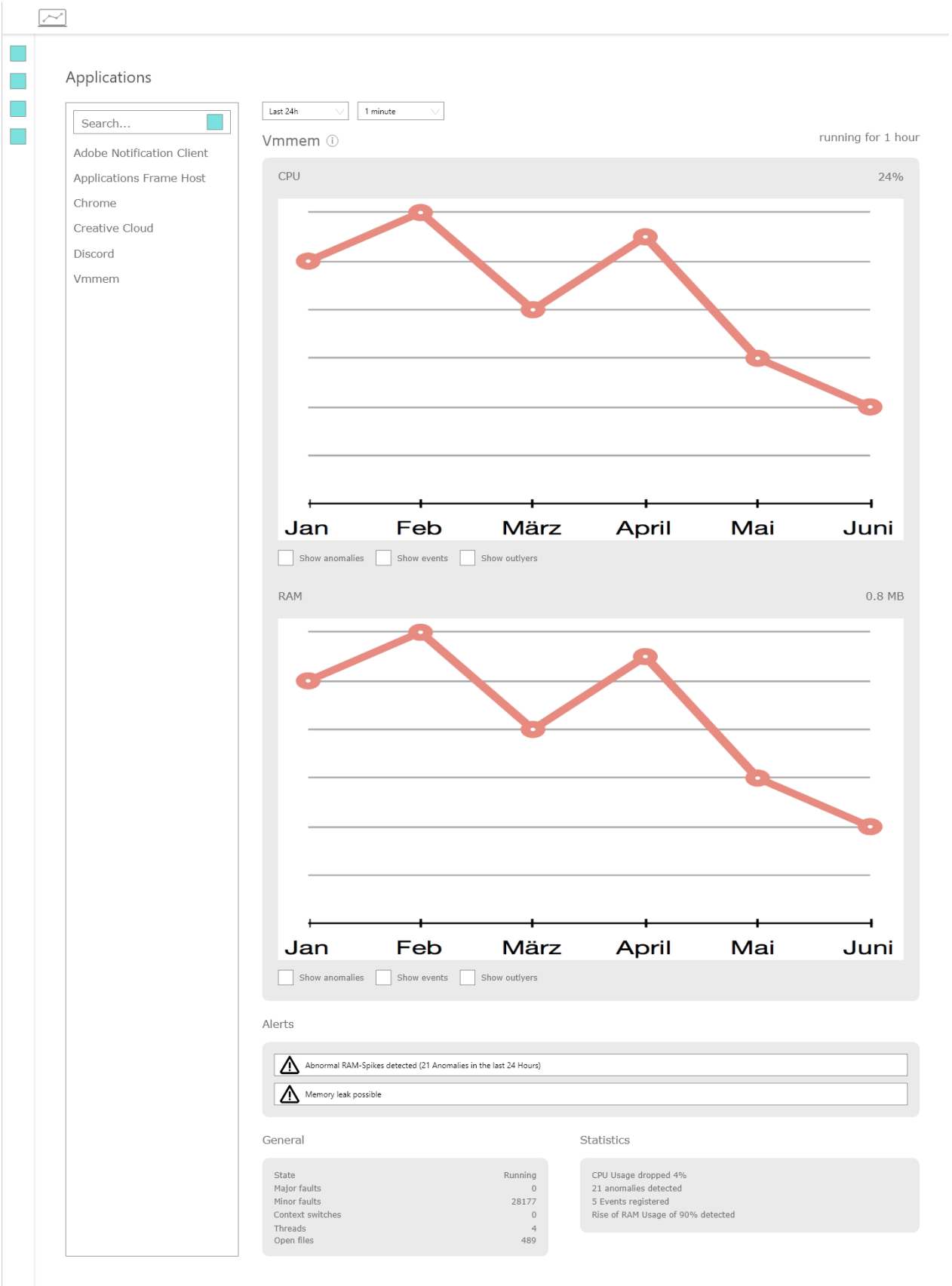


Abbildung 3.10: Finaler Entwurf der Prozess-Ansicht

In der Abbildung 3.11 ist ersichtlich, dass die beiden Liniendiagramme nebeneinander positioniert sind. Das führt dazu, dass sie weniger Platz zur Verfügung haben und die Informationen schlechter entnommen werden. Außerdem sind bei den Statistiken die Werte der Stabilität und des Durchschnittsverbrauchs entfernt worden, weil die meisten Prozesse so kurz laufen, dass diese Statistiken keine große Aussagekraft haben.

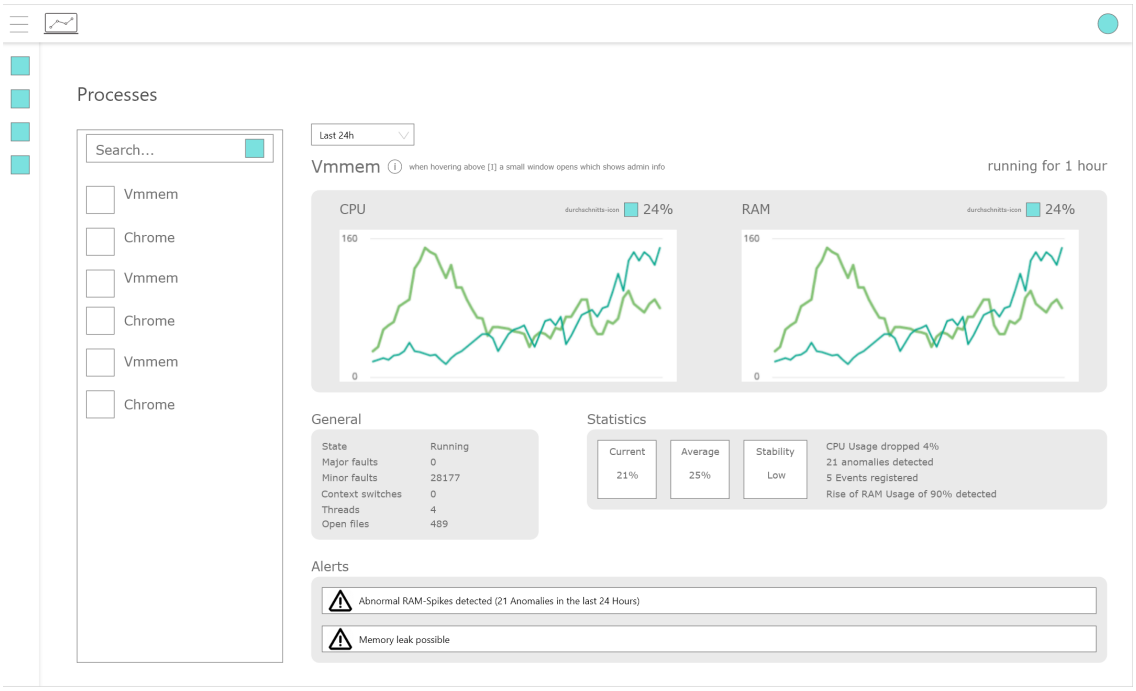


Abbildung 3.11: 1. Version des Entwurfs der Prozess-Ansicht

3.2.4 Netzwerkschnittstellen und Verbindungen

Die in der Abbildung 3.12 dargestellt Ansicht werden alle virtuellen und physischen Netzwerkschnittstellen, die ein Rechner besitzt, angezeigt. Alle aktuellen Netzwerk-Verbindungen zu dem ausgewählten Rechner werden ebenfalls detailliert aufgelistet.

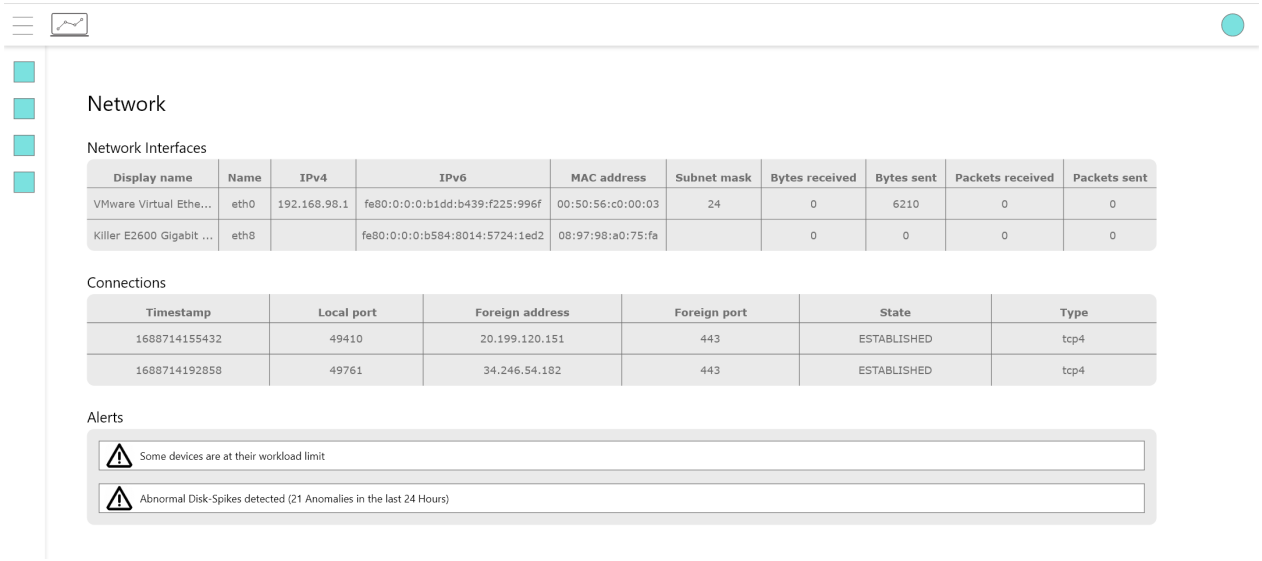


Abbildung 3.12: Finaler Entwurf der Netzwerk-Ansicht

3.2.5 Benutzerdefinierte Benachrichtigungen

In der Abbildung 3.13 ist der Entwurf für benutzerdefinierte Warnungen zu sehen. In dieser Ansicht gibt es die Möglichkeit, benutzerdefinierte Benachrichtigungen anzulegen und zu bearbeiten. Der Benutzer ist in der Lage, folgende Eigenschaften dafür zu definieren:

- **Titel**

  Durch den Titel der Warnung kann der Benutzer diese wieder finden und bearbeiten.

- **Nachricht**

  Die Nachricht wird beim Auftreten der Benachrichtigung in der jeweiligen Ansicht der Webanwendung angezeigt.

- **Wichtigkeitsgrad**

  Der Benutzer hat die Möglichkeit seine Warnungen zu priorisieren. Der Wichtigkeitsgrad 1 wird mit der höchsten Priorität angezeigt und der Wichtigkeitsgrad 5 ist der geringstmögliche Grad.

- **Ressource**

  Es kann ausgewählt werden, welche Ressource die Benachrichtigung überwacht. Zur Auswahl stehen CPU, RAM und verschiedene Eigenschaften der Festplatte.

- **Anwendung**

  Hier kann definiert werden, welche Anwendung der Benachrichtigung überwacht. Der gesamte Rechner wird überwacht, wenn keine Anwendung ausgewählt wird.

- **Grenzwert**

  Der Grenzwert kann als prozentueller und auch als absoluter Wert angegeben werden. Wenn beide Werte definiert werden, greift der Grenzwert, der als erster überschritten wird.

- **Vergleichsoperator**

  Durch den Vergleichsoperator kann festgelegt werden, ob die Benachrichtigung auftritt, wenn der gemessene Wert unter, gleich oder über dem Grenzwert liegt.

- **Erkennungsmethode**

  Der Benutzer kann zwischen ”Moving Averagesünd ... auswählen, nach welcher Methode die Benachrichtigung erkannt wird.

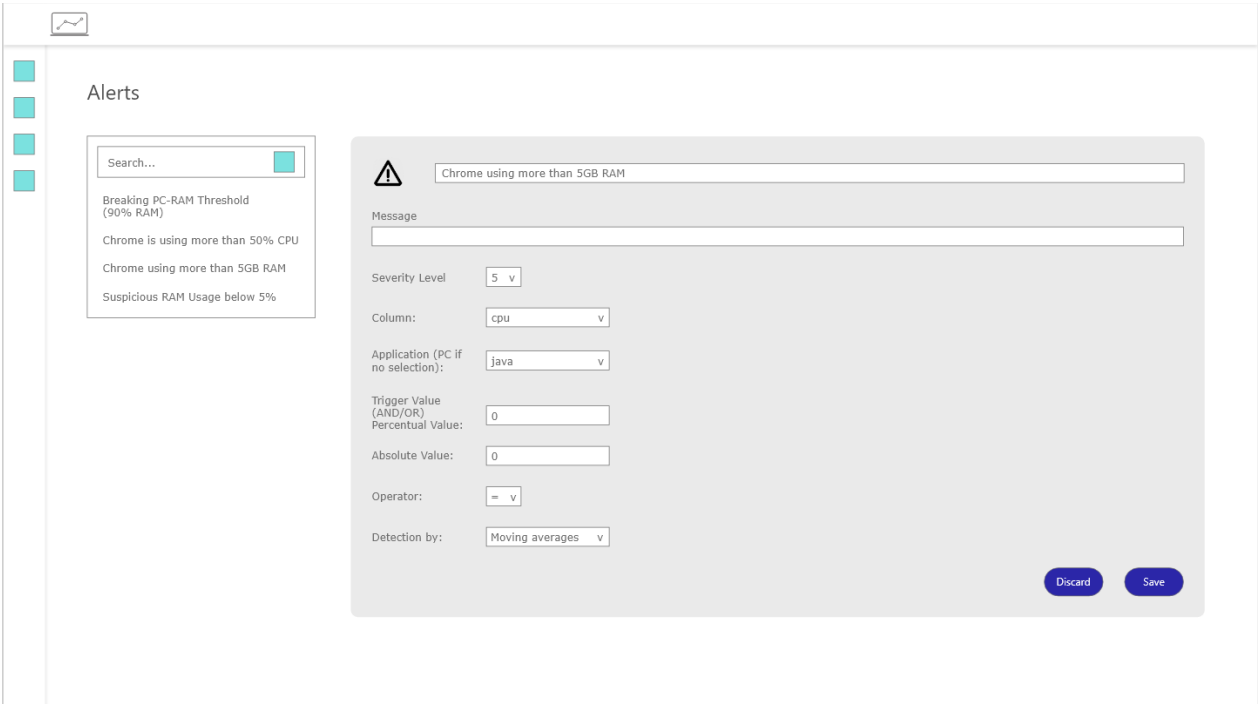


Abbildung 3.13: finaler Entwurf der benutzerdefinierten Benachrichtigungen

3.2.6 Rechner-Auswahl

Bevor die Daten über den Rechner angezeigt werden, muss zuerst ein Rechner ausgewählt werden. Dies ist in der Rechner-Auswahl-Ansicht möglich, welche in der Abbildung 3.14 dargestellt wird. Dafür wird der Name des Rechners und die Hardware UUID einmalig hinzugefügt und kann in weiterer Folge immer wieder selektiert werden. Nach dem Festlegen eines Rechners stehen auf den zuvor angeführten Ansichten die zugehörigen Daten zur Verfügung.

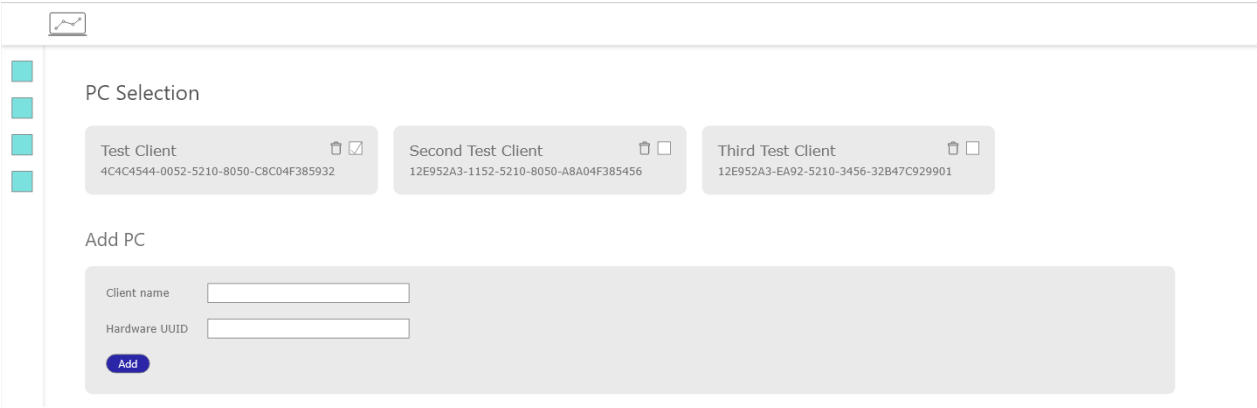


Abbildung 3.14: finaler Entwurf der Rechner-Auswahl

Die 1. Version der Rechner-Auswahl-Ansicht, die in der Abbildung 3.15 zu sehen ist, wurde überarbeitet, weil auf dieser Ansicht keine Detail-Ansicht zu einem Rechner gebraucht wird. Der Fokus der Rechner-Auswahl ist die Möglichkeit Rechner hinzuzufügen, auszuwählen und zu entfernen.

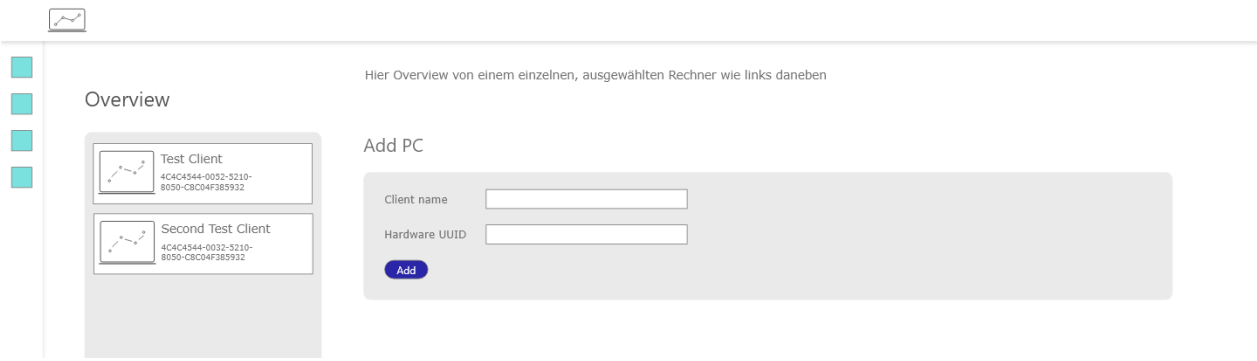


Abbildung 3.15: 1. Version des Entwurfs der Rechner-Auswahl

3.2.7 Gruppenanzeige

Anfänglich ist die Überlegung im Raum gestanden, dass Admins in der Lage sind, mehrere Rechner in Gruppen einzuteilen, um eine große Anzahl an Rechnern effizient zu verwalten. Alle Ansicht würden bei dieser Version minimal angepasst werden. Als Beispiel werden hier 3 Ansichten für Gruppen angeführt. In jeder Ansicht befindet sich auf der linken Seite eine Auswahlmöglichkeit für Gruppen beziehungsweise deren zugeordneten Rechnern. Wie es in der Abbildung 3.16 ersichtlich ist, werden in der Detail-Ansicht einer Gruppe Informationen zu den Rechnern und deren Status angeführt. Zusätzlich werden Warnungen von allen Rechnern der Gruppe hier aufgelistet.

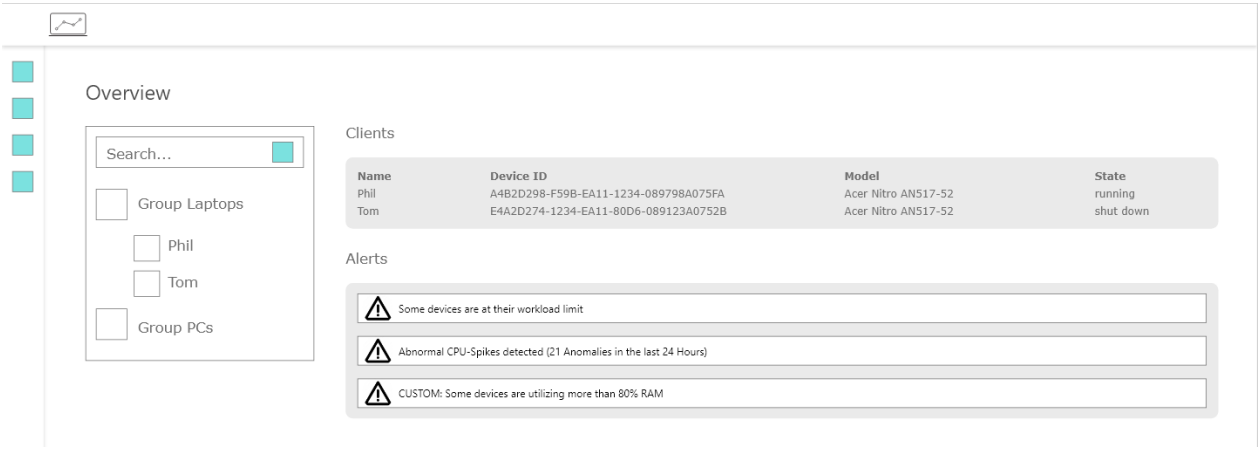


Abbildung 3.16: Entwurf der Detail-Ansicht einer Gruppe

In der Ressourcen-Ansicht gibt es, wie zuvor beschrieben, links die Auswahlmöglichkeit für Gruppen, Rechners und einzelnen Ressourcen wie RAM, CPU, Festplatte und Netzwerk. Die Abbildung 3.17 zeigt ein konkretes Beispiel für die Ansicht, die alle Informationen der CPU des ausgewählten Rechners darstellt.

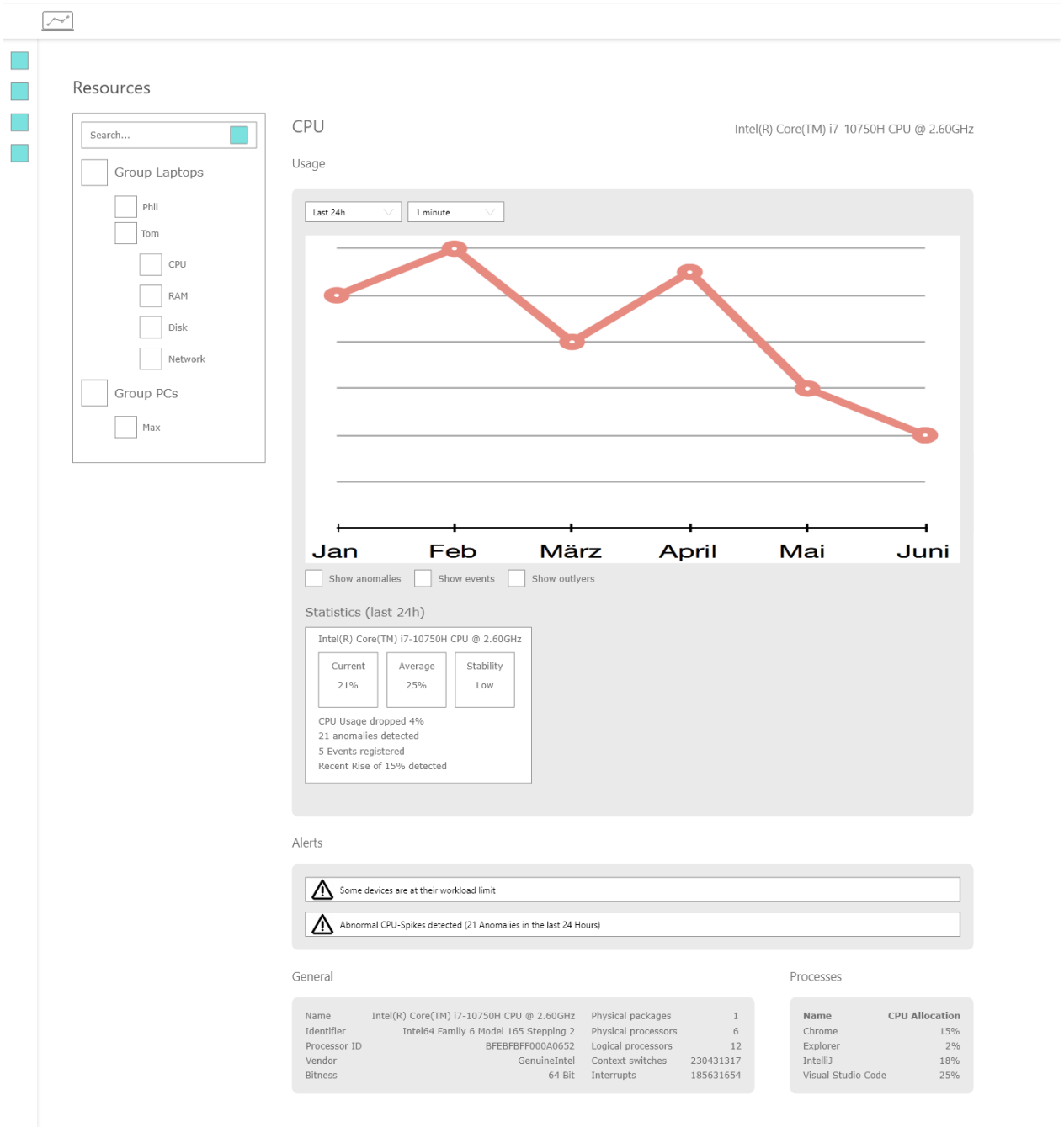


Abbildung 3.17: Entwurf der Anzeige des Ressourcenverbrauchs für die Gruppenanzeige

Natürlich besteht auch die Möglichkeit, wie in Abbildung 3.18 zu sehen, Informationen über die laufenden Prozesse eines Rechners einzuholen. Durch die Gruppenansicht werden die Prozesse nur weiter verschachtelt in der Liste angezeigt.

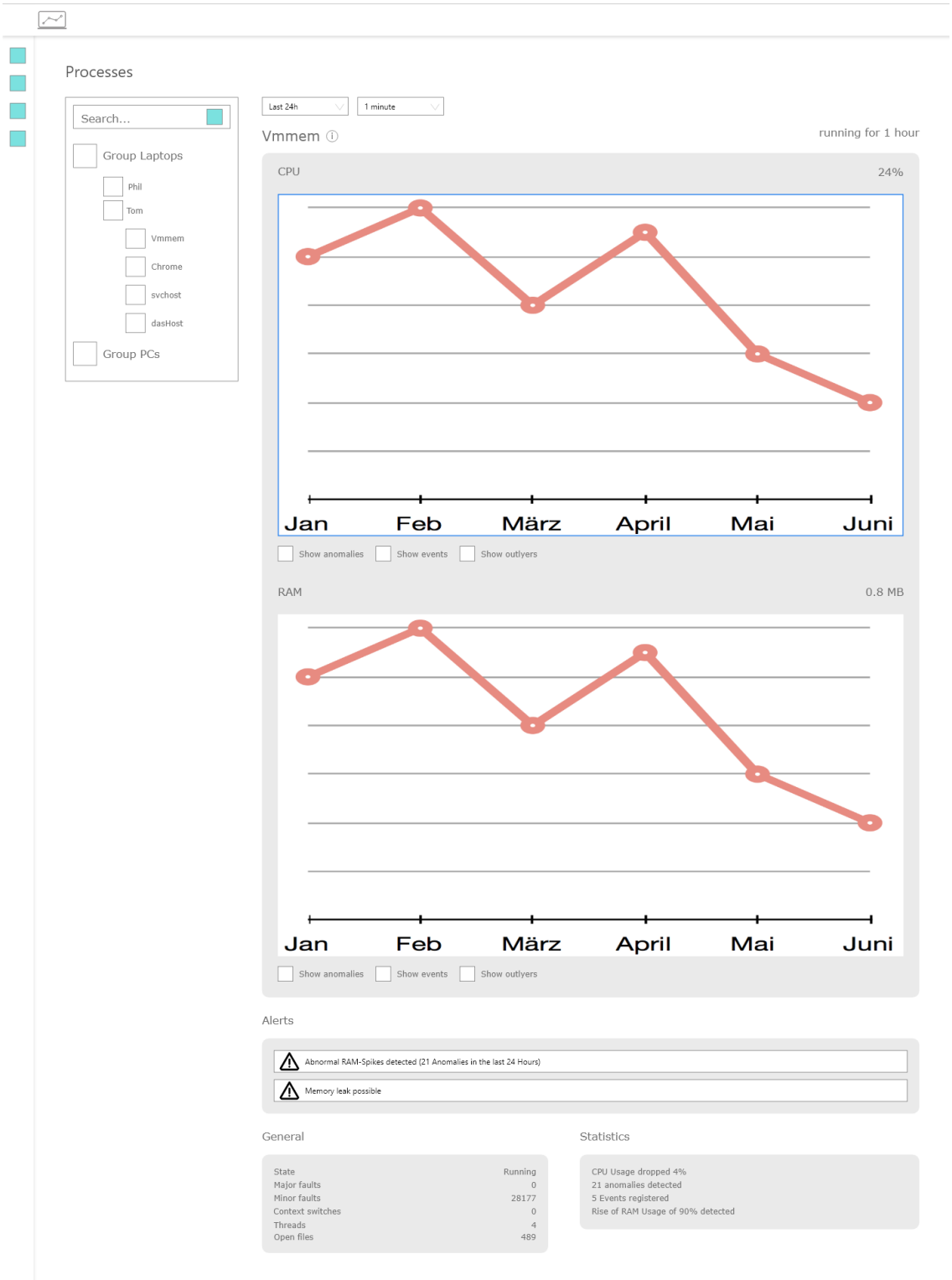


Abbildung 3.18: Entwurf der Anzeige der einzelnen Prozesse für die Gruppenanzeige

3.3 Architektur

Die Programmarchitektur im Rahmen dieser Diplomarbeit besteht aus einem Client-Rechner, überwacht durch den Agenten, einer Webapplikation zur Anzeige der Daten und einem Server, auf dem die Rest-API und die Datenbank laufen. Der Server agiert dabei als zentrale Schnittstelle zwischen allen Client-Rechnern und allen berechtigten Benutzern, die über die Webapplikation auf den Daten der Clients zugreifen. Die verwendete Architektur ist in Abbildung 3.19 dargestellt und wird in diesem Kapital genauer beschrieben.

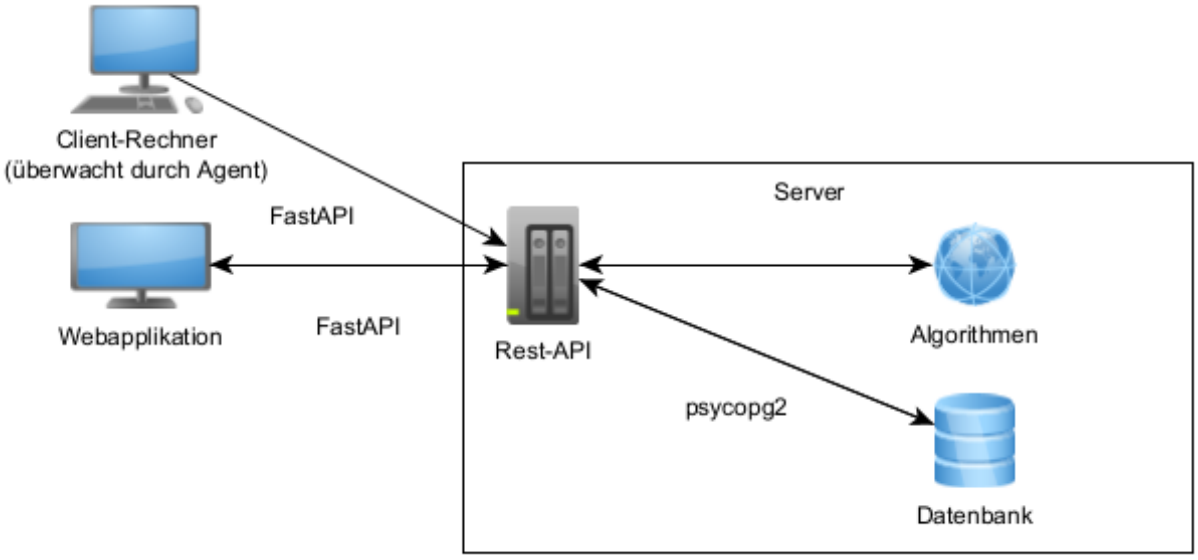


Abbildung 3.19: Programmarchitektur

3.3.1 Agent

Der Agent ist eine lokale Java-Applikation, die auf dem installierten Rechner, bei welchem die Hardware-Metriken gemessen werden, im Hintergrund läuft. Dieser ist dafür zuständig, die Hardware-Allokationsdaten und die Verwendungsdauer von Prozessen mithilfe der OSHI-Bibliothek aus dem Rechner auszulesen, diese in Applikationen zu gruppieren und anschließend im geeigneten CSV-Format zur REST-API zu senden.

3.3.2 REST-API

Die Rest-API wird auf einem Server ausgeführt und ist für die Kommunikation zwischen Webapplikation, Agent und den Algorithmen im Backend verantwortlich. Die verwendete Programmiersprache ist Python aufgrund der Verknüpfungen mit den Algorithmen. Bei dem verwendeten Web-Framework für die Restful-API handelt es sich um FastAPI. Der Zugriff auf die Datenbank erfolgt über den Datenbankadapter psycopg2.

3.3.3 Datenbank

Als Datenbank wird eine Instanz einer TimescaleDB verwendet. Die Datenbank ist verantwortlich für das Speichern der gemessenen Daten aus dem Agent, den angelegten Rechnern und den spezifizierten benutzerdefinierten Benachrichtigung.

3.3.4 Algorithmen

Die Algorithmen zur Datenanalyse und -verarbeitung sind in Python geschrieben und werden zur Erkennung von Anomalien, Change Points, Forecast und der Überprüfung von benutzerdefinierten Benachrichtigungen verwendet. Um diese Zwecke zu erfüllen, erhält die Applikation Referenzen auf die Python-Bibliotheken Pandas, NumPy, Ruptures und Scikit-learn.

3.3.5 Webapplikation

Die Webapplikation repräsentiert in LogSense die Benutzeroberfläche und bildet die Basis für die Interaktion des Benutzers mit der Gesamtapplikation. Bei der Webapplikation bekommt der Benutzer eine Übersicht über die Laufzeit der Applikationen und einen Verlauf inklusive Analysen von ihren CPU und RAM-Auslastungen. Weiters kann der Benutzer den Verlauf des freien Speichers ansehen, mit der Zusatzmöglichkeit, Prognosen über den zukünftigen Verlauf zu bekommen. Auf jeder aufgerufenen Seite erhält der Benutzer relevante Statistiken und eine Liste von aufgetretenen benutzerdefinierten Benachrichtigungen. Auf der benutzerdefinierten Benachrichtigungsseite kann dieser neue Benachrichtigungen hinzufügen und bestehende Benachrichtigungen löschen oder bearbeiten.

3.4 Informationsfluss

Beim Starten des Agenten werden die allgemeinen Daten über den Rechner mittels eines HTTP Clients an den REST Endpoint des Servers gesendet, um den Beginn einer Datenerfassung zu kennzeichnen. Wenn dieser Schritt erfolgreich durchgeführt worden ist, werden alle 60 Sekunden die Daten über den Ressourcenverbrauch seit der letzten Übertragung zusammengefasst, indem die durchschnittlichen Werte berechnet werden. Die Durchschnittswerte werden dann an den Server zur weiteren Verarbeitung gesendet. Alle bereits vollständigen Daten werden vom Server ohne Änderung in der Datenbank persistiert. Die restlichen Daten werden vor der Speicherung noch transformiert, um alle logisch zusammenhängenden Daten gemeinsam zu persistieren. Der Server stellt verschiedene REST Endpoints zur Verfügung, welche die Daten aus der Datenbank abfragen, analysieren und auswerten. Diese REST Endpoints werden vom Frontend aufgerufen, um Informationen über den Rechner und dessen Ressourcenverbrauch zu erhalten und in weiterer Folge darzustellen.

4 Implementierung (Programmierung und QS/Test)

4.1 Datenerfassung

Die Datenerfassung wird über den Agent durchgeführt, dabei handelt es sich um eine Java-Anwendung, die unter anderem die Oshi Bibliothek einbindet, um die Daten aus dem Betriebssystem auszulesen. In weiterer Folge werden die erfassten Daten zusammengefasst und mithilfe eines HTTP Clients an die RESTful API des Servers zur weiteren Verarbeitung gesendet. In den nachfolgenden Abschnitten wird die konkrete Implementierung der Agent-Anwendung beschrieben.

4.1.1 Allgemeine Rechner-Daten

Grundsätzlich wird die Datenerfassung von der Agent-Klasse dieser Java-Anwendung gesteuert. Beim Starten des Agents durch den Benutzer wird die *monitor()*-Methode dieser Klasse aufgerufen. Darin wird überprüft, ob es sich bei dem verwendeten Betriebssystem um Windows handelt. Wenn dies der Fall ist, werden die allgemeinen Rechner-Daten erfasst, die sich nie oder nur ganz selten ändern. Wie zum Beispiel die Hardware UUID, den Hersteller des Rechners, Informationen über die eingebaute CPU und den Hauptspeicher, sowie alle verbundenen Speichermedien inklusive deren Partitionen.
Diese Daten werden in einem SessionComputerData Objekt gespeichert und mithilfe des REST Clients in weiterer Folge an den Server gesendet. Da die Oshi Biliothek viele verschiedene Klassen mit unterschiedlichen Methoden zur Verfügung stellt, die jeweils einen kleinen Teil der Rechner-Daten auslesen, muss der Agent dafür sorgen, dass alle relevanten Daten in das SessionComputerData Objekt gespeichert werden.

Erfassung und Senden der Rechner-Daten

Die Implementierung für das Erfassen und Senden der Rechner-Daten befindet sich in der *monitorSessionComputerData()*-Methode, die im Listing 4.1 angeführt ist. Zuerst werden alle benötigten Daten über den Rechner, die Speichermedien und deren Partitionen erfasst und in Client, DiskStore und Partition Objekte gespeichert. Der Ablauf und die Funktionsweise davon werden im Abschnitt 4.1.1 ausführlich erklärt. Wenn die Methode das erste

Mal aufgerufen wird, das heißt, dass das sessionComputerData Attribut in dieser Klasse noch nicht initialisiert worden ist, wird das SessionComputerData Objekt mit den zuvor erstellten Objekten befüllt. Danach wird dieses Objekt dem REST Client übergeben, der es mit einem HTTP Request an den Server übermittelt. Wenn das erfolgreich erledigt worden ist, wird die stateId, also die ID des aktuellen Zustands des Rechners zurückgegeben. Diese wird bei der Erfassung des Ressourcenverbrauchs verwendet, um den gesendeten Daten einen Rechner zuzuordnen. Falls dabei ein Fehler aufgetreten ist, wird die *handleFailedSessionComputerDataPostRequest()*-Methode aufgerufen, die eine gewisse Zeit wartet, bis die Datenübermittlung nochmals versucht wird.
Da die *monitorSessionComputerData()*-Methode alle 10 Sekunden aufgerufen wird, um Änderungen der Rechner-Daten zu erkennen, wird zusätzlich überprüft, ob die erfassten Daten von den bereits gesendeten Daten abweichen. Gegebenenfalls wird das neue SessionComputerData Objekt an den Server gesendet und eine neue stateId wird zurückgegeben.

```
1  private void monitorSessionComputerData() {
2      Client client = getClientData();
3      List<DiskStore> diskStores = getDiskStores();
4      List<Partition> partitions = getPartitions();
5
6      if (this.sessionComputerData == null || hasClientDataChanged(client) ||
       haveDiskStoresChanged(diskStores) || havePartitionsChanged(partitions)) {
7          SessionComputerData sessionComputerData = new SessionComputerData();
8          sessionComputerData.setClient(client);
9          sessionComputerData.setDiskStores(diskStores);
10         sessionComputerData.setPartitions(partitions);
11         int stateId =
12             this.apiClient.postSessionComputerData(sessionComputerData);
13
14         if (stateId > 0) {
15             this.stateId = stateId;
16         } else {
17             handleFailedSessionComputerDataPostRequest();
18         }
```

```
19          this.sessionComputerData = sessionComputerData;
20      }
21  }
```

Listing 4.1: Rechner-Daten, Speichermedien und Partitionen erfassen und senden

Verwendung der Methoden der Oshi Bibliothek

Der Agent verwendet die Monitor-Klasse, welche Methoden zum Aufrufen der Funktionen der Oshi Bibliothek zur Verfügung stellt. Außerdem werden die Ergebnisse dieser Aufrufe strukturiert in den benötigten Objekten abgespeichert. Um die Funktionen der Oshi-Bibliothek zu nutzen, muss zuerst die oshi-core Abhängigkeit in das Projekt mit einem Package Manager eingebunden werden. Wird zum Beispiel Maven verwendet, könnte die Abhängigkeit wie im Listing 4.2 aussehen.

```
1  <dependency>
2      <groupId>com.github.oshi</groupId>
3      <artifactId>oshi-core</artifactId>
4      <version>6.4.4</version>
5  </dependency>
```

Listing 4.2: oshi-core Maven Abhängigkeit

Somit kann nun ein Objekt der SystemInfo Klasse, die in der Oshi Bibliothek enthalten ist, erstellt werden. Diese Klasse stellt die *Methoden getOperatingSystem()* und *getHardware()* zur Verfügung. Im Listing 4.3 wird der Konstruktor der Monitor Klasse angeführt, in dem diese beiden Methoden der SystemInfo Klasse aufgerufen werden, um die Betriebssystem- und Hardware-Komponenten der Oshi Bibliothek zu erhalten.

```
1  public Monitor() {
2      SystemInfo systemInfo = new SystemInfo();
3      this.operatingSystem = systemInfo.getOperatingSystem();
4      this.hardware = systemInfo.getHardware();
5  }
```

Listing 4.3: Auslesen der Betriebssystem- und Hardware-Komponenten

Erfassung der Rechner-Daten, Speichermedien und Partitionen

Der Agent ruft in der *getClient()*-Methode die *monitorClientData()*-Methode der Monitor Klasse auf. Ein Ausschnitt daraus ist im Listing 4.4 dargestellt. Zu Beginn wird ein neues Client-Objekt erstellt und der aktuellen Zeitpunkt wird darin als Zeitpunkt der Erfassung gespeichert. Mithilfe der Hardware-Komponente der Oshi Bibliothek werden dann durch die Methoden *getComputerSystem()*, *getMemory()* und *getProcessor()* sämtliche Daten über den Rechner, Hautspeicher und CPU erfasst. Von diesen Funktionen wird jeweils ein Objekt der Oshi Bibliothek zurückgegeben wird, das für den Agent irrelevante Informationen enthält. Um zu vermeiden, dass überflüssige Daten in das Client-Objekt gespeichert werden, werden die *getComputerData(...)*, *getMemoryData(...)* und *getProcessorData(...)* Methoden aufgerufen. Sie sortieren irrelevante Informationen aus und geben das für die Speicherung benötigte Objekt zurück.

```
1  Client client = new Client();
2  client.setTimestamp(Instant.now().toEpochMilli());
3
4  ComputerSystem computerSystem = this.hardware.getComputerSystem();
5  client.setComputer(getComputerData(computerSystem));
6
7  GlobalMemory globalMemory = this.hardware.getMemory();
8  client.setMemory(getMemoryData(globalMemory));
9
10 CentralProcessor centralProcessor = this.hardware.getProcessor();
11 ProcessorIdentifier processorIdentifier = centralProcessor.
       getProcessorIdentifier();
12 client.setProcessor(getProcessorData(centralProcessor, processorIdentifier));
```

Listing 4.4: Auslesen der Informationen über Rechner, Hautpspeicher und CPU

Auf ähnliche Art und Weise werden die aktuell eingebauten oder verbundenen Speichermedien erfasst. Das Listing 4.5 stellt die *monitorDiskStores()*-Methode dar. Als Erstes wird der Zeitpunkt der Erfassung zwischengespeichert und dann die *getDiskStores()*-Methode, der Hardware-Komponente verwendet, um eine Liste von allen Speichermedien zu bekommen. Auch hier werden nur ein paar Daten über die einzelnen Speichermedien benötigt.

Darum wird die erhaltene Liste durchlaufen und jeweils der Datenerfassungs-Zeitpunkt, die Seriennummer, das Model, der Name und die Größe des Speichermediums in ein DiskStore Objekt gespeichert, das zur diskStores Liste hinzugefügt wird. Die Liste mit allen relevanten Informationen über die Speichermedien wird zum Schluss an den Agent zurückgegeben.

```
public List<DiskStore> monitorDiskStores() {
    long timestamp = Instant.now().toEpochMilli();
    List<HWDiskStore> hwDiskStoreList = this.hardware.getDiskStores();
    List<DiskStore> diskStores = new ArrayList<>();
    for (HWDiskStore diskStore : hwDiskStoreList) {
        DiskStore diskStoreData = new DiskStore();
        diskStoreData.setTimestamp(timestamp);
        diskStoreData.setSerialNumber(diskStore.getSerial());
        diskStoreData.setModel(diskStore.getModel());
        diskStoreData.setName(diskStore.getName());
        diskStoreData.setSize(diskStore.getSize());
        diskStores.add(diskStoreData);
    }
    return diskStores;
}
```

Listing 4.5: Auslesen der Informationen über Speichermedien

Die gleiche Vorgangsweise wird auch bei den Partitionen der Speichermedien verwendet. In Listing 4.6 ist die *monitorPartitions()*-Methode angeführt. Auch hier wird zuerst der Datenerfassungs-Zeitpunkt zwischengespeichert und eine leere Liste für das Speichern der Partitionen erstellt. Da jede Partition einem Speichermedium zugeordnet ist, müssen diese zuerst erfasst werden. Die Liste der Speichermedien wird durchlaufen und für jedes Speichermedium die Partition durch die *getPartitions()*-Funktion ausgelesen. Der Zeitpunkt der Erfassung, der Name des Speichermediums, die Kennung, der Name, der Typ, der Mount Point, die Größe, sowie die Anzahl der gravierenden und kleineren Seitenfehler werden für jede Partition eines Speichermediums in ein Partition-Objekt gespeichert. Das wiederum wird zur vorher erstellten Liste hinzugefügt. Am Schluss wird diese Liste an den Agent zurückgegeben.

```java
public List<Partition> monitorPartitions() {
    long timestamp = Instant.now().toEpochMilli();
    List<Partition> partitions = new ArrayList<>();
    List<HWDiskStore> hwDiskStoreList = this.hardware.getDiskStores();
    for (HWDiskStore diskStore : hwDiskStoreList) {
        List<HWPartition> hwPartitionList = diskStore.getPartitions();
        for (HWPartition partition : hwPartitionList) {
            Partition partitionData = new Partition();
            partitionData.setTimestamp(timestamp);
            partitionData.setDiskStoreName(diskStore.getName());
            partitionData.setIdentification(partition.getIdentification());
            partitionData.setName(partition.getName());
            partitionData.setType(partition.getType());
            partitionData.setMountPoint(partition.getMountPoint());
            partitionData.setSize(partition.getSize());
            partitionData.setMajorFaults(partition.getMajor());
            partitionData.setMinorFaults(partition.getMinor());
            partitions.add(partitionData);
        }
    }
    return partitions;
}
```

Listing 4.6: Auslesen der Informationen über Partionen der Speichermedien

4.1.2 Prozesse und deren Ressourcenverbrauch

Nachdem die allgemeinen Rechner-Daten erfasst und an den Server gesendet worden sind, wird alle 10 Sekunden die *monitorData()*-Methode der Agent Klasse aufgerufen, die in Listing 4.7 dargestellt ist. Bei jedem Aufruf dieser Methode wird das Attribut measuring-Count um 1 erhöht und der Zeitpunkt der Erfassung zwischengespeichert. Der Erfassungs-Zeitpunkt wird der *getAndIngestOSProcesses(...)*-Funktion übergeben, um die aktuell laufenden Prozesse zu erfassen. Verwendet wird dafür die *monitorProcesses()*-Methode, die von der Montitor Klasse zur Verfügung gestellt wird und aus der Betriebssystem-Komponente die Prozesse ausliest. Diese erfassten Prozesse werden durch die StatSer-

vice Klasse weiterverarbeitet und für die Auswertung und Durchschnittsberechnung zwischengespeichert. Dieser Vorgang wird in Abschnitt 4.1.3 genauer beschrieben. Wenn es sich um die 6. Datenerfassung in dieser Minute handelt, wird die besagte Durchschnittsberechnung durch die StatService Klasse aufgerufen. Zusätzlich wird der gesamte Ressourcenverbrauch des Rechners, die Netzwerk-Schnittstellen und die IP-Verbindungen mithilfe der *monitorRunningData(...)*-Methode erfasst. Diese wird in Abschnitt 4.1.4 ausführlich behandelt. Abschließend werdend die allgemeinen Rechner-Daten nochmals erfasst, um zu überprüfen, ob sich die Daten verändert haben und werden gegebenenfalls erneut an den Server gesendet.

```
1  private void monitorData() {
2      this.measuringCount++;
3      long timestamp = Instant.now().toEpochMilli();
4      getAndIngestOSProcesses(timestamp);
5      if (this.measuringCount % 6 == 0) {
6          List<Application> analysedApplications = this.statService.
       getAverageOfApplicationMeasurements(timestamp);
7          monitorRunningData(analysedApplications, timestamp);
8          monitorSessionComputerData();
9      }
10 }
```

Listing 4.7: Auslesen der Informationen über Partionen der Speichermedien

4.1.3 Zusammenführen der gemessenen Daten

Nachdem einmal alle 10 Sekunden die Informationen über die aktuell laufenden Prozesse gemessen worden sind, werden diese an eine Instanz der StatService Klasse zur Weiterverarbeitung übergeben. Dabei werden folgende Schritte durchgeführt:

Herausfiltern der System-Prozesse

Da sämtliche laufende Prozesse erfasst worden sind, sind auch jene Prozesse enthalten, die vom Betriebssystem verwendet und gesteuert werden. Diese sind für den Endbenutzer nicht relevant und werden daher herausgefiltert.

Woran wird ein System-Prozess erkannt? Durch eine Analyse der Daten konnte festgestellt werden, dass es in der Regel bei System-Prozessen entweder keinen Kommandozeilen-Befehl gibt, mit dem der Prozess gestartet worden ist. Der Grund dafür ist, dass einige dieser Prozesse nicht vom Benutzer des Rechners oder von anderen Prozessen aufgerufen werden kann. Jedoch werden manche dieser System-Prozesse durch den Benutzer des Rechners oder eine installierte Anwendung gestartet. Diese Prozesse haben dann einen Kommandozeilen-Befehl, der entweder mit 'C:\Windows' oder mit 'C:\Windows\system32' beginnt.
Im Listing 4.8 ist die Methode *filterSystemProcesses(...)* angeführt, die für das Herausfiltern der System-Prozesse verantwortlich ist. Es werden die erfassten Prozesse durchlaufen und überprüft, ob der Kommandozeilen-Befehl den oben genannten Kriterien eines System-Prozesses entspricht. Wenn dies nicht der Fall ist, wird der Prozess in eine neue Liste mit allen relevanten Prozessen gespeichert und zum Schluss zurückgegeben.

```
1  private List<OSProcess> filterSystemProcesses(List<OSProcess> osProcesses) {
2      List<OSProcess> filteredOsProcesses = new ArrayList<>();
3      for (OSProcess process : osProcesses) {
4          if (process.getCommandLine() != null
5                  && !process.getCommandLine().equals("")
6                  && !process.getCommandLine().contains("C:\\Windows")
7                  && !process.getCommandLine().contains("C:\\Windows\\system32")
       ) {
8              filteredOsProcesses.add(process);
9          }
10     }
11     return filteredOsProcesses;
12 }
```

Listing 4.8: Herausfiltern der System-Prozesse

Prozesse zu Anwendungen zusammenfassen

Manche Anwendungen, die auf einem Rechner laufen, teilen ihre verschiedenen Aufgaben und Komponenten auf verschiedene Prozesse auf. Zum Beispiel gibt es für jeden Tab

der Anwendung ”Google Chromeëinen Prozess, der sich nur um die Web-Suchen in dem einen Tab kümmert. Daher muss beachtet werden, dass nicht der Ressourcenverbrauch der ganzen Anwendung erfasst wird, sondern nur der Ressourcenverbrauch pro Prozess. Benötigt werden jedoch die Daten über pro Anwendung. Somit müssen die einzelnen Prozesse mit dem gleichen Namen zu einer Anwendung zusammengefasst werden.
Dies wird mithilfe der Methode *mergeProcessesIntoApplications(...)* durchgeführt, die im Listing 4.9 dargestellt wird. Die Liste der erfassten Prozesse wird durchlaufen und es wird überprüft, ob es in der oben initialisierten TreeMap bereits einen Key mit dem Namen des jeweiligen Prozesses gibt. Wenn dies der Fall ist, wird die Anwendung, zu dem der Prozess gehört, aus der TreeMap geholt. Ansonsten wird eine neue Anwendung mit dem Namen des Prozesses und weiteren Informationen, die für die ganze Anwendung gelten, erstellt und zu der TreeMap hinzugefügt. Dazu zählen der Zeitpunkt der Erfassung, die Bitversion, der Kommandozeilen-Befehl, das aktuelle Arbeitsverzeichnis, der Name der Anwendung, der Speicherort, der Status und der Benutzer, der die Anwendung gestartet hat.
In beiden Fällen wird danach der jeweilige Prozess inklusive dessen CPU Auslastung in die containedProcesses Map der Anwendung hinzugefügt. Zusätzlich werden die Anzahl der Kontextwechsel, der schwerwiegenden Seitenfehler, der geöffneten Dateien, der verwendeten Threads, sowie der verbrauchte Hauptspeicher und die Laufzeit des Prozesses zu den bereits vorhandenen Daten der Anwendung addiert. Nachdem dieser Vorgang für alle Prozesse wiederholt worden ist, wird die Map mit den Anwendungen zurückgegeben.

```
1  private Map<String, Application> mergeProcessesIntoApplications(long timestamp
       , List<OSProcess> processList) {
2      Map<String, Application> mergedApplications = new TreeMap<>();
3      for (OSProcess process : processList) {
4          String applicationName = process.getName();
5          Application application;
6          if (mergedApplications.containsKey(applicationName)) {
7              application = mergedApplications.get(applicationName);
8          } else {
9              application = createNewApplication(application, process);
10         }
```

```
11          addProcessAndCpuUsageToApplication(application, process);
12          application.mergeData(process.getContextSwitches(), process.
    getMajorFaults(),
13                  process.getOpenFiles(), process.getResidentSetSize(),
14                  process.getThreadCount(), process.getUpTime());
15          mergedApplications.put(applicationName, application);
16
17      }
18      return mergedApplications;
19  }
```

Listing 4.9: Zusammenfassen von Prozessen zu Anwendungen

Liste der Anwendungen für Durchschnittsberechnung zwischenspeichern

Aus der zuvor erhaltenen Map mit den Anwendungen werden nacheinander die Anwendungen in eine weitere Map eingefügt, die als Key den Namen der Applikation und als Value eine Liste der Klasse Application speichert. Dadurch werden die verschiedenen Messzeitpunkte einer Anwendung einfach abgebildet.
Im Listing 4.10 wird die Methode *insertApplicationDataIntoApplicationMeasurements(...)* angeführt, die die Daten der aktuellen Messung in die Map mit den Messzeitpunkten eingefügt. Dafür wird zuerst die Map mit den erfassten Daten durchlaufen und für jede Anwendung wird überprüft, ob es in der applicationMeasurements Map, die ein Attribut der StatService Klasse ist, bereits ein Element mit dem Namen der Anwendung als Key gibt. Ist dies der Fall, wird die Liste mit den Messzeitpunkten zu dieser Anwendung aus der applicationMeasurementsMap geholt. Ansonsten wird eine leere ArrayList erzeugt.
Zu dieser Liste wird dann das Application Objekt mit den Daten des aktuellen Messzeitpunkts hinzugefügt. Abschließend wird die Liste mit dem neuen Messzeitpunkt entweder neu in die applicationMeasurements Map eingefügt oder die vorherige Liste wird durch die neue Liste ausgetauscht.

```
private void insertApplicationDataIntoApplicationMeasurements
        (Map<String, Application> mergedApplications) {
    for (Map.Entry<String, Application> applicationMeasurementsEntry :
   mergedApplications.entrySet()) {
        List<Application> applicationList;
        String applicationName = applicationMeasurementsEntry.getKey();
        if (this.applicationMeasurements.containsKey(applicationName)) {
            applicationList = this.applicationMeasurements.get(applicationName
   );
        } else {
            applicationList = new ArrayList<>();
        }
        applicationList.add(applicationMeasurementsEntry.getValue());
        this.applicationMeasurements.put(applicationName, applicationList);
    }
}
```

Listing 4.10: Aktuelle Messdaten für Durchschnittsberechnung speichern

Durchschnittsberechnung

Nachdem die 3 oben beschriebenen Schritte 6 Mal durchgeführt worden sind, ruft der Agent die *analyseApplicationMeasurements(...)*-Methode auf, um die Durchschnittsberechnung durchzuführen. Diese ruft wiederum die Methode *evaluateApplicationMeasurements(...)* mit dem Zeitpunkt der letzten Messung auf. Darin wird für jede Anwendung die Daten der einzelnen Messzeitpunkte zusammengefasst.

Im Listing 4.11 wird dargestellt, welche Schritte vorgenommen werden müssen, um alle Daten in ein Application Objekt zusammenzufassen. Dabei wird die applicationMeasurements Map durchlaufen und für jede darin gespeicherte Anwendung die Liste mit den erfassten Daten geholt. Diese Liste wird der *performStatisticalAnalysis(...)*-Methode übergeben und ein Application Objekt zurückgibt. Es enthält die durchschnittliche CPU Auslastung und die addierte Anzahl der Kontextwechsel, der schwerwiegenden Seitenfehler, der geöffneten Dateien, der verwendeten Threads, sowie den verbrauchten Hauptspeicher und die Laufzeit der Anwendung.

Das averageApplication Objekt wird weiters verwendet, um Informationen, die für sämtli-

che Prozesse der Anwendung zu schreiben. Dafür wird der setInvariableInformation Methode das averageApplication Objekt und ein beliebiges Element der Anwendungs-Liste übergeben. Hier wird das erste Element in der Liste verwendet, um die Daten über die Anwendung zu erhalten. Weiters wird die Anzahl der Prozesse, aus denen eine Anwendung besteht, vom ersten Messzeitpunkt mit dem letzten Messzeitpunkt verglichen und die Differenz im Attribut processCountDifference des averageApplication Objektes gespeichert. Zum Schluss wird die Methode *calculateAverage(...)* der Application Klasse mit der Anzahl an Messzeitpunkten aufgerufen, um die durchschnittliche Anzahl der Kontextwechsel, der schwerwiegenden Seitenfehler, der geöffneten Dateien und der verwendeten Threads, sowie die durchschnittliche CPU Auslastung zu berechnen und im averageApplication Objekt zu speichern. Nachdem sämtliche Attribute dieses Objekts gesetzt worden sind, wird es zu der evaluatedApplicationData Liste, die alle bereits ausgewerteten Anwendungen enthält, hinzugefügt.

```
1  private List<Application> evaluateApplicationMeasurements(long timestamp) {
2      List<Application> evaluatedApplicationData = new ArrayList<>();
3      for (Map.Entry<String, List<Application>> applicationMeasurementEntry
4              : this.applicationMeasurements.entrySet()) {
5          List<Application> applicationList = applicationMeasurementEntry.
       getValue();
6          Application averageApplication = performStatisticalAnalysis(
       applicationList,
7              timestamp);
8          setInvariableInformation(averageApplication, applicationList.get(0));
9          averageApplication.setProcessCountDifference(
10             compareProcessesAmount(applicationList));
11         averageApplication.calculateAverage(applicationList.size());
12         evaluatedApplicationData.add(averageApplication);
13     }
14     return evaluatedApplicationData;
15 }
```

Listing 4.11: Messzeitpunkte zusammenführen

4.1.4 Ressourcenverbrauch vom Rechner

Neben der *monitorSessionComputerData()*-Methode ist in der Agent-Klasse auch die *monitorRunningData(...)*-Methode implementiert, die alle 60 Sekunden von der *monitorData()*-Methode aufgerufen wird. Da die Daten über den Ressourcenverbrauch des Rechners, genauso wie die Netzwerk-Schnittstellen und die IP-Verbindungen auf unterschiedliche Komponenten der Oshi Bibliothek aufgeteilt ist, wird ein RunningData Objekt zur Speicherung aller relevanten Daten verwendet. Es enthält alle benötigten Informationen und kann somit an den REST Client für die Übertragung an den Server weitergegeben werden.
Im Listing 4.12 ist zu erkennen, dass die bereits ausgewerteten Prozesse und der Zeitpunkt der letzten Erfassung der Prozesse der *monitorRunningData(...)*-Methode übergeben werden. Weiters werden die Methoden *getResources()*, *getNetworkInterfaces()* und *getIPConnections()* verwendet, um die zugehörigen Funktionen der Monitor Klasse aufzurufen und die Ergebnisse in Resource-, NetworkInterface- und Connection-Objekte zu speichern. Der in Abschnitt 4.1.5 beschriebene CSV Konverter wird dann verwendet, um diese Objekte als CSV String in ein RunningData Objekt zu speichern. Das wird wiederum mit der ID des aktuellen Zustands des Rechners an den REST Client übergeben, damit die Daten zur Analyse an den Server geschickt werden.

```
1  private void monitorRunningData(List<Application> analysedApplications,
2          long timestamp) {
3      Resources resources = getResources();
4      List<NetworkInterface> networkInterfaces = getNetworkInterfaces();
5      List<Connection> connections = getIpConnections();
6
7      RunningData runningData = new RunningData();
8      runningData.setApplication_data(this.csvDataConverter
9          .convertApplicationData(timestamp, analysedApplications));
10     runningData.setPc_resources(this.csvDataConverter
11         .convertResourceData(timestamp, resources));
12     runningData.setNetwork_interface(this.csvDataConverter
13         .convertNetworkInterfacesData(timestamp, networkInterfaces));
14     runningData.setConnection_data(this.csvDataConverter
```

```
15          .convertConnectionData(timestamp, connections));
16      this.apiClient.postRunningData(runningData, this.stateId);
17  }
```

Listing 4.12: Erfassen der Informationen über Ressourcenverbrauch, Prozesse, Netzwerk-Schnittstellen und IP-Verbindungen

Erfassung des Ressourcenverbrauchs und Rechner-Daten

Der Agent ruft in der *getResources()*-Methode die *monitorResources()*-Methode der Monitor Klasse auf. Diese Methode liest folgende Werte aus den Betriebssystem- und Hardware-Komponenten der Oshi Bibliothek aus:

- Betriebssystem-Komponente
  - gesamter freier Speicherplatz von allen Speichermedien
- Hardware-Komponente
  - Anzahl der gelesenen Bytes von allen Speichermedien
  - Anzahl der lesenden Zugriffe auf alle Speichermedien
  - Anzahl der geschriebenen Bytes auf allen Speichermedien
  - Anzahl der schreibenden Zugriffe auf alle Speichermedien
  - Anzahl der gravierenden Seitenfehler von allen Partitionen
  - Anzahl der geringen Seitenfehler von allen Partitionen
  - freier Speicherplatz im Hauptspeicher
  - Namen der Stromversorgungen des Rechners
  - Ladezustand der Stromversorgungen des Rechners
  - Entladezustand der Stromversorgungen des Rechners
  - Verbindungszustand der Stromversorgungen des Rechners
  - aktuelle Ladekapazität der Stromversorgungen des Rechners
  - Anzahl der Wechsel von Aufgaben durch den Prozessor
  - Anzahl der Unterbrechungen von Aufgaben durch den Prozessor

Ein Ausschnitt aus der *monitorResources()*-Methode wird im Listing 4.13 angeführt. Dafür wird ein neues Resources Objekt initialisiert, in dem die oben angegebenen Daten gespeichert werden. Dazu gehört zum Beispiel der freie Speicherplatz im Hauptspeicher, der mit der *getMemory()*-Methode der Hardware-Komponente von der Oshi Bibliothek, die ein Objekt der Klasse GlobalMemory zurückgibt. Diese stellt wiederum die getAvailable Methode zur Verfügung, mit der der freie Speicherplatz des RAMs ausgelesen wird. Auf ähnliche Art und Weise funktioniert das mit den Prozessor-Daten. Die Methode *getProcessor()* der Hardware-Komponente liefert das CentralProcessor Objekt, das die *getContextSwitches()* und *getInterrupts()* Methoden enthält. Somit kann die Anzahl der Wechsel von Aufgaben und die Anzahl der Unterbrechungen von Aufgaben durch den Prozessor erfasst und in das Resources Objekt gespeichert werden. Alle weiteren Daten werden dann ebenfalls in dieser Methode ausgelesen und zum Schluss wird das fertige Resources Objekt an die monitorRunningData Methode des Agents zurückgegeben.

```
1 Resources resources = new Resources();
2 GlobalMemory memory = this.hardware.getMemory();
3 resources.setAvailableMemory(memory.getAvailable());
4
5 CentralProcessor processor = this.hardware.getProcessor();
6 resources.setProcessorContextSwitches(processor.getContextSwitches());
7 resources.setProcessorInterrupts(processor.getInterrupts());
```

Listing 4.13: Informationen über Ressourcenverbrauch, Prozesse, Netzwerk-Schnittstellen und IP-Verbindungen auslesen

Erfassung der physischen und virtuellen Netzwerk-Schnittstellen

Grundsätzlich funktioniert auch so die Erfassung der physischen und virtuellen Netzwerk-Schnittstellen. Dafür stellt die Monitor-Klasse die *monitorNetworkInterfaces()*-Methode zur Verfügung, welche im Listing 4.14 ersichtlich ist. Zu Beginn wird eine Liste der NetworkInterface Klasse erstellt, die nur alle relevanten Informationen über die Schnittstellen enthält. Dann werden auf die Netzwerk-Schnittstellen des Rechners mithilfe der *getNetworkIFs()*-Methode der Hardware-Komponente zugegriffen und durchlaufen.

Von jeder erfassten Netzwerk-Schnittstelle werden folgende Daten in einem NetworkInterface gespeichert und zur oben erstellten Liste hinzugefügt:

- Anzeige-Name
- Name
- zugeordneten IPv4-Adressen
- zugeordneten IPv6-Adressen
- MAC-Adresse
- Subnetzmasken
- Anzahl der durch die Netzwerk-Schnittstelle empfangenen Bytes
- Anzahl der durch die Netzwerk-Schnittstelle versendeten Bytes
- Anzahl der durch die Netzwerk-Schnittstelle empfangenen Netzwerk-Pakete
- Anzahl der durch die Netzwerk-Schnittstelle versendeten Netzwerk-Pakete

Wobei zu beachten ist, dass eine Netzwerk-Schnittstelle mehrere zugeordnete IPv4- und IPv6-Adressen haben kann. Die Klasse NetworkIF speichert daher ein Array von Strings. Die NetworkInterface Klasse benötigt jedoch eine Liste der INetAddress Klasse. Aufgrund dessen wird vor dem Speichern die Hilfsmethode *convertStringArrayToIpAddressList(...)* aufgerufen, die das String-Array in eine Liste von INetAddress Objekten umwandelt.

```
1 public List<NetworkInterface> monitorNetworkInterfaces() {
2     List<NetworkInterface> networkInterfaces = new ArrayList<>();
3     List<NetworkIF> networkIFList = this.hardware.getNetworkIFs();
4     for (NetworkIF networkIF : this.hardware.getNetworkIFs()) {
5         NetworkInterface networkInterface = new NetworkInterface();
6         networkInterface.setDisplayName(networkIF.getDisplayName());
7         networkInterface.setName(networkIF.getName());
8         networkInterface.setIpv4Addresses(convertStringArrayToIpAddressList(
    networkIF.getIPv4addr()));
```

```
9           networkInterface.setIpv6Addresses(convertStringArrayToIpAddressList(
    networkIF.getIPv6addr()));
10          networkInterface.setMacAddress(networkIF.getMacaddr());
11          networkInterface.setSubnetMasks(networkIF.getSubnetMasks());
12          networkInterface.setBytesReceived(networkIF.getBytesRecv());
13          networkInterface.setBytesSent(networkIF.getBytesSent());
14          networkInterface.setPacketsReceived(networkIF.getPacketsRecv());
15          networkInterface.setPacketsSent(networkIF.getPacketsSent());
16          networkInterfaces.add(networkInterface);
17      }
18      return networkInterfaces;
19  }
```

Listing 4.14: Netzwerk-Schnittstellen auslesen

Erfassung der IP-Verbindungen

Die gleiche Vorgehensweise wird beim Erfassen der IP-Verbindungen angewandt. Im Listing 4.15 ist die *monitorIpConnections()*-Methode angeführt, die ebenfalls eine leere Liste der Connection Klasse initialisiert. Außerdem wird die Betriebssystem-Komponente der Oshi Bibliothek verwendet, um die IP-Statistiken mit den IP-Verbindungen auszulesen. Die Liste wird wieder durchlaufen und alle benötigten Daten werden in ein Connection Objekt mit dem Namen connectionData gespeichert und in die Liste von IP-Verbindungen hinzugefügt. Zu diesen Daten gehören:

- lokale Port, mit dem sich verbunden worden ist
- externe Port, von dem aus sich verbunden worden ist
- Zustand der IP-Verbindung
- Typ der IP-Verbindung
- ID des Prozesses, der für diese IP-Verbindung verantwortlich ist
- lokale IPv4-Adresse, mit der sich verbunden worden ist
- externe IPv4-Adresse, von der aus sich verbunden worden ist

Auch hier werden die lokalen und externen IPv4-Adressen vor der Speicherung mithilfe der *parseIPv4Address(...)*-Methode von einem Byte-Array in ein Objekt der INetAddress-Klasse konvertiert.

```
public List<Connection> monitorIpConnections() {
    List<Connection> connections = new ArrayList<>();
    List<IPConnection> ipConnections = this.operatingSystem.
   getInternetProtocolStats().getConnections();
    for (IPConnection connection : ipConnections) {
        Connection connectionData = new Connection();
        connectionData.setLocalPort(connection.getLocalPort());
        connectionData.setForeignPort(connection.getForeignPort());
        connectionData.setState(connection.getState());
        connectionData.setType(connection.getType());
        connectionData.setOwningProcessID(connection.getowningProcessId());
        connectionData.setLocalAddress(parseIPv4Address(connection.
   getLocalAddress()));
        connectionData.setForeignAddress(parseIPv4Address(connection.
   getForeignAddress()));
        connections.add(connectionData);
    }
    return connections;
}
```

Listing 4.15: IP-Verbindungen auslesen

4.1.5 CSV Konverter

Um die Weiterverarbeitung der Daten zu vereinfachen, ist es notwendig die Daten im CSV-Format an den Server zu senden. Der REST Client benötigt daher eine Möglichkeit die verschiedenen Arten von erfassten Daten in CSV-Dateien zu konvertieren. Dafür ist in der Agent-Anwendung eine Schnittstelle mit dem Namen ”DataConverter”definiert worden, deren Methoden in der Tabelle 4.1 dargestellt werden.

Name	Rück-gabe-wert	Parameter	Beschreibung
convertApplicati-onData	String	long timestamp List<Application> applications	Daten über aktuell laufende Anwendungen in eine Zei-chenfolge konvertieren
convertResource-Data	String	long timestamp Resources resources	Daten über aktuellen Res-sourcenverbrauch in Zeichen-folge konvertieren
convertConnec-tionData	String	long timestamp List<Connection> connectionData	Daten über aktuell erstellte IP-Verbindungen in Zeichenfolge konvertieren
convertNetworkIn-terfacesData	String	long timestamp List<NetworkInterface> networkInterfaces	Daten über aktuell erkann-te physische und virtuelle Netzwerk-Schnittstellen in Zeichenfolge konvertieren
convertClientData	String	Client client	allgemeine Daten über den Client-Rechner in Zeichenfol-ge konvertieren

convertDiskStore-Data	String	List<DiskStore> diskStores	Daten über Speichermedien in Zeichenfolge konvertieren
convertPartiti-onData	String	List<Partition> partiti-ons	Daten über die Partitionen der Speichermedien in Zeichen-folge konvertieren

Tabelle 4.1: Methoden der DataConverter Schnittstelle

Die Verwendung dieser Schnittstelle ermöglicht es dem REST Client die konkrete Implementierung des Konvertierers beliebig auszutauschen. In diesem Fall werden die Daten im CSV-Format benötigt. Das wird durch die Klasse CSVDataConverter umgesetzt, die die DataConverter Schnittstelle mit ihren definierten Methoden implementiert. Ein Beispiel für eine Methode dieser Klasse ist im Listing 4.16 angeführt. Mithilfe der im JDK mitgelieferte Klasse StringBuilder werden Strings einfach beliebig zusammengesetzt und bearbeitet. Das ermöglicht es, die erste Zeile mit den Attributnamen und in weiterer Folge die einzelnen Daten-Zeilen mit der *append(...)*-Methode an den im StringBuilder Objekt gespeicherten String anzuhängen.
Grundsätzlich werden in einer CSV-Datei Kommas oder Strichpunkte als Trennzeichen verwendet. In den erfassten Daten kommen jedoch diese Zeichen des Öfteren vor. Daher ist für die Kommunikation zwischen Client-Rechner und RESTful API des Servers definiert worden, dass stattdessen ein senkrechter Strich verwendet wird.

```
1 @Override
2 public String convertDiskStoreData(List<DiskStore> diskStores) {
3     if (diskStores != null) {
4         StringBuilder csv = new StringBuilder();
5         csv.append("measurement_time|serialNumber|model|name|size\n");
6         for (DiskStore diskStore : diskStores) {
7             csv.append(diskStore.getTimestamp()).append("|");
8             csv.append(diskStore.getSerialNumber()).append("|");
```

```
9               csv.append(diskStore.getModel()).append("|");
10              csv.append(diskStore.getName()).append("|");
11              csv.append(diskStore.getSize()).append("|");
12          }
13          return csv.toString();
14      }
15      return null;
16  }
```

Listing 4.16: Beispiel für eine Methode der CSVDataConverterKlasse

4.1.6 REST Client

Im vorliegenden System fungiert clientseitig ein REST Client als Schnittstelle zwischen dem Agent und dem Server. Im Programmablauf werden zu unterschiedlichen Zeitpunkten 2 verschiedene Arten von Daten an die RESTful API des Servers gesendet.
Nach dem Starten des Agents werden generelle Daten über den Rechner erfasst und an den Server gesendet. Dabei handelt es sich um Informationen, die sich nie oder nur ganz selten ändern. Wie zum Beispiel die Hardware UUID, den Hersteller des Rechners, Informationen über die eingebaute CPU und Speichermedien inklusive deren Partitionen. Diese erste Übertragung markiert den Start einer Session. Alle 10 Sekunden werden diese Daten nochmals erfasst und auf Änderungen überprüft. Falls die Informationen von den vorherigen abweichen, werden sie erneut an den Server übermittelt.
Weiters werden alle 10 Sekunden Informationen über den aktuellen Ressourcenverbrauch des gesamten Rechners, alle momentan laufenden Prozesse und deren Ressourcenverbrauch, die Netzwerkschnittstellen und IP-Verbindungen erfasst. Von diesen Daten werden alle 60 Sekunden die Durchschnittswerte berechnet und an den Server zur Auswertung und Analyse an den Server gesendet.

Um die Weiterverarbeitung der Daten zu vereinfachen, ist es notwendig die Daten als CSV-Dateien an den Server zu senden. Beide Endpunkte der API definieren ”multipart/form-dataäls Inhalt des Requests fest. Dadurch kann eine Liste an Dateien übermittelt werden. Das org.apache.hc.client5.http Package bietet mit der MultipartEntityBuilder Klasse eine Möglichkeit eine HTTP Entity für den Inhalt ”multipart/form-dataßu erstellen. Wie das ge-

nau funktioniert ist im Listing 4.17 anhand des SessionComupterData Objekts ersichtlich. Eine HTTP Entity ist ein Objekt, das den Header und den Body eines Requests zusammenfasst und in weiterer Folge einem HttpPost-Objekt übergeben werden kann. Durch die statische Methode *create()* in der MultipartEntityBuilder-Klasse kann ganz einfach eine Instanz dieser Klasse erzeugt werden. Mithilfe der *addBinaryBody(...)*-Methode kann man die CSV-Dateien der HTTP Entity hinzufügen. Dafür muss der Name der HTTP Entity, die Daten als byte-Array, den Art des Inhalts und den Dateinamen angeben. Die HTTP Post Klasse enthält die Methode *setEntity(...)*, mit der die HTTP Entity zum HTTP Request hinzugefügt werden kann. Als Parameter benötigt diese allerdings ein Objekt der Klasse HttpEntity. Mithilfe der *build()*-Methode der MultipartEntityBuilder-Klasse kann dieses aus den vorher hinzugefügten Dateien erstellt werden.

```
1 MultipartEntityBuilder multipartEntityBuilder = MultipartEntityBuilder.create
    ();
2 multipartEntityBuilder.addBinaryBody("files", this.csvDataConverter.
    convertClientData(sessionComputerData.getClient()).getBytes(), ContentType.
    TEXT_PLAIN, "client");
3 multipartEntityBuilder.addBinaryBody("files", this.csvDataConverter.
    convertDiskStoreData(sessionComputerData.getDiskStores()).getBytes(),
    ContentType.TEXT_PLAIN, "disk");
4 multipartEntityBuilder.addBinaryBody("files", this.csvDataConverter.
    convertPartitionData(sessionComputerData.getPartitions()).getBytes(),
    ContentType.TEXT_PLAIN, "partition");
5 httpPost.setEntity(multipartEntityBuilder.build());
```

Listing 4.17: Erstellen der Multipart HTTP Entity

Für das Übertragen der erfassten Daten wird konkret ein ClosableHttpClient aus dem org.apache.hc.client5.http Package verwendet. Dabei handelt es sich um eine Implementierung der HttpClient Schnittstelle, die verschiedene Variationen der *execute(...)*-Methode zum Ausführen von HTTP-Requests zur Verfügung stellt. Zusätzlich implementiert der ClosableHttpClient die Closable Schnittstelle, die sich darum kümmert, dass die Instanz nach der Benutzung in einem try-resource Block geschlossen wird. Die Funktionsweise der ClosableHttpClient Klasse ist im Listing 4.18 angeführt. Es wird zuerst eine Instanz der

ClosableHttpClient Klasse kann durch den Aufruf der statischen Methode *createDefault()* in der HttpClients Klasse erzeugt. Wie bereits erwähnt, enthält die CloseableHttpClient Klasse eine *execute(...)*-Methode, der als Parameter der erstellte POST-Request und eine Implementierung der HttpClientResponseHandler Schnittstelle übergeben werden kann.

```
1 try (CloseableHttpClient httpClient = HttpClients.createDefault()) {
2     return httpClient.execute(httpPost, this.
     sessionComputerDataResponseHandler);
3 }
```

Listing 4.18: Senden des Requests mit CloseableHttpClient

Der Rückgabewert der *execute(...)*-Methode hängt vom übergebenen HttpClientResponseHandler ab. Der Name deutet schon darauf hin, dass dieser sich um die Verarbeitung der HTTP Response kümmert. Je nach Definition des REST Endpunktes ist die Antwort natürlich eine andere. Darum ist es notwendig einen eigenen HttpClientResponseHandler zu implementieren. Hier handelt es sich um einen HttpClientResponseHandler vom generischen Typ Integer, somit wird auch von der *execute(...)*-Methode ein Rückgabewert vom Datentyp Integer zurückgegeben. Nachdem der REST Client die Antwort der RESTful API erhalten hat, ruft er selbstständig die *handleResponse()*-Methode des HttpClientResponseHandler Objektes auf.

Rechner-Daten

Im Listing 4.19 wird die *handleResponse(...)*-Methode der SessionComputerDataResponseHandler Klasse angeführt, die für die Verarbeitung der Antwort des Rechner-Daten POST Endpunktes verantwortlich ist. Zu Beginn wird der Statuscode mit der *getCode()*-Methode aus dem erhaltenen ClassicHttpResponse geholt. Weiters wird die HTTP Entity, also die JSON Datei in der Antwort, mit der *getEntity()*-Methode angefragt und mithilfe der *toString(...)*-Methode in der EntityUtils Klasse in einen String umgewandelt. Danach wird die Methode *retrieveStateIdFromResponseContent(...)* mit dem Inhalt der HTTP Response aufgerufen. Diese verwendet einen ObjectMapper, um die key-value Paare in einer Map zu speichern. Mithilfe der Map kann dann die stateId ausgelesen und zurückgegeben werden. Die stateId wird dann in der Agent-Klasse der Anwendung gespeichert, um sie bei

allen nachfolgenden HTTP Requests als Identifikation für den Rechner und dessen Status zu verwenden.

```
@Override
public Integer handleResponse(ClassicHttpResponse classicHttpResponse) throws
    HttpException, IOException {
     int statusCode = classicHttpResponse.getCode();
     String responseContent = EntityUtils.toString(classicHttpResponse.
    getEntity(), StandardCharset.UTF_8);
     return retrieveStateIdFromResponseContent(responseContent);
}
```

Listing 4.19: HttpClientResponseHandler für Rechner-Daten

Ressourcenverbrauch

Wenn es sich um den Endpunkt für den durchschnittlichen Ressourcenverbrauch der letzen 60 Sekunden handelt, wird eine andere Implementierung der HttpClientResponseHandler Klasse, genauer gesagt die RunningDataResponseHandler Klasse, verwendet. Dem Listing 4.20 kann entnommen werden, dass die *handleResponse(...)*-Methode dieser Klasse ebenfalls den Statuscode aus der HTTP Response ausliest und zurückgibt. Im Vergleich zu der SessionComputerDataResponseHandler Klasse, wird hier nicht der Inhalt der HTTP Response ausgelesen, da vom aufgerufenen POST Endpunkt keine für den Agent relevanten Daten zurückgegeben werden.

```
@Override
public Integer handleResponse(ClassicHttpResponse classicHttpResponse)
        throws HttpException, IOException {
    return classicHttpResponse.getCode();
}
```

Listing 4.20: HttpClientResponseHandler für den Ressourcenverbrauch

4.1.7 Programmablauf

Der Agent erfasst alle 10 Sekunden die Netzwerk-Schnittstellen, die IP-Verbindungen, sowie den Ressourcenverbrauch des Rechners und der aktuell laufenden Prozesse. Das

heißt, dass die Methode zur Überwachung dieser Informationen nach Ablauf einer gewissen Zeitspanne aufgerufen werden muss. Um dies zu erreichen, kann folgende Technologie verwendet werden:

ScheduledExecutorService

Das java.util.concurrent Package bietet Möglichkeiten bestimmte Aufgaben innerhalb von Anwendungen gleichzeitig auszuführen, beziehungsweise den Zeitpunkt der Ausführung zu steuern. Ein Teil dieses Packages ist die Executor Schnittstelle, die die Registrierung und die Ausführung einer Aufgabe zeitlich voneinander trennt. Implementiert wird diese Schnittstelle von einer weiteren Schnittstelle ExecutorService, die Funktionen zum Beenden und Überwachen der Aufgabenausführung hinzufügt. Wiederum eine Subschnittstelle davon ist die SchedulededExecutorService Schnittstelle. Diese bietet die Möglichkeit die Ausführung von Aufgaben in periodischen Abständen oder nach Ablauf einer gewissen Zeit. [20] Zusätlich zu den vererbten Methoden der Executor und ExecutorService Schnittstellen definiert die ScheduledExecutorService Schnittstelle folgende Methoden, die in der Tabelle 4.2 angeführt werden.

Name	Rückgabe-wert	Parameter	Beschreibung
schedule	Scheduled-Future	Callable callable long delay TimeUnit unit	registriert eine Aufgabe (Implementierung der Callable Schnittstelle), welche nach Ablauf der angegebenen Wartezeit einmalig ausgeführt wird
schedule	Scheduled-Future	Runnable command long delay TimeUnit unit	registriert eine Aufgabe (Implementierung der Runnable Schnittstelle), welche nach Ablauf der angegebenen Wartezeit einmalig ausgeführt wird
scheduleAtFixedRate	Scheduled-Future	Runnable command long initialDelay long period TimeUnit unit	registriert eine Aufgabe (Implementierung der Runnable Schnittstelle), welche nach Ablauf einer anfänglichen Verzögerung in periodischen Zeitabständen ausgeführt wird
scheduleWithFixedDelay	Scheduled-Future	Runnable command long initialDelay long delay TimeUnit unit	registriert eine Aufgabe (Implementierung der Runnable Schnittstelle), welche nach Ablauf einer anfänglichen Verzögerung und mit einer fixen Verzögerung zwischen den einzelnen Aufrufen ausgeführt wird

Tabelle 4.2: Methoden der ScheduledExecutorService Schnittstelle

Für die Umsetzung der Datenerfassung in dieser Diplomarbeit ist die Methode *scheduleAtFixedRate(...)* verwendet worden, da sie die Anforderungen genau erfüllt. Der Aufruf dieser Methode wird im Listing 4.21 dargestellt.
Zuerst muss eine Instanz der ScheduledExecutorService Schnittstelle erstellt werden. Das übernimmt die Executors Klasse, die verschiedene Factory-Methoden, wie die *newScheduledThreadPool(...)*-Methode zur Verfügung stellt. Dabei kann angegeben werden, wie viele Threads der ExecutorService zum Ausführen der Aufgaben zur Verfügung hat. Mehrere Threads sind vor allem dann hilfreich, wenn es sich um komplexe Aufgaben wie Datei- oder Datenbankzugriffe handelt. Hier genügt ein Thread, da die Laufzeit der Datenerfassung in der Regel weit unter 10 Sekunden liegt.
Danach kann die Aufgabe dem ExecutorService übergeben werden – in diesem Fall wird eine Methodenreferenz auf die Methode *monitorData()* übergeben. Weiters sind als anfängliche Verzögerung ein Wert von 0 Sekunden und als generelle Verzögerung ein Wert von 10 Sekunden festgelegt worden.

```
1 ScheduledExecutorService scheduler = Executors.newScheduledThreadPool(1);
2 scheduler.scheduleAtFixedRate(this::monitorData, 0, 10, TimeUnit.SECONDS);
```

Listing 4.21: Datenerfassung alle 10 Sekunden durchführen mit ScheduledExecutorService

4.2 Datenverarbeitung

Die Datenverarbeitung und Datenanalyse wird auf dem Server durchgeführt beim Senden und Abrufen von Daten durch den Benutzer. Konkret ist es notwendig, die Daten vor dem Abspeichern zusammenzuführen und vor der Visualisierung die Daten auf Anomalien und Ereignissen im Trendverlauf zu analysieren.

4.2.1 Zusammenführung

Da der Agent den Hardware-Verbrauch pro laufendem Prozess misst, die Applikation aber den Hardware-Verbrauch von Applikationen und den gesamten Rechner darstellt, müssen vor der Speicherung der Daten diese zunächst zusammengeführt werden. Zum Zweck der Berechnung des gesamten PC-Verbrauchs wird die *groupby*-Methode von Pandas verwendet, welche Daten pro Timestamp zusammenführt und die einzelnen Werte dabei summiert. Da der Agent bereits die Daten auf die Korrektheit überprüft und filtert, ist keine Bereinigung der Daten am Server notwendig.

4.2.2 Datenbeschaffenheit

Bei der Datenverarbeitung und vor allem bei der Anwendung von statistischen Methoden oder Maschine-Learning-Algorithmen ist es notwendig, sich mit den Eigenschaften der vorhandenen Daten auseinanderzusetzen. Viele Modelle oder Methoden benötigen spezifische Eigenschaften von Daten um sie zu analysieren oder zu verarbeiten, erfüllen die Daten diese Anforderungen nicht, müssen entweder einfachere oder andere Methoden verwendet werden oder aber die Daten transformiert werden. In diesem Kapitel werden hauptsächlich die relevanten Eigenschaften beschrieben, welche die verwendeten Modelle benötigen. Zusätzlich werden Probleme angesprochen, die sich durch die Beschaffenheit der Daten ergeben haben.

Stationarität

Stationarität von Daten bedeutet, dass statistische Eigenschaften wie Mittelwert und Varianz über die Zeit konstant bleiben. Viele statistische Methoden oder Modelle setzen die Stationarität von Daten voraus, da sie das Arbeiten mit Zeitreihendaten erleichtern. Bei

den Messdaten handelt es sich um nicht-stationäre Daten, weswegen sie bei Modellen wie ARIMA in stationäre Daten durch Differenzierung umgewandelt werden. Für die genaue Erklärung siehe Kapitel 4.2.5. [68]

Normalverteilung

In der Statistik handelt es sich bei der Normalverteilung, als auch Gauß-Verteilung bekannt, um eine Wahrscheinlichkeitsverteilung. Daten werden als normalverteilt bezeichnet, wenn die Verteilung einer spezifischen Form folgt, welche als Glockenkurve gekennzeichnet wird. Diese ist symmetrisch um den Mittelwert der Daten und fällt nach beiden Seiten hin steil ab, Zweidrittel aller Messwerte liegen innerhalb der Entfernung der Standardabweichung zum Mittelwert. Viele statistische Verfahren, unter anderem die erwogene Z-Score-Anomalieerkennung, beruhen auf der Annahme einer Normalverteilung. [57]

Saisonalität

Mit Saisonalität sind wiederkehrende Muster oder Zyklen zu bestimmten Intervallen von Zeitreihen gemeint. Dies führt dazu, dass die Daten nicht-stationär sind. Statistische Modellen wie ARIMA sind in der Lage, Daten mit Saisonalität zu behandelt, allerdings treten wiederkehrende Muster wie das Installieren von Applikationen durch den Benutzer bei den Messdaten unregelmäßig auf. Zu Beginn des Projektes ist angenommen worden, dass es sich bei dem freien Speicherplatz um saisonale Daten handelt, allerdings ist beim Arbeiten mit Daten schnell erfasst worden, dass durch die irregulären Intervalle die Daten keinerlei Saisonalität aufweisen. [27]

Anomalien

Anomalien, also Extremwerte, erschweren das Analysieren und das Arbeiten mit den verfügbaren Daten, da sie möglicherweise Erkenntnisse bei statistische Methoden und Maschine-Learning-Algorithmen verfälschen oder verzerren. In vielen Datenbeständen werden daher Anomalien entfernt. Allerdings handelt es sich bei Anomalien in den Messdaten nicht um Messfehler, daher ergibt sich nicht die Möglichkeit, diese ausfiltern zu lassen. [113]

Eine andere Möglichkeit ist es, die Auswirkung von Anomalien mit Methoden wie beispielsweise der Verwendung vom gleitenden Durchschnitt zu reduzieren. Daher gibt es die Option, das Auslösen von Benachrichtigungen anhand vom gleitenden Durchschnitt durchzuführen. Die Überlegung dies auch auf andere Aspekte wie Vorhersagen und Change-Point-Erkennung zu erweitern hat es zwar gegeben, allerdings ist das Vorkommen von Anomalien mit den verwendeten Modellen und Algorithmen kein Problem gewesen. Aus diesem Grund ist darauf verzichtet worden.

Einschränkungen der Daten

Die verfügbaren Daten werden zwar von Client-Rechnern erzeugt, allerdings ist es der Benutzer, welcher in den meisten Fällen für den Anstieg oder Fall von Ressourcenverbrauch verantwortlich ist. Solche Probleme werden herkömmlicherweise durch Hinzufügen zusätzlicher Parameter gelöst, allerdings ist der Agent nicht in der Lage, mehr Daten als RAM, CPU und Speicherverbrauch zu messen. Dies macht es schwierig, bestimmte Features wie Vorhersagen zu implementieren. Bei RAM- und CPU-Verbrauch wird es vollständig unmöglich, da Benutzeraktionen nicht abgeschätzt werden können, bei Vorhersagen von freien Speicher muss vom derzeitigen Trend ausgegangen werden.

Zeitlücken in Messdaten

Ein weiteres aufgetretenes Problem sind Zeitlücken bei Messdaten. Der Grund dafür ist, dass der Agent möglicherweise nicht ständig ausgeführt wird oder der PC gestoppt oder gestartet worden ist. Das führt dazu, dass enorme Sprünge in den verfügbaren Daten vorkommen, was vor allem bei den Graphen des freien Speichers erkennbar ist.

Diese Datensprünge müssen bei der Begründungsnachricht beachtet werden da diese potenziell zu falschen Erkenntnisse oder Analysen führen. Beispielsweise kann ein Sprung bei den Messzeitpunkten zur Erkennung einer nicht vorhandenen Anomalie und Ereignisses führen.

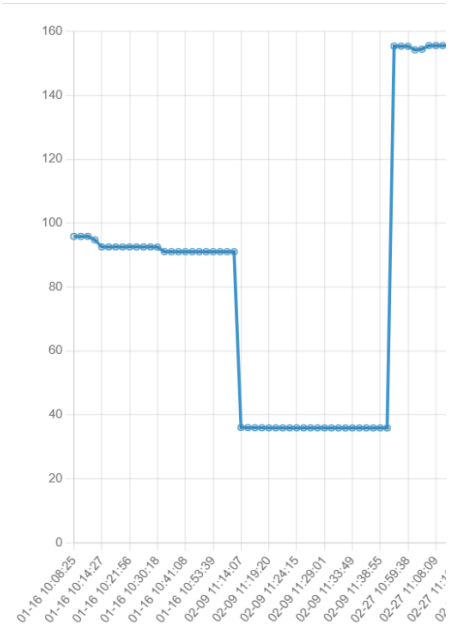


Abbildung 4.1: Zeitsprünge bei freien Speicher

Wie an der Abbildung 4.1 erkennbar ist, gibt es unregelmäßige Zeitlücken bei den erfassten Daten. Diese erzeugen Schwierigkeiten, Forecasts vom freien Speicher zu erstellen. Vor allem die Bildung einer linearen Regression wird dadurch beeinträchtigt und das Verwenden der Gesamtdaten mit Zeitsprüngen verfälscht die Vorhersage von Daten vollständig. Die Lösung dieses Problems wird im entsprechenden Kapitel 4.2.5 genauer erläutert.

4.2.3 Anomalieerkennung

Eine Zielsetzung der Anwendung besteht darin, Anomalien und Abweichungen in den PC- und Anwendungsdaten zu identifizieren und sie dem Benutzer kenntlich zu machen. Die Anomalieerkennung zielt darauf ab, ungewöhnliches Verhalten von Applikationen aufzuzeigen, wodurch das Erkennen von Problemen beim Rechner vereinfacht wird. Durch die Kombination mit Begründungen, bekommt der Benutzer zusätzliche Hinweise darauf, warum diese Anomalien auftreten. [113]

Implementierungsmöglichkeiten

Für das Erkennen von Extremwerten bieten sich eine Vielzahl von Möglichkeiten an, dazu zählen etwa das Aufteilen der Datensätze in Quantile, das Verwenden von Maschine-Learning-Algorithmen wie Isolation Forest und K-means oder das Anwenden von statisti-

schen Methoden in Verbindung mit eigenen Algorithmen.

Händische Implementierung

Im ersten Schritt werden Extremwerte anhand von bestimmten, prozentuellen Abweichungen vom gleitenden Durchschnitt erkannt. Zunächst wird dafür der gleitende Durchschnitt von Datensätzen berechnet und dann die prozentuelle Änderung von einem Datenpunkt zum gleitenden Durchschnitt. Überschreitet dieser Wert eine definierte Grenze, beispielsweise 30 Prozent, wird der Datenpunkt als Extremwert identifiziert. Diese Vorgehensweise ist zwar ausreichend für das Erkennen von den meisten Extremwert-Punkten, jedoch nicht geeignet für Daten welche hohe Varianz aufweisen. Der verwendete Code ist in Listing 4.22 angeführt.

```
def findAnomaliesRoll(selected_row, selected_value, column,
    anomaly_map, event_sensitivity: float = 0.3):
    selected_row['PercentageChange'] =
    selected_row[column] / selected_row['MovingAvg'].shift()

    anomaly_map[selected_value] = []

    for index, row in selected_row.iterrows():
        percentage_change = row['PercentageChange']
        if abs(percentage_change - 1) > event_sensitivity:
                anomaly_map[selected_value].append(AnomalyData(index,
                percentage_change, row[column]))
```

Listing 4.22: Händische Implementierung eines Algorithmus

Anomalieerkennung mit Isolation Forest

Zur Anomalieerkennung ist Isolation Forest zur Verwendung gekommen, wobei es sich um ein unüberwachtes Maschine-Learning-Algorithmen handelt. Dieser basiert auf das Konzept der Isolation, also der Tatsache, dass Anomalien weniger wahrscheinlich als herkömmliche Datenpunkte auftreten und dadurch leichter isoliert werden.

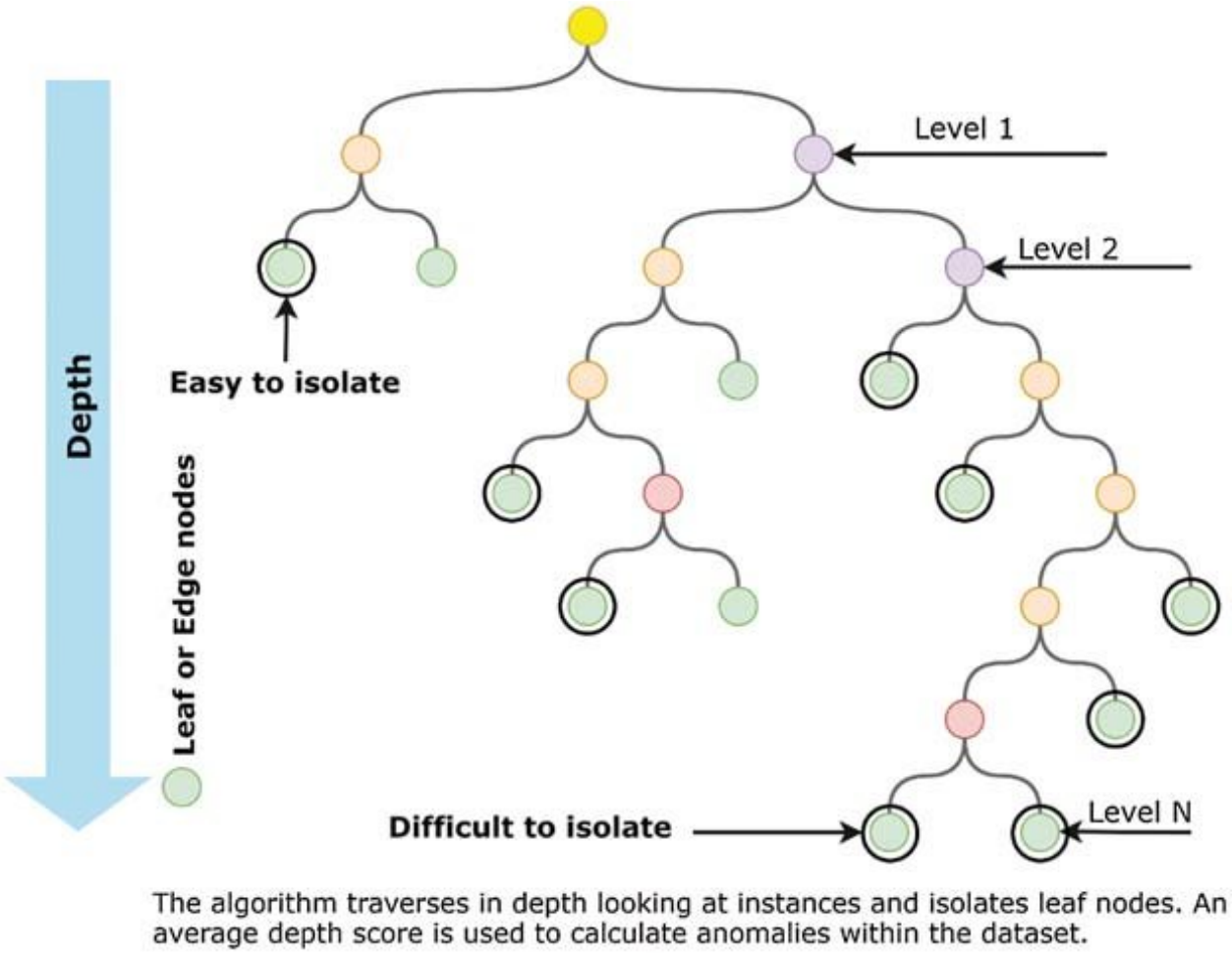


Abbildung 4.2: Entscheidungsbaum von einem trainierten Isolation Forest

Funktionsweise

Zuerst wählt der Algorithmus ein zufälliges Attribut und für dieses Attribut einen zufälligen Schwellenwert aus. Danach isoliert es die Datenpunkte, indem er sie entlang der ausgewählten Attribute trennt. Dies geschieht durch Erstellung eines Entscheidungsbaumes, bei dem die Datenpunkte mit einem Wert über dem Schwellenwert auf einer Seite des Baumes platziert werden und diejenigen mit einem Wert unter dem Schwellenwert auf der anderen Seite platziert werden. Dieser Prozess wird mit jedes Mal mit einem neuen Attribut wiederholt, bis jeder Datenpunkt isoliert ist oder ein vordefiniertes Abbruchkriterium erreicht ist. Anomalien werden von Isolation Forest als Datenpunkte betrachtet, die weniger Partitionen benötigen, also weniger Schritte brauchen, um isoliert zu werden. Die Anomalieerkennung erfolgt daher durch die Tiefe des Baumes, wobei Datenpunkte in den

oberen Knoten als Anomalien und alle restliche als herkömmliche Datenpunkte betrachtet werden. In Abbildung 4.2 wird ein trainierter Entscheidungsbaum von Isolation Forest dargestellt. [95] [36]

Implementierung

Die Anomalieerkennung mit dem Isolation Forest hat gegenüber der händischen Implementierung und statistischen Methoden wie Z-Scores den Vorteil, dass es keinerlei Annahmen über die Verteilung der Daten benötigt und problemlos mit Daten hoher Varianz arbeiten kann. Aus diesen Gründen ist für dessen Verwendung entschieden worden. Bei der Implementierung sind allerdings Probleme bezüglich Underfitting aufgetreten, falls nur eine geringe Menge an Datensätzen zum Trainieren des Algorithmus verfügbar gewesen sind. Dies ist der Fall, wenn der Benutzer die Datenanalyse in einem Zeitrahmen anfordert, wo nur eine geringe Menge an Datensätzen verfügbar sind. Als Lösung sind zum Trainieren des Algorithmus nicht nur die Daten im gewählten Zeitrahmen genommen worden, sondern auch vorher liegende Datensätze.

```
1    training_df = pc_total_df  # dataframe used to train the models
2
3    # fetch more data if needed to avoid underfitting
4    if len(df.index) < 50:
5        training_df, extended_list = get_ram_time_series_limited(pc_id, 100)
6
7    # detect anomalies
8    anomaly_measurements = detect_anomalies(pc_total_df, training_df, 'value')
```

Listing 4.23: Aufruf von detect anomalies

In dem Listing 4.23 wird der Aufruf der Funktion erläutert. Bei Aufruf der Funktion zur Anomalieerkennung müssen ein DataFrame zur Vorhersage von Anomalien und ein DataFrame zum Trainieren des Models übergeben werden. In den meisten Fällen weisen beide DataFrames die gleichen Daten auf, außer es kommt zum angesprochenen Problem, dass zu wenige Trainingsdaten vorhanden sind. Dies ist der Fall, wenn weniger als 50 Datenpunkte im DataFrame vorkommen. Um mehr Trainingsdaten zu haben, wird in

diesem Fall eine Funktion aufgerufen, die von der Datenbank die letzten 100 Datenpunkte anfordert. Als letzter Parameter wird der Name der Spalte übergeben, in diesem Beispiel wird die gewählte Spalte aus der Datenbank selektiert.

```
def detect_anomalies(predicted_df: DataFrame, training_df: DataFrame,
    column: str, contamination_rate: float = 0.03):
    # Extract values from the data
    df_training_values = training_df[column].values.reshape(-1, 1)
    df_prediction_values = predicted_df[column].values.reshape(-1, 1)

    # Model training and prediction
    clf = IsolationForest(contamination=contamination_rate, random_state=42)
    clf.fit(df_training_values)
    predicted_labels = clf.predict(df_prediction_values)

    # Extract measurement times when anomalies occur
    anomaly_indices = np.where(predicted_labels == -1)[0]
    anomaly_measurement_times = predicted_df['measurement_time'].
    iloc[anomaly_indices].tolist()

    return anomaly_measurement_times
```

Listing 4.24: Algorithmus für die Anomalienerkennung

In Listing 4.24 wird die Funktion für die Anomalieerkennung angeführt.Nach Aufruf der Funktion müssen die betroffenen Daten aus den DataFrames extrahiert werden, als Übergabewert verlangt der IsolationForest-Algorithmus nämlch ein numpy-Array. Mit *.values* von Pandas wird die betroffene Spalte des DataFrames als Series ausgewählt und anschließend mit *reshape(-1,1)* in eine Single-Column umgewandelt. Danach wird das Model trainiert mit einer festgesetzten Kontaminierungsrate von 0.03. Die Kontaminierungsrate gibt den Prozentsatz der erwarteten Anomalien an, in diesem Fall ist der Prozentsatz von 3 gewählt worden, da dieser Wert bei Versuche die besten Ergebnisse erzielt hat. Da das Ergebnis reproduzierbar sein sollte, wird ein fixer Random State von 42 gesetzt. Die Wahl dessen hat keinerlei Einfluss auf die Qualität der Anomalieerkennung und dient nur für

konstante Ergebnisse bei gleichen Daten. Ohne einen konkreten Random State besteht ansonsten die Möglichkeit, dass bei identen Datenbeständen verschiedene Anomalien pro Request erkannt werden. Der Algorithmus markiert alle Anomalien mit -1 und alle herkömmlichen Datenpunkte mit 1, daher werden mithilfe von *.where* der numpy-Bibliothek die Anomalie-Datenpunkte ausgewählt, bei dem die predicted labels -1 betragen. Die Anomalieerkennung ist damit abgeschlossen, allerdings besteht die erzeugte Liste nur aus den Indizes der Datenpunkte, bei denen eine Anomalie aufgetreten ist. Damit die restlichen Algorithmen in der Lage sind, leichter mit den Daten zu arbeiten, werden die Messzeiten der Indizes aus der Vorhersagens-DataFrame ausgewählt und zurückgegeben.

4.2.4 Change Point Detection

Zusätzlich zur Erkennung von Anomalien benötigt LogSense auch das Erkennen von Change Points zur Identifizierung von Ereignissen und zum Feststellen von Trends zwischen den Change Points. Change Points sind Zeitpunkte in einer Zeitreihe, an denen sich das Verhalten der Daten oder statistische Eigenschaften signifikant ändern, beispielsweise durch einen enormen Anstieg oder Abfall der Daten.

Pelt-Algorithmus

Bei der verwendeten Implementierung wird der Pelt (Pruned Exact Linear Time)-Algorithmus aus der ruptures-Bibliothek angewendet.

Der Pelt-Algorithmus basiert auf die Idee, dass Change Points dazu führen, dass sich die Daten in einer Zeitreihe in verschiedene Segmente unterteilen lassen, bei denen sich das Verhalten der Daten ähnelt. Beispielsweise könnten gemessene Temperaturen in Segmenten von hohen, niedrigen, schwankenden und stabilen Temperaturen unterteilt werden. Die Change Points liegen dann zwischen den unterteilten Segmenten.

Pelt verwendet außerdem eine dynamische Programmierungstechnik, um eine optimale Anzahl von Change Points zu finden, damit Daten in sinnvolle Segmente unterteilt werden. Konkret bestraft Pelt das Modell für jeden zusätzlich gefundenen Change-Point, indem zu den Kosten eines Segments ein Bestrafungswert addiert wird. Der Algorithmus wählt dann

die Segmentierung aus, bei der die Gesamtkosten (Kosten der Segmente plus Bestrafung der Änderungspunkte) am geringsten sind. [96] [63] Dieser Bestrafungswert muss je nach Beschaffenheit der Daten angepasst werden, bei einem zu niedrigen Wert kommt es zu vielen falschen positiven Ergebnissen, während bei einem zu hohen Wert relevante Change Points nicht erkannt werden.

Wahl vom Bestrafungswert

Der Bestrafungswert ist durch manuelle Anpassung bestimmt worden durch das Vergleichen der erkannten Change Points mit dem Graphen des RAM beziehungsweise CPU-Verbrauchs. Bei den ersten Implementierungen ist als Bestrafungswert der Wert 1 verwendet worden, dieser hat jedoch zu sehr vielen falschen positiven Ergebnisse geführt, wie in der Abbildung 4.3 durch die lila Punkte dargestellt.

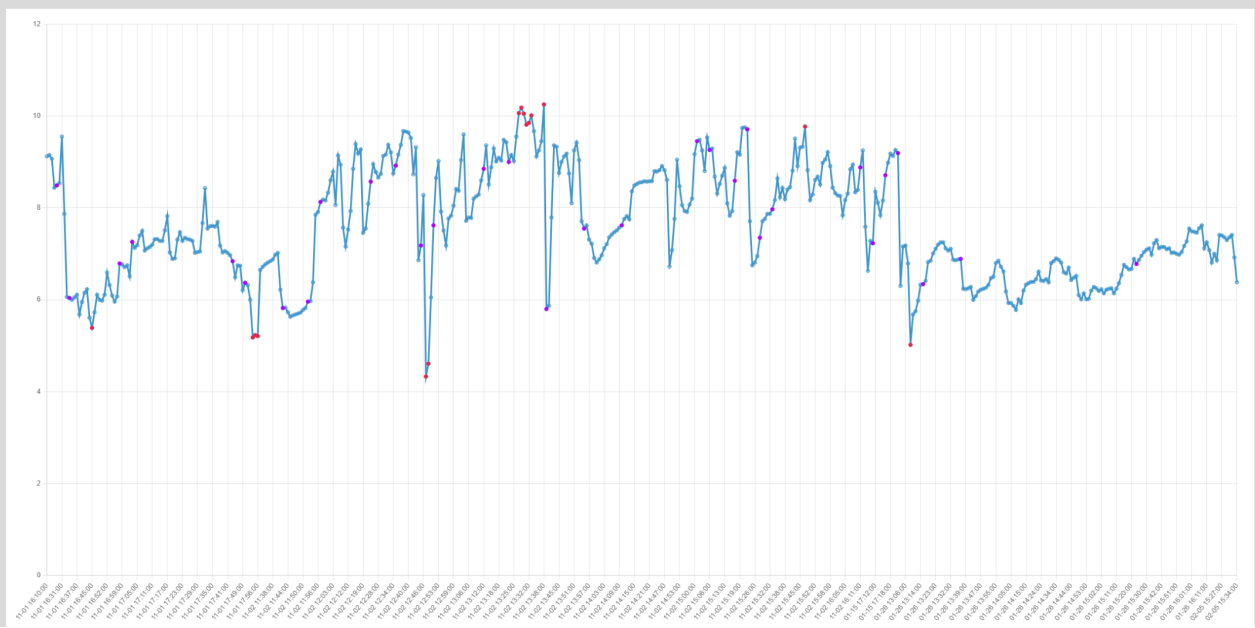


Abbildung 4.3: Verwendung von Bestrafungswert 1

Ob etwas als falsches positives Ergebnis zählt oder nicht kann bei der Change Point Erkennung nicht eindeutig bestimmt werden, jedoch verliert die Change Point Erkennung den Nutzen, wenn beinahe jede kleine Änderung im Trend als Change Point registriert wird. Umgekehrt sind bei einem Bestrafungswert von 10 die erkannten Change Points sehr akkurat gewesen, weswegen dieser Wert auch bei der Segmentierung bei den späteren Vor-

hersagen verwendet wird, allerdings ist die erkannte Anzahl enorm spärlich gewesen und viele potenzielle Change Points nicht registriert worden. Nach mehrmaligen Testen ist ein Bestrafungswert von 3 für optimal bestimmt worden. Dadurch werden alle relevanten und sinnvolle Change Points erkannt mit nur einer geringen Anzahl an falschen Ergebnissen.

4.2.5 Forecast

Ein weiteres Feature von LogSense ist die Möglichkeit, den zukünftigen Verlauf von Zeitserien vorherzusagen. Zwar kann man dies in der Theorie auf alle anderen Daten anwenden, sinnvoll ist es allerdings nur bei der Vorhersage von freiem Speicherplatz, da andere Metriken wie etwa RAM- oder CPU-Auslastung keine Trends aufweisen.

ARIMA

ARIMA (Autoregressive Integrated Moving Average) ist ein statistisches Modell welches häufig bei der Zeitreihenanalyse und -vorhersage verwendet wird. Aus diesem Grund hat es die Überlegung gegeben, die Vorhersage von freien Speicher mithilfe des ARIMA-Modells durchzuführen, welches basierend auf frühere Zeitreihen kurzfristige Voraussagen treffen kann.
Bei ARIMA handelt es sich um die Erweiterung des sogenannten ARMA-Modells zur Trendbeseitigung und Herstellung der Stationarität um die Differenzierung und Integration. Durch die Verwendung von ARIMA ist es auch möglich, Zeitreihen mit Trends zu analysieren. Im Grunde besteht ARIMA aus drei elementaren Bestandteilen:

Autoregression (AR): Statistisches Konzept, welches sich auf die Abhängigkeit einer Variable von ihren vorherigen Werten bezieht und verwendet wird, um Beziehungen zwischen Variablen und vergangenen Werten zu beschreiben. Dadurch werden Muster und Trends in Zeitreihendaten gefunden und anhand dessen Voraussagen über zukünftige Werte getroffen werden. Autoregressive Modelle wenden dafür eine lineare Regression an, im Gegensatz zur herkömmlichen linearen Regression werden allerdings keine anderen unabhängigen Variablen verwendet. Die Formel dafür lautet: $y_t = \phi_1 y_{t-1} + \phi_2 y_{t-2} + \ldots + \phi_p y_{t-p} + \epsilon_t$

- y_t: Der aktuelle Wert in der Zeitreihe t.

- $y_{t-1}, y_{t-2}, \ldots, y_{t-p}$: Die vorherigen Werte der Zeitreihe, die in die Vorhersage des aktuellen Werts einbezogen werden.
- $\phi_1, \phi_2, \ldots, \phi_p$: Die Autoregressionskoeffizienten, die die Stärke der Beziehung zwischen den vorherigen Werten und dem aktuellen Wert bestimmen.
- p: Die Ordnung der Autoregression, also die Anzahl der vorherigen Werte, die in die Vorhersage einbezogen werden.
- ϵ_t: Der Fehlerterm zur Zeit t, der die Abweichungen zwischen dem tatsächlichen und dem vorhergesagten Wert darstellt.

Bei Verwendung von einem Autoregressiven Modell müssen p und der Autoregressionskoeffizient korrekt geschätzt werden um genaue Vorhersagen zu erstellen. Alternativ kann auch Autoarima eingesetzt worden. Autoarima ist ein statistischer Algorithmus welches einen Bereich von möglichen Parameterkombinationen durchsucht und jene auswählt, die am besten geeignet ist. [44]

Integration (I): Integration von Differenzen in der Zeitreihe, um diese stationär zu machen. Stationarität bedeutet, dass statistische Eigenschaften wie Mittelwert und Varianz über die Zeit konstant bleiben. Außerdem darf es keine Saisonalität geben, also keine regelmäßigen Muster oder Zyklen in Intervallen. Das ARIMA-Modell arbeitet mit der Annahme, dass die Zeitreihendaten stationär sind, daher müssen nicht-stationäre Daten zuerst durch Differenzierung umgewandelt werden. Die ausgerechneten Differenzen ersetzten danach die Werte der Datenpunkte. Die Differenzierung wird mit folgender Formel ausgerechnet:

$$\Delta^d y_t = y_t - y_{t-d}$$

- $\Delta^d y_t$: Die d-te Differenz der Zeitreihe y_t. Diese Differenz wird berechnet, indem der aktuelle Wert y_t um den Wert y_{t-d} subtrahiert wird
- y_t: Der aktuelle Wert der Zeitreihe zur Zeit t
- y_{t-d}: Der Wert der Zeitreihe d Schritte zurück in der Zeit.

[68] MA (Moving Average): Der gleitende Durchschnitt im ARIMA-Modell bezieht sich darauf, dass die Vorhersage eines aktuellen Wartes basierend auf die Fehler der vorigen Vorhersagen gemacht wird. Durch das Einbeziehen von vergangenen Fehlern ist es möglich, die Vorhersagen zu verbessern, da es dem Modell ermöglicht, sich an Veränderungen in den Daten anzupassen, die nicht vollständig im AR-Teil modelliert werden konnten. [43]

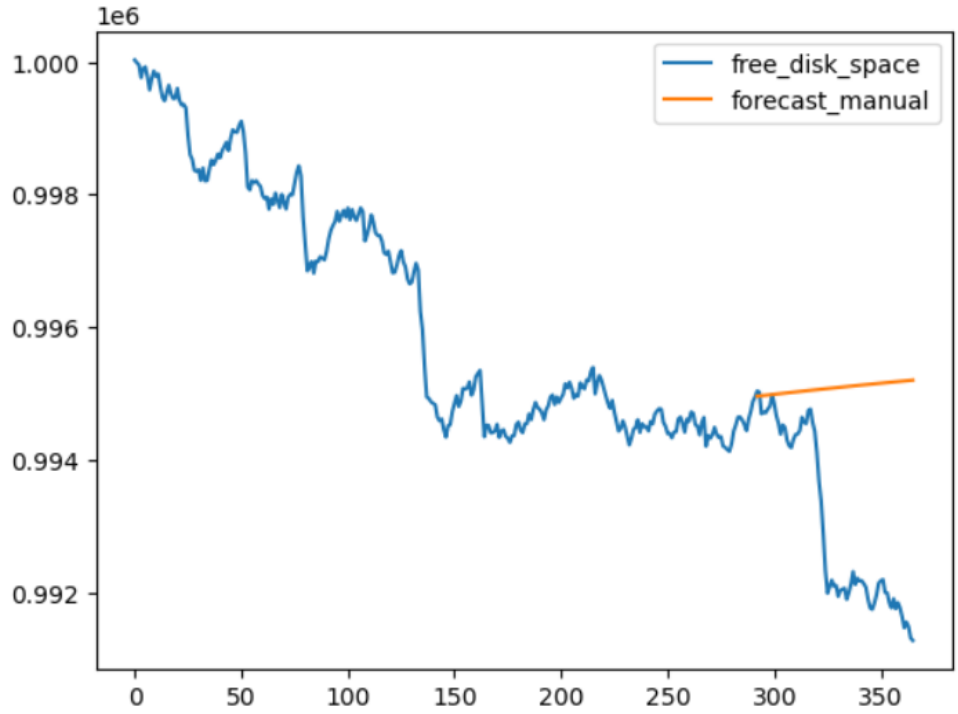


Abbildung 4.4: Vorhersage von freiem Speicher mit Testdaten

Abbildung 4.4 zeigt, dass mit dem trainierten ARIMA-Modell keine sinnvollen Vorhersagen erstellt werden. Das Problem, welches bei der Vorhersage aufgetreten ist, dass immer nur ein kurzer Zeitraum vorhergesagt werden kann. Dies eignet sich nicht für den Anwendungsfall, den Forecast potenziell so lange durchzuführen, dass der freie Speicher verbraucht ist.

Außerdem haben die gemessenen Daten nicht die geeigneten Eigenschaften, um mit ARIMA zu arbeiten. Der Anstieg oder Fall von freien Speicher tretet in zufälligen Intervallen auf, je nachdem, ob der Benutzer etwa Applikationen hinzufügt oder entfernt. Es gibt also keine konkreten Muster oder Saisonalität, mit denen das Modell arbeiten kann, da diese Ereignisse unmöglich zu vorhersagen sind.

Aufgrund der angeführten Gründe ist es schwierig, allgemein bewährte Modelle oder Algorithmen zur Vorhersage von Daten zu verwenden. Daher ist entschieden worden, die Vorhersage anhand einer einfachen statistischen Methode wie der linearen Regression durchzuführen, da nur eine lineare Abhängigkeit zwischen zwei Variablen verlangt wird und sich in den meisten Fällen ein eindeutiger Trend bestimmen lässt.

Lineare Regression

Das Ziel der linearen Regression ist es, einen abhängigen Wert Y durch einen unabhängigen Wert X vorherzusagen. Voraussetzung ist, dass zwischen den beiden Werten ein linearer Zusammenhang besteht und es sich um stetige Werte handelt. [54] In dem konkreten Anwendungsfall den freien Speicher vorherzusagen, ist der abhängige Wert der freie Speicher und der unabhängige Wert der gemessene Zeitpunkt. Im Vergleich zu anderen statistischen Methoden wie ARIMA oder Maschine-Learning-Algorithmen ist die lineare Regression nicht in der Lage, komplexe Muster zu erfassen. Allerdings sind komplexe Modelle aufgrund der zufälligen Datenbeschaffenheit nicht sinnvoll anwendbar, da Voraussetzungen wie Stationarität oder Saisonalität für ihre Verwendung nicht erfüllt sind.

Formel:

y (geschätzte abhängige Variable) $= b$ (Steigung)$\times x$ (unabhängige Variable)$+ a$ (y-Achsenabschnitt)

Im Normalfall wird der freie Speicher durch das Installieren von zusätzlichen Applikationen und mehr gespeicherten Daten stets sinken, daher ergibt sich eine Korrelation zwischen Speicher und Zeit sowie ein stetiger Verlauf des freien Speichers. Allerdings besteht die Möglichkeit, dass der Benutzer am Rechner Daten löscht oder Programme deinstalliert und sich daher der Trend verändert.

```
1     events = detect_events(df, column, 10)
```

```
    df = df.drop(index=df.index[df.index <= events[len(events)-1]]) # only work with latest course (last change point)

    df = df.filter(['measurement_time', column])
    df = df.set_index(pd.to_datetime(df['measurement_time']).astype('int64') // 10 ** 6)
    LR = fit_linear_regression(df, column)

    def fit_linear_regression(df, column):
        LR = LinearRegression()
        LR.fit(df.index.values.reshape(-1, 1), df[column].values)
        return LR
```

Listing 4.25: Auszug aus dem Algorithmus für den Forecast

Aus diesem Grund werden vor dem Bilden der linearen Regression zunächst alle Change-Points in der Zeitreihe bestimmt und die Daten des letzten Segments zum Trainieren des Linear-Regression-Models verwendet, wie am obigen Code-Beispiel angeführt. Dadurch wird auch immer nur der aktuellste Trend beachtet. Der Code dafür wird in Listing 4.25 angeführt. Jedoch dürfen nicht zu viele Change-Points erkannt werden, da ansonsten einfache Schwankungen bereits einen neuen Trend bedeuten. Deswegen wird ein hoher Bestrafungswert verwendet. Bestrafungswerte werden zur Bestrafung von übermäßigen Hinzufügen von Change Points durch den Change-Point-Erkennungsalgorithmus verwendet, zur genauen Erklärung siehe Kapitel 4.2.4. Ein hoher Wert führt dazu, dass kleinere Change Points nicht erkannt werden und dadurch sich die Menge reduziert. Dementsprechend verbleiben nur noch relevante, ausschlaggebende Change Points.

Nach Erstellung des trainierten Modells werden anhand dessen die Werte bei zukünftigen Zeitpunkten vorhergesagt. Da das Feature hauptsächlich der Vorhersage dessen dient, wann der freie Speicher aufgebracht wird, werden die vorhergesagten Werte tageweise gruppiert und Vorhersagen nur bis zu einer bestimmten Grenze von Tagen getroffen (angegeben durch den Benutzer per API-Request). Ohne einer bestimmten Grenze besteht sonst das Risiko, dass Tausende von Tage vergehen müssen, damit der Speicher aufgebraucht ist und daher der Request ungewöhnlich lange benötigt. Auch wird die Vorhersage

von Werten abgebrochen, sobald der Wert negativ ist, da solche Vorhersagen dem Benutzer keinen Nutzen mehr bringen.

4.2.6 Benutzerdefinierte Benachrichtigungen

LogSense bietet dem Benutzer auch die Möglichkeit an, definierte Benachrichtigungen mit bestimmten Auslösern anzulegen, die es ihm ermöglichen, Meldungen zu potenziellen Problemen zu erhalten. Diese bekommt dieser nicht als Push-Benachrichtigung, sondern als Meldung in der Weboberfläche.

Benachrichtigungs-Format

```
{
  "user_id": 1,
  "type": "Breaking threshold alert for Java",
  "message": "Breaking threshold alert for Java",
  "severity_level": 5,
  "conditions": [
    {
      "percentage_trigger_value": 0.1,
      "absolute_trigger_value": 1000000,
      "operator": ">",
      "column": "ram",
      "application": "java",
      "detect_via_moving_averages": true
    }
  ]
}
```

Listing 4.26: Beispiel Benachrichtigung in JSON

Das erstellte Format, welches in Listing 4.26 angeführt ist, ermöglicht eine Benachrichtigung einen bestimmten Typ und eine Nachricht zuzuweisen. Außerdem wird ein Schweregrad gesetzt, der angibt, wie ernstzunehmend die aufgetretene Warnung ist. Unter *Conditions* werden die verschiedenen Auslöser festgelegt. Diese definieren, welche Bedingungen erfüllt sein müssen, damit eine Benachrichtigung ausgelöst und angezeigt wird. `Conditions` bestehen aus folgenden Elementen:

- `percentage_trigger_value`: Der prozentuale Schwellenwert, der überschritten werden muss, um eine Benachrichtigung auszulösen.
- `absolute_trigger_value`: Der absolute Schwellenwert, der überschritten werden muss, um eine Benachrichtigung auszulösen.
- `operator`: Der Operator, der angibt, wie die Bedingungen ausgewertet werden sollen. Möglichkeiten sind >, = und <
- `column`: Gibt die Spalte an auf die sich die Bedingungen beziehen. Zu den möglichen Spalten gehören:
  - `cpu`
  - `ram`
  - `partition_major_faults`
  - `partition_minor_faults`
  - `available_memory`
  - `open_files`

  In den meisten Fällen werden aber nur RAM oder CPU verwendet.
- `application`: Dies gibt die Anwendung an, auf die sich die Bedingungen beziehen. Wenn die Applikation nicht angegeben wird, also null ist, bezieht sich die Bedingung auf den gesamten Rechner.
- `detect_via_moving_averages`: Dies gibt an, ob die Bedingungen über den gleitenden Durchschnitt oder über die einzelnen Datenpunkte erkannt werden. Meist sind eher Bewegungsdurchschnitte sinnvoll, da eine Benachrichtigung nur relevant ist, wenn dieser über eine längere Zeitdauer andauert. Durch die Verwendung vom gleitenden Durchschnitt kann das Anschlagen von nicht-relevanter Benachrichtigungen vermieden werden.

Es muss mindestens ein Schwellenwert definiert werden, wenn mehrere definiert werden, verhalten sie sich wie eine ODER-Bedingung. Die Benachrichtigung wird daher durch den Schwellenwert ausgelöst, der als erster aufgerufen wird.

Implementierung in Python

Für Benachrichtigungen sind in Python die beiden Methoden *check_for_custom_alerts* und *apply_condition* verantwortlich. Hierbei ist es notwendig gewesen, die Überprüfung in mehreren if-Bedingungen zu verschachteln.

```
def check_for_custom_alerts(pc_id, df, custom_alerts, start, end):
    alert_notifications = []

    for alert in custom_alerts:
        for condition in alert.conditions:
            selected_column = condition.column

            # check if it is an application
            if condition.application:
                df, application_data_list = get_application_between(pc_id,
    condition.application, start, end)

            # check if data frame is none (no data has been found)
            if df is not None:
                # check values should be detected with moving averages
                if condition.detect_via_moving_averages:
                    selected_column = 'moving_average_' + condition.column
                    df[selected_column] = df[condition.column].rolling(window
    =5).mean()
                    df[selected_column].fillna(df[condition.column], inplace=
    True)

                filtered_df = apply_condition(df, condition, selected_column)
                create_alert_notifications(filtered_df, alert_notifications,
    alert, condition)

    return alert_notifications
```

Listing 4.27: Durchlaufen der definierten Benachrichtigungen und Bedingungen

Die Funktion check_for_custom_alerts wird in Listing 4.27 angeführt. Diese durchläuft die

angelegten Benachrichtigungen und dessen enthaltene Bedingungen. Wenn in der Bedingung eine Applikation angegeben ist, wird von der Datenbank ein Dataframe der jeweiligen Applikation angefordert. Wenn in der Bedingung angegeben ist, dass Benachrichtigungen mithilfe vom gleitenden Durchschnitt entdeckt werden sollen, wird mithilfe der *.rolling()* Methode von Pandas in einem Zeitfenster von 5 Datenpunkten der gleitende Durchschnitt berechnet. Nachdem diese Bedingungen durchgeführt worden sind, wird die *apply_condition* Funktion aufgerufen.

```
def apply_condition(df, condition, selected_column):
    if condition.percentage_trigger_value:
        state_dict = select_recent_state()
        return df.query(
            f'{selected_column} / {state_dict[condition.column]} {condition.operator} {condition.percentage_trigger_value}')
    elif condition.absolute_trigger_value:
        return df.query(f'{selected_column} {condition.operator} {condition.absolute_trigger_value}')
```

Listing 4.28: Überprüfen der Bedingungen

In der aufgerufenen Funktion, welche in Listing 4.28 dargestellt wird, findet die Überprüfung statt, ob die jeweilige Bedingung aufgetreten ist. Wenn ein prozentueller Auslösewert definiert ist, wird diese in Form eines dynamisch generierten Strings an *df.query()* übergeben, mit dieser Pandas-Funktion werden Dataframes basierend auf eine bestimmte Abfrage gefiltert, ähnlich wie in einer SQL-Query. *df.query()* filtert hierbei alle Zeilen in einem Dataframe danach, ob der jeweilige Wert geteilt durch den gesamten Wert den definierten Schwellenwert übersteigt, beispielsweise 4GB RAM Verbrauch von Chrome dividiert durch 16GB an maximalen RAM-Speicher.

Wenn ein absoluter Auslösewert definiert ist, findet der gleiche Vorgang wie bei einer prozentuellen Bedingung statt. Die einzige Änderung ist, dass es sich hierbei um keine Division handelt sondern nur die jeweilige Zeile mit den absoluten Wert verglichen wird.

```
def create_alert_notifications(filtered_df, alert_notifications, alert, condition):
```

```
    if not filtered_df.empty:
        timestamps = filtered_df['measurement_time'].tolist()
        alert_notification = AlertNotification(
            type=alert.type,
            message=alert.message,
            severity_level=alert.severity_level,
            column=condition.column,
            application=condition.application,
            detected_alert_list=timestamps
        )
        alert_notifications.append(alert_notification)
```

Listing 4.29: Speichern der aufgetretenen Benachrichtigung

Zuletzt wird die *create_alert_notifications*-Funktion aufgerufen, welche die aufgetreten Benachrichtigungen in die alert_notifications-Liste hinzufügt. Diese wird in Listing 4.29 angeführt. Von der gefilterten Liste werden die Timestamps extrahiert durch das Zugreifen auf die Spalte measurement_time welches eine Pandas Series zurückgibt. Damit diese Daten in der Lage sind, mit JSON übertragen zu werden, wird die Pandas-Series in eine Liste umgewandelt durch *.tolist()*. Das Speichern der Zeitpunkte dient zur potenziellen Anzeige der aufgetretenen Benachrichtigungen auf einem Zeitgraphen.

4.2.7 Begründungen

Die Idee hinter dem sogenannten Begründungs-Feature ist es, dem Benutzer zusätzliche Informationen zur Begründung von Ereignissen und Anomalien anzuzeigen. Nur den Zeitpunkt von unvorhergesehenen Verhalten zu kennen ist im Normalfall nicht genug, um sinnvolle Schlussfolgerungen oder zusätzliche Erkenntnisse über Probleme zu erhalten. Aus diesem Grund werden immer beim Auftreten von Anomalien und Ereignissen die Daten im Zeitrahmen vom ersten Auftreten bis zu einer bestimmten Grenze wie etwa 5 Minuten untersucht. Konkret liefern Begründungen folgende Daten:

- `timestamp - till_timestamp`: Gibt den Zeitrahmen an, bei dem die Daten untersucht wurden
- `is_anomaly`: Ob die Begründung für eine Anomalie oder ein Ereignis erstellt wurde

- `justification_message`: Enthält die Begründungsnachricht

Begründungs-Nachricht

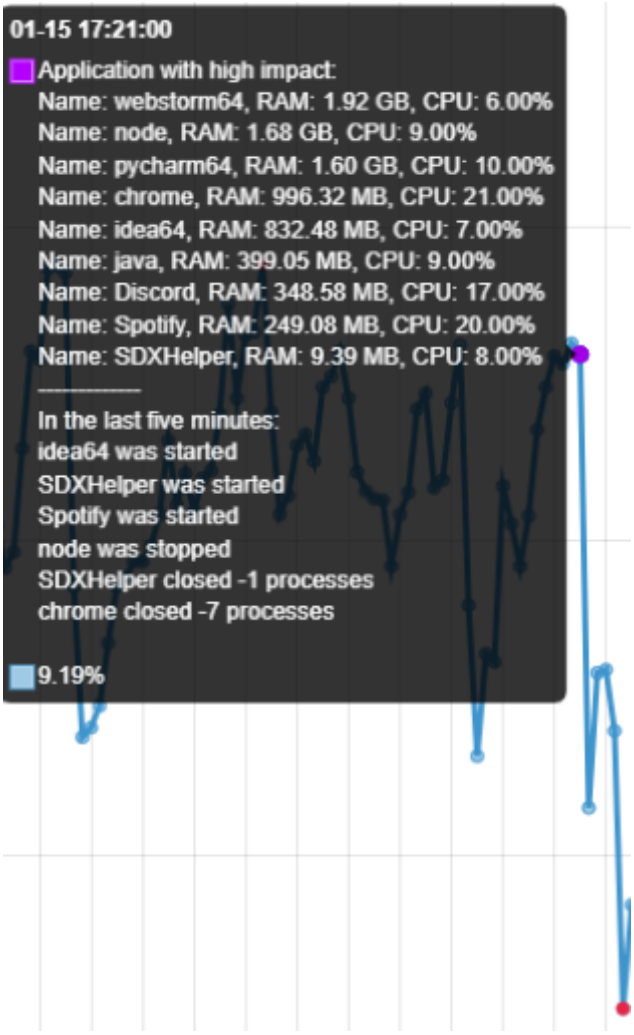


Abbildung 4.5: Begründung eines Ereignisses im RAM-Graph

Die Begründungsnachricht enthält die erkannten und relevanten Informationen, die im untersuchten Zeitrahmen vorliegen. Wie an Abbildung 4.5 erkannt werden kann, beinhaltet eine Begründungsnachricht eine Liste von Applikationen und Informationen darüber, ob Applikationen und deren Prozesse geschlossen oder gestartet worden sind.

Die Liste der Applikationen besteht aus den Applikationen, welche zum jeweiligen Zeitpunkt den höchsten Verbrauch von RAM oder CPU ausgewiesen haben. Durch den ho-

hen Verbrauch wird angenommen, dass diese Applikationen für das Auftreten einer Änderung im Graphenverlauf (Ereignis) oder eines Extrempunktes (Anomalie) am relevantesten gewesen sind. Dadurch bekommt der Benutzer Bescheid, welche Applikationen sich am meisten ausgewirkt haben und kann unvorhergesehenes Verhalten besser zuordnen und behandeln.

Außerdem wird im untersuchten Zeitraum angezeigt, welche Applikationen und Prozessen geschlossen oder gestartet worden sind. Diese sind für die Begründung am hilfreichsten, da sie explizit angeben, was im Zeitraum passiert ist, damit eine Anomalie oder ein Ereignis zustande gekommen sind. Beim angeführten Beispiel etwa ist erkennbar, dass in den letzten fünf Minuten einige Applikationen wie IntelliJ(idea64) und Spotify gestartet worden sind. Dies kann erklären beziehungsweise begründen, warum der Graph bei den letzten Datenpunkten angestiegen ist. Auch steht in der Begründung, dass Chrome 7 Prozesse geschlossen hat. Durch den nächsten, viel tieferliegenden Datenpunkt kann daraus gedeutet werden, dass dies der Grund für den Fall im RAM-Verbrauch gewesen ist.

Implementierung von Begründungen

```
1 ram_relevancy_threshold = 0.05
2 cpu_relevancy_threshold = 0.05
3 applications_df, application_data_list = get_relevant_application_data(pc_id,
     point, ram_relevancy, cpu_relevancy_threshold)
4 pc_just_started = False
5 if applications_df is not None:
6     if applications_df['measurement_time'].nunique() <= 1:
7         pc_just_started = True
8         till_timestamp = applications_df['measurement_time'].min()
9         important_applications, summary_df, started, stopped =
     perform_justification_processing(applications_df)
```

Listing 4.30: Laden der relevanten Daten

In Listing 4.30 wird das Laden der relevanten Daten angeführt. Zu den relevanten Daten zählen die Datensätze von Applikationen, welche prozentuell beim CPU- oder RAM-Verbrauch über 5 Prozent liegen. Zu Beginn wird ein Dataframe von den ausgewählten

Applikationen erstellt, die Daten dafür werden aus der Datenbank geladen. Eine Applikation wird als relevant eingestuft, wenn dessen RAM oder CPU-Verbrauch mindestens 5 Prozent im Zeitrahmen ausmacht. Wenn Daten gefunden worden sind, wird der Zeitpunkt ermittelt, zu dem die geladenen Daten reichen. Standardmäßig beträgt dies immer 5 Minuten, es kann jedoch vorkommen, dass zu wenig Datensätze vorhanden sind und der Zeitrahmen daher verkürzt wird. Danach wird der Applikations-Dataframe an die Funktion *perform_justification_processing*übergeben. Diese Funktion wird in Listing 4.31 angeführt.

```
def perform_justification_processing(df: DataFrame):
    """
    Function to perform data manipulation and processing required for making justifications
    :return:
    """
    # sort by most important applications
    important_applications = df.loc[df.groupby('name')['measurement_time'].idxmax()].sort_values(by='ram',

                        ascending=False)

    # find out process changes in applications
    summary_df = df.groupby('name')['process_count_difference'].sum().reset_index()
    summary_df = summary_df[summary_df['process_count_difference'] != 0]

    # find out which applications were started or stopped
    grouped = df.groupby('measurement_time')['name'].unique().reset_index()
    grouped = grouped.sort_values(by='measurement_time')
    first_names = set(grouped['name'].iloc[0])
    grouped['added'] = grouped['name'].apply(lambda x: list(set(x) - first_names))
    grouped['removed'] = grouped['name'].apply(lambda x: list(first_names - set(x)))

    # put the data into array of string to make working with them easier
```

```
22      started = set([name for name in np.concatenate(grouped['added']) if not pd
    .isna(name)])
23      stopped = set([name for name in np.concatenate(grouped['removed']) if not
    pd.isna(name)])
24
25      return important_applications, summary_df, started, stopped
```

Listing 4.31: Notwendige Datenmanipulationen zur Bildung von Justificatons

Innerhalb dieser Funkton werden alle notwendigen Datenmanipulations-Operationen durchgeführt um Begründungen zu bilden. Der Dataframe wird nach Relevanz sortiert um relevante Ergebnisse innerhalb der Begründungsnachricht als erste anzuführen.

Mithilfe der Pandas-Funkton .groupby() werden die Applikationen im Dataframe nach den Namen gruppiert und die Gesamtanzahl an Prozessänderungen gespeichert solange dessen Änderungen nicht 0 betragen, also keine Prozesse geschlossen oder gestartet worden sind da diese nicht relevant für die Begründung sind.
Als nächstes wird auf ähnliche Art ermittelt, welche Applikationen im Zeitrahmen hinzugefügt (gestartet) oder entfernt (gestoppt) worden sind. Dafür wird die Lambda-Funktion verwendet, welche ein Set bildet aus der Liste der Applikationen und diese mit den Anwendungen subtrahiert, die zu Beginn des Zeitraums bereits vorhanden waren. Das Ergebnis ist eine Liste von Anwendungen, die während des Zeitraums hinzugefügt (gestartet) wurden. Umgehrt wird diese Methode auch zur Ermittlung der gestoppten Applikationen verwendet.
Damit mit den Daten leichter gearbeitet werden kann, werden diese in Sets von Strings umgewandelt. Mit der .concatenate Methode von NumPy wird die Liste von hinzugefügten und entfernten Applikationen zu einem einzelnen NumPy-Array zusammengeführt, überprüft, ob die Einträge nicht Null sind, und schließlich in einem String-Set umgewandelt. Aus diesen Sets wird später eine sinnvolle Begründungsnachricht erstellt, wie in der Abbildung 4.5 angeführt.

4.2.8 Statistiken berechnen

In der Datenwissenschaft spielen Statistiken stets eine Rolle bei der Analyse und Interpretation von Daten. In LogSense werden folgende Statistiken zu Applikationen und Hardware-Ressourcen geliefert:

- Derzeitiger Wert
- Durchschnittlicher Wert im Zeitbereich
- Stabilität (Variationskoeffizient)
- Absolutes Delta
- Prozentuelles Delta
- Standardabweichung

```
def calculate_trend_statistics(df: DataFrame, column: str, name: str) ->
    StatisticData:
    # calculate the stability of data
    std = df[column].std()
    mean = df[column].mean()
    cov = (std / mean) * 100  # stands for coefficient_of_variation

    # calculate changes that occurred from start to end
    recent_row = df.loc[df['measurement_time'].idxmax()]
    oldest_row = df.loc[df['measurement_time'].idxmin()]
    change = ((recent_row[column] - oldest_row[column]) / oldest_row[column])
    * 100
    delta = recent_row[column] - oldest_row[column]

    stability = f"Stability: {determine_stability(cov)}\n"
    message = create_statistics_message(change, delta, name)

    if name.lower() == 'ram':
        average = round(mean / (1024 ** 3), 2)
        current = round(recent_row[column] / (1024 ** 3), 2)
```

```
    elif name.lower() == 'cpu':
        average = round(mean, 2)*100
        current = "{:.2f}".format(round(recent_row[column], 2) * 100)

    statistic_data = StatisticData(
        average=average,
        current=current,
        stability=stability,
        message=message
    )
    return statistic_data
```

Listing 4.32: Berechnung der Statistiken

In Listing 4.32 wird die Berechnung der Statistiken angeführt. Hierbei werden zur Berechnung zur Gänze die bereitgestellten Statistik-Methoden der Pandas-Bibliothek verwendet. Zur besseren und leichteren Darstellung werden außerdem die Werte average (Durchschnittswert) und current (derzeitiger Wert) von Kilobyte in Gigabyte beziehungsweise Megabyte umgerechnet, falls es sich um den RAM-Verbrauch handelt. Beim CPU-Verbrauch wird der Wert auf zweistellige Nachkommastellen abgerundet und mit 100 multipliziert, um die Werte in Prozent darzustellen.

Stabilitätsfestellung

Die Stabilität eines Graphen kann unter anderem mit dem sogenannten Koeffizienten der Variation (Coefficience of Variance) berechnet und bestimmt werden. Der Koeffizient der Variation ist ein Maß dafür, wie stark die Streuung der Daten im Verhältnis zum Mittelwert ist. Durch die Division der Standardabweichung durch den Mittelwert kann der Koeffizient berechnet werden.

Bei der aufgerufenen Funktion determine_stability wird danach die Regel angewendet, dass wenn der Koeffizient geringer als 15 Prozent ist, der Graph als stabil gilt. Zwischen 15 Prozent und 30 Prozent gilt es als mittelmäßig stabil und über 30 Prozent als instabil. Die ermittelte Stabilität wird als Zeichenkette (String) abgespeichert.

4.3 Datenhaltung

Die Datenhaltung erfolgt auf dem Server, die verwendete Datenbank ist eine Timescale Datenbank, welche auf einem Docker läuft. Auf die Datenbank wird von der API mithilfe von Psycopg zugegriffen. Die Daten werden gespeichert, um sie später zur Analyse zur Verfügung zu haben und sie in der Benutzeroberfläche bei Bedarf darzustellen. In den nachfolgenden Abschnitten wird die konkrete Implementierung der Datenbank und der Zugriff darauf beschrieben.

4.3.1 Datenbank

Einleitung

Die verwendete Datenbank ist TimescaleDB, welche auf Postgres aufbaut. Der Vorteil dessen ist es, dass sowohl relationale Datenstrukturen, als auch die für Timeseries Data optimierten, sogenannten 'Hypertable', zur Verfügung stehen. Ein Vorteil dessen ist, dass TimescaleDB schnelle Lese- und Schreibgeschwindigkeiten für große Mengen an Timeseries-Daten anbietet, etwa wie die vom Agent gemessen Datensätze. Die Datenbank läuft in einem Docker Container, da dies eine einfache Möglichkeit bietet, die Datenbank zu verwalten und skalierbar zu machen, unabhängig von der zugrunde liegenden Infrastruktur.

Warum wird eine Datenbank benötigt?

Die vom Agent gemessen Daten müssen für die spätere Analyse und Visualisierung persistiert werden. Weiters wird die Datenbank zum Abspeichern von Benutzerinformationen verwendet.

Datenmodell

Das Datenmodell visualisiert die Tabellen und deren Relationen und wird in Abbildung 4.6 dargestellt.

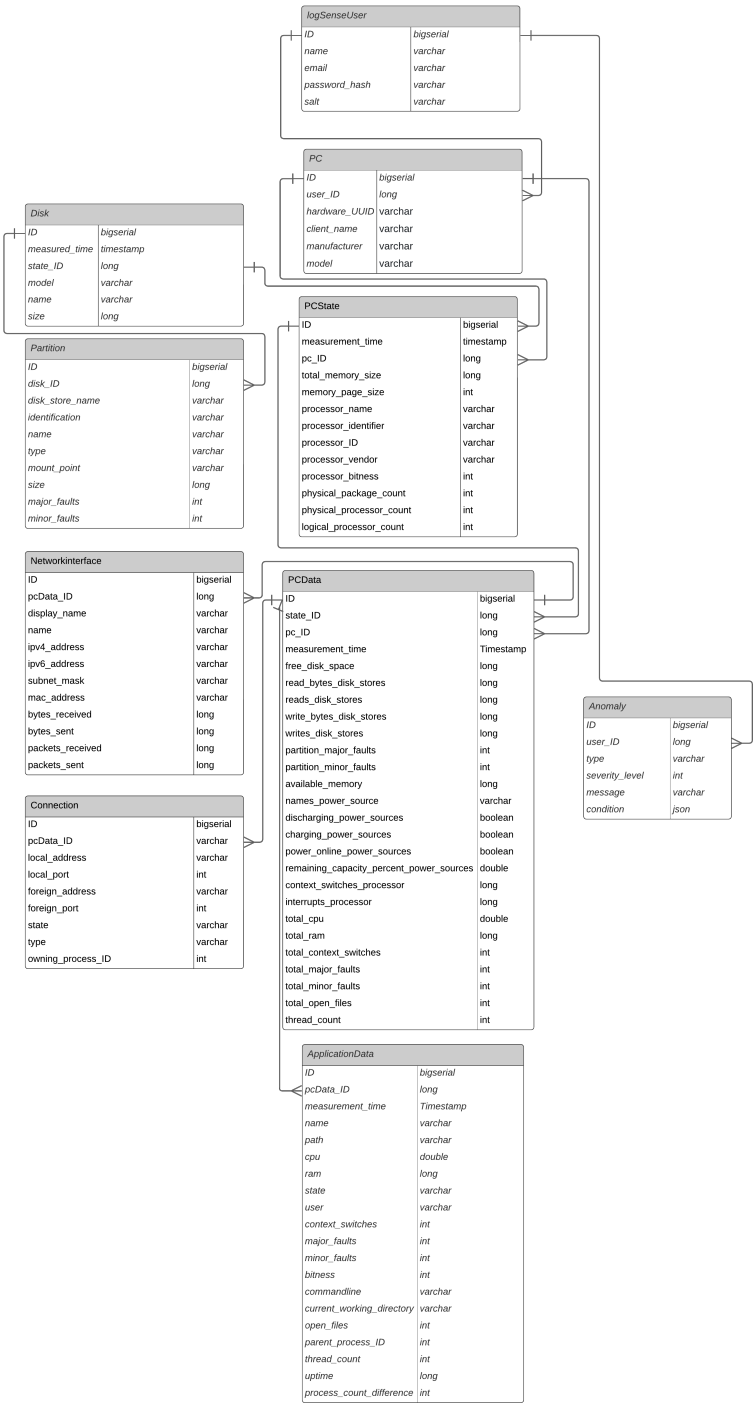


Abbildung 4.6: Datenbankmodell der Anwendung

Datenkatalog

Der Datenkatalog beschreibt die einzelnen Tabellen, deren Funktion und die jeweiligen Attribute.

Hier ist ein PC ist ein Windows Rechner, der sich mit seiner Hardware UUID in der Web Application angemeldet hat und der ab der Anmeldung Daten mithilfe des Agents (Desktop Application) erfassen und an den Server senden kann. Die Attribute eines PCs werden in Tabelle 4.3 beschrieben.

Name	Typ	Beschreibung
ID	bigserial	ein fortlaufender Wert, der einen PC eindeutig identifiziert und der als Primärschlüssel agiert
user_ID	long	ID des Users, zu welchem der PC gehört; dieser Wert agiert als Fremdschlüssel
hardware_UUID	varchar	Vom Hersteller vergebene UUID, die einen PC eindeutig identifiziert. Die UUID ist mit dem Unique Constraint versehen.
client_name	varchar	Der Name, den der Benutzer in der Web Application als Bezeichnung für den PC angegeben hat.
manufacturer	varchar	Herstellername des PCs
model	varchar	Modell des PCs

Tabelle 4.3: Attribute für den Entitätstyp PC

Der Zustand, in dem sich der PC befindet, also welche Hardware dieser PC zum aktuellen Zeitpunkt zur Verfügung hat. Die Attribute eines PCStates werden in Tabelle 4.4 beschrieben.

Name	Typ	Beschreibung
ID	bigserial	fortlaufender Wert, welcher einen PCState eindeutig identifiziert; dieser Wert agiert als Primärschlüssel
measurement_time	timestamp	Zeitpunkt, an dem sich die Hardware geändert hat und diese neuen Daten gemessen worden sind
pc_ID	long	Referenz auf einen eindeutig identifizierten PC; agiert als Fremdschlüssel
total_memory_size	long	verfügbarer Arbeitsspeicher angegeben in Bytes
memory_page_size	int	Größe der RAM Pages
processor_name	varchar	Name des Prozessors
processor_identifier	varchar	Kennzeichnung des Prozessors
processor_ID	varchar	ID des Prozessors
processor_vendor	varchar	Hersteller des Prozessors
processor_bitness	int	Prozessorarchitektur
physical_package_count	int	Anzahl der physischen CPUs
physical_processor_count	int	Anzahl der physischen Kerne der CPU
logical_processor_count	int	Anzahl der logischen Kerne der CPU

Tabelle 4.4: Attribute für den Entitätstyp PCState

PCData ist eine Hypertable, die relevante Hardware-Verbrauchs-Daten für einen gesamten PC speichert. Diese Daten werden minutenweise gespeichert und stellen immer den Verbrauch von Ressourcen eines PCs pro Minute dar. Diese setzt sich aus der Summe aller ApplicationData zu dem jeweiligen Zeitpunkt zusammen. Die Attribute von PCData werden in Tabelle 4.5 beschrieben.

Name	Typ	Beschreibung
ID	bigserial	fortlaufender Wert, welcher eine PCData Tabelle eindeutig identifiziert
state_ID	long	Verweis auf den aktuellen Zustand, in dem sich die Hardware des PCs befindet, also wie viel totale CPU-Leistung, freier Speicher usw. vorhanden sind; agiert als Fremdschlüssel.
pc_ID	long	Referenz auf einen eindeutig identifizierten PC; agiert als Fremdschlüssel
measurement_time	Timestamp	der gemessene Zeitpunkt von einem Datensatz, der diesen eindeutig identifiziert; agiert als Primärschlüssel
free_disk_space	long	Menge von freiem Festplattenspeicher, angegeben in Bytes.
read_bytes_disk_stores	long	Menge von gelesenen Bytes vom Festplattenspeicher.
reads_disk_stores	long	Menge von gelesenen Daten vom Festplattenspeicher.
write_bytes_disk_stores	long	Menge von geschriebenen Bytes auf den Festplattenspeicher.
writes_disk_stores	long	Menge von geschrieben Daten auf die Festplatte.
partition_major_faults	int	Anzahl der größeren Probleme, die auf den Partitionen aufgetreten sind.
partition_minor_faults	int	Anzahl der geringeren Probleme, die auf den Partitionen aufgetreten sind
available_memory	long	Anzahl des verfügbaren RAM-Speichers auf dem PC.
names_power_source	varchar	Namen der Stromquellen, die beim PC eingetragen sind
discharging_power_sources	boolean	Gibt an, ob der Akku des PCs sich entlädt.

Name	Typ	Beschreibung
charging_power_sources	boolean	Gibt an, ob der PC angesteckt ist.
power_online_power_sources	boolean	Gibt an, ob der PC derzeit aufgeladen wird
remaining_capacity_percent_power_sources	double	Prozentueller Wert, der die noch verfügbare Akkuleistung angibt.
context_switches_processor	long	Anzahl der Änderungen, auf welchen Prozess sich eine Applikation fokussiert.
interrupts_processor	int	Anzahl der Unterbrechungen eines Prozessors
total_cpu	double	Der CPU-Verbrauch eines PCs, angegeben in prozentuellen Werten.
total_ram	double	Der RAM-Verbrauch eines PCs, angegeben in Bytes.
total_context_switches	int	Anzahl der Änderungen, auf welche Threads fokussiert wird
total_major_faults	int	Anzahl von aufgetretenen großen Speicherfehlern, wenn Daten von der Festplatte in eine Page geladen werden müssen
total_minor_faults	int	Anzahl von aufgetretenen kleinen Speicherfehlern, wenn eine neue Page geladen werden muss
total_open_files	int	Anzahl der geöffneten Dateien, auf die eine Applikation zugreift
total_thread_count	int	Anzahl der Threads, die der PC verwendet

Tabelle 4.5: Attribute für den Entitätstyp PCData

ApplicationData ist eine Hypertable, die relevante, aufgezeichnete Hardware-Daten für Applikationen speichert. Diese werden minutenweise gespeichert und stellen immer den Verbrauch von Ressourcen einer Applikation pro Minute dar. Die Attribute von ApplicationData werden in Tabelle 4.6 beschrieben.

Name	Typ	Beschreibung
ID	bigserial	fortlaufender Wert, welcher eine Application-Data eindeutig identifiziert
pcData_ID	long	ID des PCs, zu welchem die Applikation gehört; agiert als Fremdschlüssel
measurement_time	Timestamp	Zeitpunkt der Messung eines Datensatzes; agiert als Primärschlüssel
name	varchar	Name der Applikation, der der Datensatz zugewiesen ist
path	varchar	Dateipfad, in dem die Applikation gespeichert ist
cpu	doube	CPU-Verbrauch einer Applikation, in Prozenten
ram	long	RAM-Verbrauch einer Applikation, angegeben in Bytes
state	varchar	Zustand einer Applikation, ob diese noch bei der letzten Messung am Laufen war oder während den Messungen bereits geschlossen wurde
user	varchar	hinterlegter Windows-User einer Applikation zum Messzeitpunkt
context_switches	int	Anzahl der Änderungen, auf welchen Prozess sich eine Applikation fokussiert.
major_faults	int	Anzahl von aufgetretenen größeren Speicherproblemen
minor_faults	int	Anzahl von aufgetretenen geringeren Speicherproblemen
bitness	int	Gibt an, ob die Applikation einen 32 oder 64-Bitprozess darstellt.
commandline	varchar	CMD-Argumente, die für die Ausführung der Applikation verwendet werden
current_working_directory	varchar	derzeitiges Verzeichnis, unter dem die Applikation läuft
open_files	int	Anzahl der geöffneten Dateien, auf die eine Applikation zugreift
parent_process_ID	int	ID eines übergeordneten Prozesses bzw. einer Applikation
thread_count	int	Anzahl der Threads, die eine Applikation verwendet.
uptime	long	Dauer, wie lange eine Applikation seit dem Starten am Laufen ist
process_count_difference	int	Gibt an, ob Prozesse einer Applikation neu gestartet, geschlossen oder gleichgeblieben sind.

Tabelle 4.6: Attribute für den Entitätstyp ApplicationData

Eine Disk ist eine Festplatte, die in einem PC verbaut und in Partitionen unterteilt ist. Die Attribute von Disks werden in Tabelle 4.7 beschrieben.

Name	Typ	Beschreibung
ID	bigserial	fortlaufender Wert, der eine Disk eindeutig identifiziert; agiert als Primärschlüssel
state_ID	long	ID des PCs, zu welchem die Applikation gehört; agiert als Fremdschlüssel
measured_time	timestamp	Verweis auf den aktuellen PCState, also den Hardware-Zustand, in dem sich der PC gerade befindet
serialnumber	varchar	Zeitpunkt, an dem die Daten über die Festplatte gemessen worden sind
model	varchar	Modell der Festplatte
name	varchar	vom Betriebssystem vergebene Bezeichnung für die Festplatte
size	long	Größe der Festplatte

Tabelle 4.7: Attribute für den Entitätstyp Disk

Die Partition Tabelle beschreibt die Partitionen bzw. Unterteilungen einer Festplatte. Die Attribute von Partitions werden in Tabelle 4.8 beschrieben.

Name	Typ	Beschreibung
ID	bigserial	fortlaufender Wert, der eine Partition eindeutig identifiziert; agiert als Primärschlüssel
disk_ID	long	Verweis auf die Disk, zu welcher die Partition gehört; agiert als Fremdschlüssel
disk_store_name	varchar	vom Betriebssystem des Clients vergebener Name der Festplatte, zu welcher die Partition gehört
identification	varchar	Vom Betriebssystem vergebene Nummer für die Disk und die Partition
name	varchar	bei Erstellung angegebener Name der Partition
type	varchar	Art der Partition
size	varchar	Größe der Partition in Bytes
mount_point	long	Ort im Dateisystem, der der Partition zugewiesen ist
major_fault	int	Anzahl der Fehler, die auftreten, wenn auf einen Speicherbereich zugegriffen wird, der nicht im RAM geladen ist oder für welchen man nicht die benötigten Berechtigungen besitzt.
minor_fault	int	Anzahl der Fehler, die auftreten, wenn auf einen Speicherbereich zugegriffen wird, der im RAM geladen ist, aber für den es keinen Assoziationseintrag gibt.

Tabelle 4.8: Attribute für den Entitätstyp Partition

NetworkInterface beschreibt eine physische oder virtuelle Netzwerkkarte, die den PC mit einem Computernetzwerk bzw. den Geräten im Netzwerk verbinden kann. Die Attribute von NetworkInterface werden in Tabelle 4.9 beschrieben.

Name	Typ	Beschreibung
ID	bigserial	fortlaufender Wert, der eine Netzwerkkarte eindeutig identifiziert; agiert als Primärschlüssel
pcData_ID	long	Verweis auf den jeweiligen Eintrag in PCData, also die gemessenen Daten über den PC
display_name	varchar	Bezeichnung der Netzwerkkarte
name	varchar	Name, den das Betriebssystem an die Netzwerkkarte vergibt und der davon abhängt, ob es sich um eine physische oder eine virtuelle Netzwerkkarte handelt
ipv4_address	varchar	IPv4 Adresse der Netzwerkkarte
ipv6_address	varchar	IPv6 Adresse der Netzwerkkarte
subnet_mask	varchar	Anzahl der Bits der Subnetzmaske
mac_address	varchar	MAC-Adresse der Netzwerkkarte
bytes_received	long	Anzahl der Bytes, die die Netzwerkkarte erhalten hat
bytes_sent	long	Anzahl der Bytes, die die Netzwerkkarte versendet hat
packets_received	long	Anzahl der Netzwerkpakete, die die Netzwerkkarte erhalten hat
packets_sent	long	Anzahl der Netzwerkpakete, die die Netzwerkkarte versendet hat

Tabelle 4.9: Attribute für den Entitätstyp NetworkInterface

Eine Connection beschreibt eine Netzwerkverbindung zu einer Netzwerkkarte eines Rechners in einem beliebigen Computernetzwerk. Sie besteht immer aus zwei „Verbindungspartnern“. Die Attribute von Connection werden in Tabelle 4.10 beschrieben.

Name	Typ	Beschreibung
ID	bigserial	fortlaufender Wert, der eine Netzwerkkarte eindeutig identifiziert; agiert als Primärschlüssel
pcData_ID	long	Verweis auf die ID des jeweiligen PCs
local_address	varchar	IP-Adresse des lokalen Rechners
local_port	int	verwendeter Port des lokalen Rechners
foreign_address	varchar	IP-Adresse des Rechners, mit dem der lokale Rechner verbunden ist
foreign_port	int	Port des Rechners, mit dem der lokale Rechner verbunden ist
state	varchar	Zustand, in dem sich die Netzwerkverbindung befindet
type	varchar	Art der Netzwerkverbindung
owning_process_ID	int	Prozess, der die Netzwerkverbindung verwendet bzw. sie „erstellt“ hat

Tabelle 4.10: Attribute für den Entitätstyp Connection

Anomaly ist eine Tabelle, welche festgelegte Benachrichtigungen, die bei der Analyse der Daten auftreten, abspeichert. Diese werden vom User erstellt und daher auch immer zu einem dazugehörigen User abgespeichert. Beim Abfragen bekommt der User dann eine Liste von aufgetretenen Benachrichtigungen. Die Attribute von Anomaly werden in Tabelle 4.11 beschrieben.

Name	Typ	Beschreibung
ID	bigserial	fortlaufender Wert, der eine Anomalie eindeutig identifiziert; agiert als Primärschlüssel
user_ID	long	Verweis auf die ID des jeweiligen PCs; agiert als Fremdschlüssel
type	varchar	E-Mail-Adresse des Benutzers
severity_level	int	Wichtigkeitsgrad der Anomalie
message	varchar	Nachricht, welche die Anomalie beschreibt
condition	json	Die Bedingung, die abgespeichert wird und abgeprüft wird, um eine Benachrichtigung an den User anzuzeigen. Dabei wird die Bedingung im Format eines JSON-Arrays abgespeichert. Die Werte, welche in dem JSON gesetzt werden, sind die Folgenden: • **percentage_trigger_value** Festgelegter prozentueller Wert, bei dem eine Benachrichtigung erscheint • **absolute_trigger_value** Festgelegter absoluter Wert, bei dem eine Benachrichtigung erscheint. • **operator** Legt die Bedingung fest, ob die Werte einer zu analysierenden PC- oder PCData kleiner, größer oder gleich wie die festgelegten Trigger-Values sein müssen. • **column** Spalte in PC oder ApplicationData, bei der die Bedingung überprüft wird • **application** als string gespeicherter Wert, welcher festlegt, bei welcher Applikation die Bedingung überprüft werden muss (falls nicht, wird der gesamte PC überprüft) • **detect_via_moving_average** als boolean gespeicherter Wert, welcher festlegt, ob die Überprüfung mithilfe von Werten für den jeweiligen Zeitpunkt oder mithilfe des gleitenden Durchschnitts durchgeführt werden soll (letzteres ist sinnvoll, damit Ausreißer keine falschen Benachrichtigungen erzeugen).

Tabelle 4.11: Attribute für den Entitätstyp Anomaly

Die logSenseUser Tabelle beschreibt einen registrierten Benutzer bei LogSense mit all seinen Zugangsinformationen. Die Attribute von logSenseUser werden in Tabelle 4.12 beschrieben.

Name	Typ	Beschreibung
ID	bigserial	festgelegte ID eines Benutzers; agiert als Primärschlüssel
name	varchar	Name des Benutzers
email	varchar	E-Mail-Adresse des Benutzers
password_hash	varchar	abgespeicherter SHA-256 Hash eines Benutzer-Passwortes
salt	varchar	abgespeicherter SALT, um ein User-Passwort sicherer zu machen

Tabelle 4.12: Attribute für den Entitätstyp logSenseUser

4.3.2 Verbindung zur Datenbank

Um mit der Datenbank zu interagieren, wird eine Verbindung zu dieser benötigt. Hierfür wird in dieser Arbeit ein von Psycopg bereitgestellter Connection Pool verwendet. Der Connection Pool hat den Vorteil, dass er mehrere Verbindungen hält, welche bei Bedarf einfach angefragt werden, was den Overhead von Erstellen und Schließen von Verbindungen wegfallen lässt, was wiederum die Leistung erhöht.
Daten, wie zum Beispiel das Datenbankpasswort, der Datenbank Benutzername, der Datenbankname, die IP-Adresse des Datenbankhosts und der Port der Datenbank werden in einer Konfigurationsdatei gespeichert. Dies wird getan, um sie nicht statisch innerhalb des Programms zu speichern, sondern sie jederzeit, ohne den Programmcode ändern zu müssen, anpassen zu können.
Die Datenverbindung wird in der __init__.py Datei des db_access Package erstellt, wobei zuerst die Konfigurationen aus der config.ini Datei ausgelesen werden, dann der Connection Pool und zum Abschluss sowohl die Tabellen, als auch die standardmäßig definierten Anomalien, erstellt werden.

Die folgende Funktion, welche in Listing 4.33 dargestellt ist, liest die definierten Werte aus der config.ini Datei aus und gibt sie zurück.

```
1 def get_database_config(path):
```

```
    config = configparser.ConfigParser()
    config.read(path)

    db_host = config.get('Database', 'db_host')
    db_port = config.get('Database', 'db_port')
    db_name = config.get('Database', 'db_name')
    db_user = config.get('Database', 'db_user')
    db_password = config.get('Database', 'db_password')

    if not db_user or not db_password or not db_name or not db_port or not
   db_host:
        raise WrongConfigurationException()

    return db_host, db_port, db_name, db_user, db_password
```

Listing 4.33: Auslesen der Konfiguration

Die im Listing 4.34 dargestellte Funktion liest die SQL Datei zum Erstellen der Tabelle aus und erstellt damit die Tabellen, jedoch nur, falls diese nicht bereits existieren.

```
def create_tables(conn_pool):
    conn = conn_pool.getconn()
    cursor = conn.cursor()
    try:
        with open('tables.sql', 'r') as file:
            sql_statements = file.read()
            cursor.execute(sql_statements)
            conn.commit()
        logging.info("Tables generated successfully.")
        return True
    except FileNotFoundError:
        logging.error("SQL file not found.")
        return False
    except Exception as e:
        logging.error(f"Error generating tables: {str(e)}")
        return False
    finally:
        conn_pool.putconn(conn)

```

```
20 def create_standard_anomalie(conn_pool):
21     conn = conn_pool.getconn()
22     cursor = conn.cursor()
```

Listing 4.34: Erstellen der Tabellen

In der folgenden Funktion (Listing 4.35) werden die in der Datei 'standardAnomalie.sql' festgelegten Anomalien in die Datenbank eingefügt.

```
1  def create_standard_anomalie(conn_pool):
2      conn = conn_pool.getconn()
3      cursor = conn.cursor()
4
5      try:
6          with open('standardAnomaly.sql', 'r') as file:
7              sql_statements = file.read()
8              cursor.execute(sql_statements)
9              conn.commit()
10         logging.info("Anomalie generated successfully.")
11         return True
12     except FileNotFoundError:
13         logging.error("SQL file not found.")
14         return False
15     except Exception as e:
16         logging.error(f"Error generating Anomalies: {str(e)}")
17         return False
18     finally:
19         conn_pool.putconn(conn)
```

Listing 4.35: Einfügen der standardmäßig definierten Anomalien

In dem Listing 4.36 wird die Verbindung zur Datenbank und Erstellung der Tabellen mithilfe der Funktionen der Listings 4.33 bis 4.35, sowie die Erstellung des Connection Pools dargestellt.

```
1 logging.basicConfig(filename='app.log', level=logging.INFO)
2
3 db_host, db_port, db_name, db_user, db_password = get_database_config('config.
    ini')
4
```

```
conn_pool = pool.SimpleConnectionPool(
    0,
    8,
    host=db_host,
    port=db_port,
    database=db_name,
    user=db_user,
    password=db_password,
)

success = False
while not success:
    success = create_tables(conn_pool)
    if not success:
        sleep(10)

success = False
while not success:
    success = create_standard_anomalie(conn_pool)
    if not success:
        sleep(10)
```

Listing 4.36: Verbindungsaufbau zur Datenbank

4.3.3 Einfügen in die Datenbank

singe inserts

Fügt einzelne Datensätze in Tabellen ein, beispielsweise, wenn ein neuer PC mithilfe eines API Endpoints erstellt wird.

Eine Beispielfunktion mit dem passenden Exception Handling wird in dem Listing 4.37 demonstriert.

```
def add_pc(user_id, hardware_uuid, client_name):
    conn = conn_pool.getconn()
    cursor = conn.cursor()
    try:
```

```
        query = "INSERT INTO PC (USER_ID, hardware_uuid, client_name) VALUES (%s, %s, %s) RETURNING ID;"
        params = (str(user_id), str(hardware_uuid), str(client_name))

        pc_id = -1
        cursor.execute(query, params)
        pc_id = cursor.fetchone()[0]
        print("Insertion successful. PC ID:", pc_id)

        conn.commit()

        return pc_id
    except psycopg2.DatabaseError as e:
        conn.rollback()
        if e.pgcode == psycopg2.errorcodes.FOREIGN_KEY_VIOLATION:
            raise NotFoundException(detail="User not found.")
        else:
            raise DataBaseException()
    except Exception as e:
        conn.rollback()
        raise e
    finally:
        conn_pool.putconn(conn)
```

Listing 4.37: Einfügen eines PCs

Mass Inserts

Durch Mass Inserts kann auf eine effiziente Art und Weise eine große Menge an Datensätzen in die Datenbank eingefügt werden. Hierbei wird anstelle der 'cursor.execute' Methode die 'psycopg2.extras.execute_values' Methode aufgerufen.
Diese Art des Einfügens wird beispielsweise beim Einfügen von mehreren Netzwerkdatensätzen, selbst definierten Benachrichtigungen (wie in der Beispielfunktion in Listing 4.38) oder ähnlichem verwendet.

```
def ingestCustomAlerts(alert: CustomAlert):
    conn = conn_pool.getconn()
```

```
    cursor = conn.cursor()
    try:
        insert_query = """
            INSERT INTO anomaly (user_id, type, severity_level, message, condition)
            VALUES %s RETURNING id
        """
        alert_tuples = []

        conditions_json = json.dumps([condition.dict() for condition in alert.conditions])
        alert_tuples.append((alert.user_id, alert.type, alert.severity_level, alert.message, conditions_json))
        psycopg2.extras.execute_values(cursor, insert_query, alert_tuples)

        anomaly_id = cursor.fetchone()[0]

        conn.commit()
        return anomaly_id
    except Exception as e:
        print(str(e))
    finally:
        conn_pool.putconn(conn)
```

Listing 4.38: Masseneinfügen von selbst definierten Benachrichtigungen

4.3.4 Auslesen aus der Datenbank

queries

Abfragen (queries) werden in dieser Arbeit äußerst häufig verwendet, weil die gespeicherten Daten für jede Analyse abgefragt werden müssen. Außerdem werden häufig die Daten bereits bei der Abfrage transformiert, beispielsweise werden die Daten mit der sogenannten 'bucket' Funktion in gewissen Zeitabständen zusammengefügt (siehe Listing 4.39).

```
def get_ram_time_series_between(pc_id, start, end, bucket_value: str = '1 minutes'):
    conn = conn_pool.getconn()
```

```
    cursor = conn.cursor()
    try:
        query = """
        SELECT
        time_bucket(%s, measurement_time) AS bucket_time,
        AVG(ram) as value
    FROM
        pcdata
    WHERE
        pc_id = %s AND
        measurement_time BETWEEN %s AND %s
    GROUP BY
        bucket_time
    ORDER BY
        bucket_time;
        """

        cursor.execute(query, (bucket_value, pc_id, start, end))
        result = cursor.fetchall()

        if result:
            columns = [desc[0] for desc in cursor.description]
            df = pd.DataFrame(result, columns=columns)
            df = df.rename(columns={'bucket_time': 'measurement_time'})
            df['value'] = df['value'].astype(float)
            data_list = []
            for _, row in df.iterrows():
                data_list.append(PCTimeSeriesData(**row.to_dict()))
            return df, data_list
        return None, None
    except psycopg2.DatabaseError as e:
        raise DataBaseException()
    finally:
        conn_pool.putconn(conn)
```

Listing 4.39: Abfragen der RAM Werte, in Zeitabständen zusammengefügt

4.3.5 Löschen aus der Datenbank

delete

In der API ist es nur möglich, selbst definierte Benachrichtigungen wirklich aus der Datenbank zu löschen. Wenn man einen PC löscht, wird bei diesem nur die ID auf NULL gesetzt, die PC Daten selbst werden nicht mithilfe der API gelöscht. Die Funktion zur Löschung von Benachrichtigungen ist im Listing 4.40 dargestellt.

```
def deleteCustomAlerts(alert_id: int):
    conn = conn_pool.getconn()
    cursor = conn.cursor()
    try:
        query = """
            DELETE FROM anomaly WHERE id = %s
        """
        cursor.execute(query, (alert_id,))
        conn.commit()
        return True, "Delete successful!"
    except psycopg2.DatabaseError as e:
        if e.pgcode == "23503":
            return False, "The specified ID was not found."
        else:
            return False, "An error occurred:"+ e.__str__()

    finally:
        conn_pool.putconn(conn)
```

Listing 4.40: Löschen einer selbst definierten Benachrichtigung

Um Speicherplatz auf dem Server zu sparen, wäre es eine Möglichkeit, eine ’Drop Chunk Policy’[70] für die Tabellen zu benutzen, wobei man konfigurieren kann, dass Chunks nach einer bestimmten Zeit automatisch gelöscht werden. Dies wurde in der Arbeit nicht umgesetzt.

4.3.6 Updaten der Datenbank

update

Die Datensätze werden nur aktualisiert, wenn sich Daten ändern. Dies wird, wie in dem Listing 4.41 dargestellt, beim Initial Data Insert ausgeführt. Ansonsten sind keine API Calls für Updates definiert.

```
def update_pc_description(pc_id, client_df):
    conn = conn_pool.getconn()
    cursor = conn.cursor()
    try:
        row = client_df.iloc[0]
        manufacturer = row['computerManufacturer']
        model = row['computerModel']
        update_pc = """
        UPDATE PC
        SET manufacturer = %s,
        model = %s
        WHERE ID = %s;
        """
        cursor.execute(update_pc, (manufacturer, model, pc_id))
        conn.commit()

        return True
    except Exception as e:
        print(f"Error updating PC description: {e}")
        return False
    finally:
        conn_pool.putconn(conn)
```

Listing 4.41: Aktualisieren der PC Beschreibung

4.4 Schnittstellen

Bei den Schnittstellen handelt es sich um ein auf dem Server gehostetes Python Programm, welches mithilfe des Fast Api Frameworks als API fungiert und somit die jeweiligen Komponenten verbindet. Die Schnittstellen werden verwendet, um die gemessenen Daten an den Server zu senden und dort auf der Datenbank zu persistieren. Die Daten werden dann in weiterer Folge von der Benutzeroberfläche angefordert, analysiert und dargestellt. In den nachfolgenden Abschnitten wird die konkrete Implementierung der Schnittstellen beschrieben.

4.4.1 Allgemeines

Um die erfassten Daten zu der Datenbank und weiter zur Visualisierung zu transportieren, wird eine Schnittstelle benötigt. Für die Kommunikation im Internet bietet sich dabei die REST Architektur an, welche in dieser Arbeit auch gewählt wurde.

4.4.2 Umsetzung

Der erste Prototyp der Python API wurde mithilfe von Flask umgesetzt. Die Implementierung der Open API Dokumentation erwies sich in Praxis jedoch als problematisch, weswegen dann auf das Fast API Framework gewechselt wurde. Die Open API Dokumentation wird für eine einfache Anbindung von Clients benötigt.

GET

GET Anfragen werden verwendet, um Daten vom Server anzufordern. In dieser Arbeit werden sie dazu verwendet, um die Daten für Graphen und die analysierten Daten anzufragen.
Im Listing 4.42 wird dargestellt, wie eine Funktion zum Anfordern der PC Beschreibung als Endpoint mit Fast API definiert wird.

```
1 @pc.get('/details/{pc_id}', response_model=PCDetails, tags=["PC"])
```

```
2 def get_pc_detail_by_pc_id(pc_id: str):
```

Listing 4.42: Funktion zur Anforderung der PC-Beschreibung

POST

Durch POST Anfragen werden Daten an den Server gesendet. In der Arbeit werden sie verwendet, um die laufenden PC Daten an den Server zu übermitteln und PCs (siehe Listing 4.43) sowie auch andere Entitäten hinzuzufügen.

```
1 @pc.post('/add_pc', response_model=dict, status_code=201, tags=["PC"])
2 def add_pc_api(data: PCItem = Body(...)):
```

Listing 4.43: Funktion zum Hinzufügen eines PCs

PUT

PUT Anfragen werden dafür verwendet, um Daten zu aktualisieren. Diese Funktion wird jedoch in dieser Arbeit nicht verwendet.

DELETE

DELETE Anfragen werden zur Anforderung der Löschung von Entitäten verwendet. Diese Art von Anfragen werden in der Arbeit zum Löschen von PCs (siehe Listing 4.44) und Anomalien verwendet.

```
1 @pc.delete('/{pc_id}/', response_model=dict, tags=["PC"])
2 def delete_pc(pc_id: int):
```

Listing 4.44: Funktion zum Löschen eines PCs

Dokumentation

Die Dokumentation wird bei Fast API mit der Hilfe von Kommentaren erledigt, was eine saubere Umsetzung äußerst simpel gestaltet. Ein Beispiel für einen Kommentar zur Dokumentation kann im Listing 4.45 betrachtet werden.

```
1 @pc.get('/', response_model=dict, tags=["PC"])
2 def get_all_pcs():
```

```
    """
    Get all PCs.

    Returns:
        dict: A dictionary with a 'pcs' key containing a list of PCs.
    """
    return {'pcs': get_pcs()}
```

Listing 4.45: Beispiel für Dokumentation in Form eines Kommentars

Funktionskontrolle

Um die Funktionalität der API Calls manuell zu überprüfen, wurde der Pycharm HTTP Client verwendet.

GET Tests für API Aufrufe, welche Daten der API abrufen, wie im Listing 4.46 demonstriert.

```
GET http://localhost:8000/pc/
```

Listing 4.46: Aufruf zur abfrage aller PCs

POST

JSON Tests für API Aufrufe, welchen ein JSON übergeben wird. Ein Beispiel hierfür kann man im Listing 4.47 einsehen.

```
POST http://localhost:8000/pc/add_pc
Content-Type: application/json

{
  "user_id": "1",
  "hardware_uuid": "b1e6904d-c68b-44a9-a4e1-725b36903eac",
  "client_name": "thomas"
}
```

Listing 4.47: API Aufruf zum Hinzufügen eines PCs

multipart/form-data Tests für API Aufrufe, welchen mehrere Dateien übergeben werden, beispielsweise beim Einfügen der initialen PC Daten (siehe Listing 4.48).

```
POST http://localhost:8000/data/initial
Content-Type: multipart/form-data; boundary=----FormBoundary

------FormBoundary
Content-Disposition: form-data; name="files"; filename="client.csv"
Content-Type: text/csv

measurement_time,computerHardwareUUID,computerManufacturer,
   computerModel,memoryTotalSize,memoryPageSize,processorName,
   processorIdentifier,processorID,processorVendor,processorBitness,
   physicalPackageCount,physicalProcessorCount,logicalProcessorCount
<example data>

------FormBoundary
Content-Disposition: form-data; name="files"; filename="disk.csv"
Content-Type: text/csv

measurement_time,serialNumber,model,name,size
<example data>

------FormBoundary
Content-Disposition: form-data; name="files"; filename="partition.csv"
Content-Type: text/csv

diskStoreName,identification,name,type,mountPoint,size,majorFaults,
   minorFaults
<example data>

------FormBoundary--
```

Listing 4.48: API Aufruf zum Hinzufügen von Daten

DELETE Tests für API Aufrufe, welchen löschen, wie in dem Listing 4.49 zu erkennen.

```
1 DELETE http://localhost:8000/pc/1
```

Listing 4.49: Löschen des PCs mit der ID 1

4.4.3 Endpoints

Endpunkte, genauer API Serverendpunkte[42], sind der Ort, an dem Client Anfragen erfüllt werden.
Diese Endpunkte werden in der Arbeit in User, Benachrichtigung, PC und Data Endpunkte unterteilt. Man kann auf diese Endpunkte mit der URL des Rechners, auf dem die API läuft, zugreifen. Hierfür muss man zuerst die URL und dann den Pfad des Endpunktes angeben. Die URL des Diplomarbeitsservers lautet ”logSense.htl-perg.ac.at”.

User

Endpunkte, welche sich mit dem Erstellen, Abfragen oder anderen Operationen der User beschäftigen.

- **Abfragen aller Benutzer**

  GET /user/
  Bei dieser Abfrage werden alle existierenden Benutzer abgefragt.

- **Benutzer hinzufügen**

  POST /user/add_user
  Bei diesem Aufruf wird ein Benutzer hinzugefügt.

Benachrichtigung

Endpunkte, welche sich mit dem Erstellen, Abfragen oder anderen Operationen, die sich mit Benachrichtigungen beschäftigen.

- **alle Benachrichtigungen eines Benutzers**

  GET /alerts/{user_id}
  Abfragen aller Benachrichtigungen, welche in einem bestimmten Zeitrahmen bei einem Benutzer passiert sind.

- **Begründung einer Benachrichtigung**

  GET /alerts/justify/{pc_id}
  Abfragen der Begründungen, also warum eine Benachrichtigung passiert ist.

- **Alle Custom Benachrichtigungen**

  GET /alerts/all/{user_id}
  Abfrage aller benutzerdefinierten Benachrichtigungen.

- **Benachrichtigung hinzufügen**

  POST /alerts/
  Hinzufügen einer benutzerdefinierten Benachrichtigung.

- **Benachrichtigung löschen**

  DELETE /alerts/{alert_id}
  Löschen einer benutzerdefinierten Benachrichtigung.

PC

Endpunkte, welche sich mit dem Erstellen, Abfragen oder anderen Operationen der PCs beschäftigen.

- **Alle PCs**

  GET /pc/
  Abfragen aller PCs

- **PCs eines Benutzers**

  GET /pc/user/{user_id}
  Abfragen aller PCs, welche zu einem Benutzer gehören

- **PC hinzufügen**

  POST /pc/add_pc
  hinzufügen eines PCs

- **RAM Daten**

  GET /pc/{pc_id}/ram
  Abfrage der analysierten RAM Daten eines PCs

- **CPU Daten**

  GET /pc/{pc_id}/cpu
  Abfrage der analysierten CPU Daten eines PCs

- **Festplatten Daten**

  GET /pc/{pc_id}/disk
  Abfrage der Festplattendaten eines PCs

- **Daten über die Partitionen**

  GET /pc/{pc_id}/disk-partition
  Abfrage der Partitionsdaten eines PCs

- **Netzwerk Daten**

  GET /pc/{pc_id}/network
  Abfrage der Netzwerkdaten eines PCs

- **Festplattendaten mit Prognose**

  GET /pc/{pc_id}/data/foryecast/{days}
  Abfragen der Festplattendaten, inklusive Daten, welche eine gewisse Anzahl an Tagen in die Zukunft projiziert werden.

- **generelle Daten**

  GET /pc/general_specs/{pc_id}
  Abfragen der generellen Daten über den PC, wie beispielsweise den Prozessor oder die Anzahl der Bit des Prozessors

- **Details**

  GET /pc/details/{pc_id}
  Abfrage der Details, wie beispielsweise, ob der PC gerade lädt oder der Hersteller des PCs

- **Ressourcen Daten**

  GET /pc/resource_metrics/{pc_id}
  Abfrage der Ressourcendaten, wie beispielsweise die gesamte Menge an RAM oder die RAM Page Größe

- **Nutzungsdauer**

  GET /pc/{pc_id}/time_metrics
  Abfrage der Nutzungsdauer des PCs

- **PC löschen**

  DELETE /pc/{pc_id}
  löschen eines PCs, definiert mit seiner ID

Application

Endpunkte, welche sich mit Daten zu spezifischen Applikationen beschäftigen.

- **Applikations-Daten**

  GET /pc/{pc_id}/application/{application_name}
  Abfrage aller Daten zu einer spezifischen Applikation.

- **Applikations-Daten Zusammengefasst**

  GET /pc/{pc_id}/application/{application_name}/bucket
  Abfrage aller Daten zu einer spezifischen Applikation, welche jedoch in einem definierten Zeitraum zusammengefasst werden.

- **Alle Applikationen**

  GET /pc/{pc_id}/application
  Abfrage einer Liste aller Applikationen.

- **Laufzeit der Applikation**

  GET /pc/{pc_id}/application/{application_name}/time-metrics
  Abfrage der Laufzeit einer Applikation in einem bestimmten Zeitrahmen.

Data

Daten des PCs werden vom Agent an die API gesendet.

- **initiale Daten**

  POST /data/initial
  Hinzufügen der initialen Daten, welche generelle Werte über den PC beinhalten.

- **laufende Daten**

  POST /data/
  Hinzufügen der laufenden Daten, welche die Performancedaten des PCs beinhalten.

4.5 Server

4.5.1 Allgemeines

Sowohl das Frontend, als auch das Backend wird auf einem Ubuntu Server ausgeführt, welcher in einer virtuellen Maschine auf den Servern der HTL Perg läuft. Auf dem Server läuft ein Docker, welcher die Timescale Datenbank zur Verfügung stellt. Und auch ein nginx Webserver, welcher den Sinn hat, die Python API und die Angular Applikation zu hosten.
Die Version des Betriebssystems ist Ubuntu Server 22.04 LTS. Ubuntu wurde unter anderem wegen der Stabilität und Zuverlässigkeit gewählt. Außerdem ist Ubuntu kostenlos und Open-Source.
Um auf den Server zuzugreifen, wurde das SSH Kommando in der Kommandozeile verwendet.

4.5.2 Webserver

Webserver[26] sind die Plattformen, auf denen Webinhalte gespeichert werden und von denen aus sie jederzeit für die Nutzer abrufbar sind. Im Moment des Aufrufs einer Internetadresse im Webbrowser übernimmt der Webserver die Aufgabe, die verschiedenen Bestandteile der Website an den Computer zu senden. Die kontinuierliche Verfügbarkeit einer Website erfordert, dass der Webserver rund um die Uhr online ist.
In dieser Arbeit wird ein Webserver verwendet, um die Python API und Angular Application zu hosten. Bei dem verwendeten Webserver handelt es sich um nginx, da dieser äußerst stabil und performant ist.

4.6 Visualisierung

Die folgenden Abschnitte beschreiben die konkrete Implementierung der Webanwendung.

4.6.1 Benutzeroberfläche

Die Benutzeroberfläche besteht aus einigen Komponenten. Zwei Komponenten werden durchgehend eingeblendet. Der Header und die Toolbar. Im Header wird das eigens entworfene Logo von LogSense angezeigt. Die Toolbar ist für die Navigation durch die Webanwendung zuständig und wird im nächsten Abschnitt noch näher beschrieben. Weiteres kann zwischen den Ansichten für den Ressourcenverbrauch, einem generellen Überblick, Statistiken einzelner Prozesse, Benutzerdefinierte-Warnungen und der PC-Auswahl gewechselt werden.

Prinzipiell ist der Aufbau der Webanwendung durch die HTML-Dokumente definiert. Um dynamische Daten anzuzeigen, werden die Variablen aus der TypeScript-Datei mit folgender Syntax verwendet, die im Listing 4.50 zu sehen ist.

```
1     <h3>{{cpuGeneral.processor_name}}</h3>
```

Listing 4.50: Syntax für die Verwendung einer TypeScript-Variable in einem HTML-Dokument

Einige Elemente werden nicht dauerhaft eingeblendet. Um einzelne Komponenten oder HTML-Elemente abhängig von dynamischen Daten ein- und auszublenden, kann die Attributdirektive *ngIf verwendet werden, wie im Listing 4.51 beispielhaft dargestellt ist.

```
1 <span *ngIf="process.allocation>0" class="txt-16">{{process.allocation}}%</
     span>
```

Listing 4.51: Verwendung von *ngIf

Das Listing 4.52 zeigt, wie mehrere Daten dynamisch mithilfe der Strukturdirektive *ngFor angezeigt werden. *ngFor durchläuft einen, im TypeScript angelegten, Array und zeigt die Daten nach der Reihe, wie im HTML-Dokument definiert, an. Das Schema eignet sich ideal, um Daten in Form von Tabellen aufzulisten, wie bei der Auflistung der Speichermedien oder auch der Netzwerkverbindungen.

```
<tr *ngFor="let diskStore of diskInfo.disks; let i = index">
    <td class="cell txt-16">{{diskStore.serialnumber}}</td>
    <td class="cell txt-16">{{diskStore.name}}</td>
    <td class="cell txt-16">{{diskStore.model}}</td>
    <td class="cell txt-16">
    {{roundDecimalNumber(convertBytesToGigaBytes(diskStore.size), 3)}} GB</td>
    <td (click)="openDialog(i)" class="cell txt-16">
        <span class="link">See Partitions</span>
    </td>
</tr>
```

Listing 4.52: Verwendung von *ngFor

Eines der Schlüsselkonzepte von Angular ist die Wiederverwendung von Komponenten. Die Benachrichtigungs-Komponente ist einmalig definiert und wird in den einzelnen Ansichten, sollten Warnung vorhanden sein, angezeigt. Um die Warnungen passend zur aktuellen Seite anzuzeigen, müssen diese an die Alert-Komponente übergeben werden. Die Übergabe von Werte in eine andere Komponente ist im Listing 4.54 beispielhaft dargestellt. Wird im TypeScript der Array definiert und mit @Input() markiert, wie im Listing 4.53 zusehen ist, so wird im HTML-Dokument der Alert-Array einfach übergeben.

```
    @Input() alerts: Alert[] = [];
```

Listing 4.53: Ausschnitt aus der TypeScript-Datei der Benachrichtigungs-Komponente

```
    <app-alerts [alerts]="alerts"></app-alerts>
```

Listing 4.54: Einbindung der Benachrichtigungs-Komponente

Um auf Benutzerinteraktionen zu reagieren, wird das Ereignis (click) verwendet, wie im Listing 4.55 zusehen ist. Diese Funktion kann auf jedes beliebige Element angewandt werden, nicht nur auf Buttons und Links. Wird vom Benutzer das jeweilige Element angeklickt, so wird die, im TypeScript definierte, Methode aufgerufen.

```
<span (click)="showAll()" *ngIf="!showAllProcesses">Show All</span>
```

Listing 4.55: (click)-Ereignis

Eine weitere Möglichkeit für Benutzerinteraktionen sind Dropdowns. Um die Zeitspanne auszuwählen, werden Dropdowns verwendet. Mit einer Kombination aus einer *ngFor-Schleife und dem [ngValue]-Tag, die im Listing 4.56 zu sehen ist, werden dynamische Daten im Dropdown angezeigt. [(ngModel)] ermöglicht es, den aktuell ausgewählten Wert in der TypeScript-Datei einzusehen. Mit der Funktion (change) werden die Diagramme bei einer Änderung an die neue Benutzereingabe angepasst.

```
<select [(ngModel)]="selectedTime" class="dropdown-btn" (change)="loadStats();
    loadData();">
    <option *ngFor="let elem of times" [ngValue]="elem">{{elem.time}}</option>
</select>
```

Listing 4.56: Dropdown

Außerdem gibt es noch die Möglichkeit, eine Benutzereingabe mittels Radio-Buttons zu steuern. Hierfür wird eine Angular Material Komponente verwendet, die an dem 'mat'-Tag identifiziert wird. Die konkrete Verwendung wird im Listing 4.57 beispielhaft dargestellt.

```
<mat-radio-group aria-labelledby="radio-group-label" class="radio-group"
[(ngModel)]="checked" (change)="reloadChart()">
    <mat-radio-button class="radio-button" *ngFor="let option of radioOptions"
    [value]="option">
        {{option}}
    </mat-radio-button>
</mat-radio-group>
```

Listing 4.57: Radio Buttons

Eine weitere Angular Material Komponente, die in dieser Diplomarbeit Verwendung findet, ist ein Dialog. Ein Dialog ist ein kleineres Fenster, das über der derzeitigen Ansicht temporär angezeigt wird. Konkret bindet man den Dialog ein, indem eine eigene Angular-Komponente erstellt wird, in der das Aussehen und der Inhalt des Fensters definiert wird. Der Dialog kann aufgerufen werden, indem in jeweiligen Komponente eine TypeScript-Funktion ausgelöst wird, die für den Aufruf zuständig ist. Diese Funktion ist im Listing 4.58 zu sehen. Beim Aufruf des Dialogs gibt es auch die Möglichkeit Daten und den Dialog zu schicken, die dort angezeigt werden sollen.

```
openDialog(diskListIndex: number) {
    this.dialog.open(PartDialogComponent, {
      data: this.diskInfo.disks.at(diskListIndex)!.partitions
    });
}
```

Listing 4.58: Angular Material Dialog

4.6.2 Toolbar

Die Toolbar repräsentiert eine Navigationsleiste mit Icons, die auf verschiedene Routen innerhalb der Anwendung verlinken. Jedes Icon ist in einem Container mit einem Router-Link eingebettet, um die Navigation zu ermöglichen. Die Icons der gesamten Webanwendung werden von Angular Material zur Verfügung gestellt und werden, wie im Listing 4.59 zu sehen ist, eingebunden:

```
<mat-icon class="icon-32 icon">timeline</mat-icon>
```

Listing 4.59: Angular Material Icon

Die Pfade werden im äpp-routing.module.ts"definiert, wie im Listing 4.60 zu sehen ist. Beim Anklicken der Icons in der Toolbar werden die Pfade aufgerufen und die jeweilige Komponente wird angezeigt.

```
const routes: Routes = [
  {path: '', component: OverviewComponent},
  {path: 'overview', component: OverviewComponent},
  {path: 'cpu', component: CpuComponent},
  {path: 'processes', component: SingleProcessesComponent},
  {path: 'ram', component: RamComponent},
  {path: 'disk', component: DiskComponent},
  {path: 'network', component: NetworkComponent},
  {path: 'alerts', component: CustomAlertsComponent},
  {path: 'pc_selection', component: PcSelectionComponent}
];
```

Listing 4.60: Definition der Routen im routing.module

In der ”main-page.component.html”wird, abgesehen von der Toolbar und dem Header, die dauerhaft eingeblendet sind, ein Router-Outlet eingebunden, wie im Listing 4.61 zu sehen ist. Dieses dient als Platzhalter für die, in der Toolbar, aufgerufenen Komponenten. Beim Aktiveren eines Router-Links wird die Komponente von Angular injiziert und angezeigt.

```
<router-outlet></router-outlet>
<app-header></app-header>
<app-toolbar></app-toolbar>
```

Listing 4.61: main-page.component.html

4.6.3 Responsives Design

Für das konkrete Layout der Webseite wird Flexbox verwendet. Der CSS-Code im Listing 4.62 definiert ein flexibles Layout mit einem Container, einem Element mit 45% Breite und einer Gruppe von zwei Elementen, von denen jedes 50% Breite hat. Der Container hat eine Gesamtbreite von 100% abzüglich 64px. Dieses Layout ist darauf ausgerichtet, die verfügbare Breite effizient zu nutzen und eine responsive Struktur zu schaffen.

```
.container {
  display: flex;
  flex-direction: row;
  width: calc(100% - 64px);
  .first {
    flex: 0 0 45%;
    width: 100%;
  }
  .group {
    display: flex;
    flex-direction: row;
    flex: 0 0 55%;
    .second {
      flex: 0 0 50%;
      width: 100%;
    }
    .third {
      flex: 0 0 50%;
      width: 100%;
```

```
    }
  }
}
```

Listing 4.62: Flexbox

Der SASS-Code im Listing 4.63 definiert eine Media Query für Bildschirme mit einer maximalen Breite von ”lgöder kleiner. Das bedeutet, dass die darin enthaltenen Stilregeln nur auf Bildschirme mit einer Breite kleiner oder gleich der ”lgBreite angewendet werden und den CSS-Code für eine größere Bildschirmbreite überschreibt. Die Bildschirmgrößen werden von Bootstrap, wie in der Tabelle 4.13 angeführt, definiert:

Small	sm	≥576px
Medium	md	≥768px
Large	lg	≥992px
Extra large xl	xl	≥1200px
Extra extra large	xxl	≥1400px

Tabelle 4.13: Bootstrap default breakpoints [3]

Wird die Bildschirmbreite so weit verringert, dass der CSS-Code im Listing 4.63 verwendet wird, werden die Elemente auf der Webseite neu angeordnet und optisch an die kleinere Fläche angepasst.

```
@include media-breakpoint-down(lg) {
  .container .group {
    width: 100% !important;
    flex-direction: column;
    .second {
      margin: 0 0 24px 0;
    }
    .third {
      margin: 0;
    }
  }
}
```

Listing 4.63: Media Query für responsives Design

4.6.4 Hover-Effekte

Um Details zu einzelnen Prozessen oder dem Gerät anzuzeigen, wird ein kleines Info-Icon verwendet. Das Icon wird, wie alle anderen Icons, von Angular Material zur Verfügung gestellt. Wird der Mauszeiger über das Icon bewegt, so werden die Details sichtbar. Konkret wird hierbei mithilfe von CSS die Information ausgeblendet, indem ’display’ auf ’none’ gesetzt wird, wie dem CSS-Code im Listing 4.64 zu entnehmen ist. Wird ’:hover’ aktiviert, so wird ’display’ für diese Dauer auf ’block’ gesetzt, um die Informationen sichtbar zu machen.

```
.icon-hov:hover + .hidden {
  display: block;
}
.hidden {
  display: none;
  position: absolute;
  transform: translate(48px, 136px);
  padding: 8px 16px;
  width: max-content;
  .lower {
    white-space: normal;
  }
}
```

Listing 4.64: CSS-Coder für einen Hover-Effekt

4.6.5 Kommunikation mit der API

In Angular wird die Kommunikation mit einer API durch das HTTP-Modul, das von Angular bereitgestellt wird, ermöglicht. Dieses Modul bietet Dienste, um HTTP-Anfragen zu senden und zu empfangen. Der Prozess beginnt mit der Erstellung von Service-Komponenten, die die API-Aufrufe koordinieren und verwalten. Ein Ausschnitt der PC-Service-Komponente ist im Listing 4.65 beispielhaft angeführt. Zunächst wird ein HTTP-Client erstellt, der die notwendigen Methoden für HTTP-Anfragen bereitstellt, wie:

- GET
- POST
- PUT
- DELETE

Um eine Anfrage an die API zu senden, wird die entsprechende Methode des HTTP-Services aufgerufen, und dieser kümmert sich um den Versand der Anfrage an die angegebene URL. Die Antwort der API wird als Observable behandelt, was es ermöglicht, asynchrone Vorgänge zu verwalten. Der angegebene Datentyp muss an das JSON Schema der API angepasst sein.

```
export class PcService {
    private url: string = "http://localhost:8000/pc";

    constructor(private httpClient: HttpClient) {
    }

    getPCsOfUser(user_id: number): Observable<PCs> {
        return this.httpClient.get<PCs>(this.url + `/user/${user_id}`);
    }

    addPCToUser(userPC: UserPC): Observable<any> {
        return this.httpClient.post<any>(this.url + "/add_pc", userPC);
    }
}
```

Listing 4.65: Ausschnitt aus der PC-Service-Komponente

Im TypeScript-Code im Listing 4.65 werden die Geräte eines Benutzers von der API angefragt und neue Geräte hinzugefügt. Für das Anfragen der Rechner wird eine GET-Methode verwendet. Um den Benutzer zu identifizieren, wird die Benutzer-ID übergeben. Die PCs-Klasse ist an den JSON-String, der von der API zurückgegeben wird, angepasst. Bei der POST-Methode wird nicht nur eine einzelne ID übergeben, sondern ein ganzes Objekt.

Dieses wird in die Datenbank gespeichert und wird mit den restlichen Geräten des Nutzers beim nächsten Mal geladen.
Dieser Service wird in den Komponenten verwendet, die die Antworten der API verwerten. Die erhaltenen Daten werden in den Komponenten weiterverarbeitet und in der Benutzeroberfläche dargestellt. Angulars Datenbindungsfunktionen erleichtern das Aktualisieren der Ansicht, sobald die Daten von der API zurückkommen.

Im Listing 4.66 ist die *loadData()*-Methode zusehen, in der die Auslastungsdaten der CPU geladen werden. Abhängig von der Zeitspanne, die der Benutzer auswählt und dem zeitlichen Abstand zwischen den einzelnen Messwerten wird mit dem Service eine Anfrage an die API geschickt. Der Rückgabewert ist in diesem Fall ein CPUModel-Objekt. In diesem sind sämtliche Auslastungsdaten, sowie Anomalien, Ereignissen und Statistiken enthalten. Diese Informationen werden hier transformiert und für die Anzeige präpariert.

```
1    loadData() {
2    let dateNow = Date.now();
3    if (this.selectedTime.valueInMilliseconds != 0) {
4      this.pcDataService.getCPUData(this.pcId, this.datePipe.transform(dateNow
     -
5      this.selectedTime.valueInMilliseconds, 'yyyy-MM-ddTHH:mm') ?? "",
6      this.datePipe.transform(dateNow, "yyyy-MM-ddTHH:mm") ?? "",
7      this.selectedBucketingTime.value).subscribe((data: CPUModel) => {
8        this.cpu = data;
9        this.notes = data.statistic_data.message.split("\n");
10       this.transformData();
11       this.showAll();
12       this.reloadChart();
13     });
14   } else {
15     this.pcDataService.getCPUData(this.pcId,
16     this.datePipe.transform(dateNow - dateNow, 'yyyy-MM-ddTHH:mm') ?? "",
17     this.datePipe.transform(dateNow, "yyyy-MM-ddTHH:mm") ?? "",
18     this.selectedBucketingTime.value).subscribe((data: CPUModel) => {
19       this.cpu = data;
```

```
20          this.notes = data.statistic_data.message.split("\n");
21          this.transformData();
22          this.showAll();
23          this.reloadChart();
24        });
25      }
26    }
```

Listing 4.66: Ausschnitt der cpu.component.ts

4.6.6 Grafik der Zeitaufzeichnung einzelner Prozesse

Der Benutzer kann den Zeitraum der Aufzeichnung auswählen. Abhängig davon werden als Erstes die relevanten Daten der einzelnen Prozesse beim Aufruf der Komponente, wie zuvor beschrieben, über die API geladen. Die Prozessnamen und die dazugehörigen Daten werden jeweils in einem eigenen Array gespeichert und für die Anzeige präpariert. Der Benutzer kann auswählen, ob er die 10 Prozesse mit der längsten Laufzeit anzeigen möchte oder ob er die Laufzeit von allen Prozessen, die in dem ausgewählten Zeitraum aktiv waren, dargestellt haben möchte. Nach dem Speichern der Daten in zwei separate Arrays wird der Graph erstellt. Um die Zeitaufzeichnung zu visualisieren, wird die Chart.js Bibliothek verwendet. In der TypeScript-Datei kann eine Vielzahl an Konfigurationen vorgenommen werden, um das Diagramm seinen eigenen Wünschen entsprechend anzupassen, wie im Listing 4.67 zu sehen ist.

```
1     this.timeMetricsChart = new Chart("timeChart", {
2       type: 'bar',
3       data: {
4         labels: this.timeMetrics.name,
5         datasets: [{
6           data: (this.timeMetrics.total_running_time_minutes),
7           borderColor: "#3e95cd",
8           backgroundColor: "rgba(62,149,205, 0.4)",
9           borderWidth: 1
10        }]
11      },
12      options: {
```

```
13        plugins: {
14          legend: {
15            display: false
16          },
17          tooltip: {
18            callbacks: {
19              label: function (context) {
20                let label = context.dataset.label || '';
21                if (label) {
22                  label += ' ';
23                }
24                label += context.parsed.y + ' hrs';
25                return label;
26              }
27            }
28          }
29        },
30        scales: {
31          y: {
32            beginAtZero: true,
33          },
34        },
35      },
36    });
```

Listing 4.67: Zeitaufzeichnungsdiagramm mit ChartJS

Mit dem ’type’-Attribut kann der Diagrammtyp festgelegt werden. Speziell für das Diagramm für die Zeitaufzeichnung der Prozesse wird ein Balkendiagramm verwendet, wie in der obigen Abbildung zusehen ist. Ein Array, der mit den Namen der aufgezeichneten Prozesse gefüllt ist, wird als ’labels’ angegeben. Die Labels bezeichnen die einzelnen Datenpunkte auf der x-Achse. Die dazugehörigen Werte werden in ’datasets’ unter ’data’ definiert. Der Array mit der Laufzeit der aufgezeichneten Prozesse muss mit den Labels übereinstimmen. Da es die Möglichkeit gibt, mehrere Graphen in einem Diagramm anzuzeigen, wird bei dem Dataset direkt das Aussehen, wie die Farbe und die Breite, des Graphen festgelegt. Zusätzlich gibt es noch zahlreiche weitere Konfigurationen. Für diesen

konkreten Fall wird jedoch nur die Legende ausgeblendet und der Start der y-Achse auf null gesetzt. Die Tooltips werden im Abschnitt 4.6.10.Details zu einzelnen Datenpunkten erklärt. Das Diagramm wird, wie im Listing 4.68 beispielhaft dargestellt ist, in das HTML-Dokument eingebunden:

```
1    <canvas class="white-bg w-100" id="timeChart"></canvas>
```

Listing 4.68: Einbindung des ChartJS-Diagramms

Über die ID wird es eindeutig identifiziert. Um das Diagramm zu aktualisieren, muss es erst zerstört werden. Danach kann es mit neuen Daten wieder erstellt werden.

4.6.7 Zeitseriendaten grafisch darstellen

Ähnlich wie bei der Zeitaufzeichnung wird für die Visualisierung der Auslastung der einzelnen Hardwarekomponenten die Chart.js Bibliothek verwendet. Zusätzlich zum Zeitraum der Aufzeichnung kann hier noch der Zeitraum zwischen den Datenpunkten ausgewählt werden. Beim Laden der Komponente oder bei einer Änderung der Zeitangaben werden die Daten, wie im Abschnitt 4.6.5 Kommunikation mit der API beschrieben, geladen. Nach dem Laden werden die Daten noch präpariert, indem die Zeiten in das gewünschte Format transformiert werden, wie im Listing 4.69 beispielhaft dargestellt ist.

```
1    this.datePipe.transform(dataPoint.measurement_time, 'MM-dd HH:mm')
```

Listing 4.69: Umwandlung der Daten

Dafür wird DatePipe verwendet. DatePipe wird von Angular für den Umgang mit Daten und Zeiten zur Verfügung gestellt und ermöglicht es, die Zeitstempel der Messungen in ein beliebiges Format umzuwandeln, wie im Listing 4.69 zu sehen ist. Außerdem müssen die Messwerte gerundet und teilweise in die gewünschte Datengröße, wie von Byte zu Gigabyte, umgerechnet werden.

Für die Darstellung der Auslastung wird ein Liniendiagramm verwendet. Die konkrete Konfiguration des Diagramms ist im Listing 4.70 definiert. Als Label werden die Zeitstempel der Messungen verwendet. Die dazugehörigen Werte, die Auslastung zu dem Zeitpunkt, werden unter ’data’ definiert. Dass der Bereich unter dem Graphen nicht ausgemalt wird,

wird ’fill’ auf false gesetzt. Weiteres wird noch die Farbe im ’dataset’ angegeben. Wie auch bei dem Zeitaufzeichnungsdiagramm wird der Start der y-Achse wieder auf null gesetzt und die Legende ausgeblendet. Die Option ’responsiv’ ermöglicht es, dass das Diagramm auf Veränderungen der Bildschirmgröße automatisch reagiert.

```
this.cpuChart = new Chart("usage", {
      type: "line",
      data: {
        labels: this.cpuData.time,
        datasets: [{
          data: this.cpuData.value,
          borderColor: "#3e95cd",
          fill: false
        }]
      }, options: {
        responsive: true,
        scales: {
          y: {
            beginAtZero: true,
          },
        },
        plugins: {
          legend: {
            display: false
          },
          tooltip: {
            callbacks: {
              label: function (context) {
                let label = context.dataset.label || '';
                if (label) {
                  label += ' ';
                }
                label += context.parsed.y + ' %';
                return label;
              }
            }
          }
```

```
        }
      },
    });
```

Listing 4.70: CPU-Auslastungsdiagramm mit ChartJS

Einen kleinen Unterschied gibt es bei der Visualisierung des freien Speichers. ES gibt die Möglichkeit, dem Benutzer eine Vorhersage für die Entwicklung des freien Speichers der nächsten 30 Tage anzuzeigen. Somit werden 2 unterschiedliche Graphen in demselben Diagramm benötigt. Um dies zu ermöglichen, werden zwei Datasets angegeben, wie im Listing 4.71 zu sehen ist. Das erste ist nach wie vor für die Darstellung des gemessenen freien Speichers zuständig. Das zweite ist für die Vorhersage verantwortlich. Damit der Benutzer diese beide Graphen auf den ersten Blick unterscheiden kann, werden unterschiedliche Farben verwendet, die dem jeweiligen Dataset zugeordnet werden. Um die Labels für beide Graphen korrekt zu definieren, wird der Array mit der Beschriftung für die Vorhersage an den Array mit dem Zeitstempel der Messwerte angehängt. Dafür wird die Funktion ’concat’ verwendet, die es ermöglicht Arrays oder auch Strings aneinander zu reihen.

```
    data: {
        datasets:[{
          type: "line",
          data: this.diskChartData.value,
          backgroundColor: 'rgba(62, 149, 205, 0.5)',
          borderColor: '#3e95cd'
        }, {
          type: "line",
          data: this.getForecastData(),
          backgroundColor: '#e82546',
          borderColor: '#e82546'
        }],
        labels: this.diskChartData.time.concat(this.forecastData.time)
      }
```

Listing 4.71: Verwendung mehrerer Datasets

4.6.8 Anomalien anzeigen

Der Benutzer hat die Möglichkeit über Radio-Buttons unter dem Auslastungsdiagramm auszuwählen, dass die, in der Analyse erkannten, Anomalien angezeigt werden. Die *getAnomalies()*-Methode, die im Listing 4.72 zu sehen ist, erstellt den Array, der für das Dataset benötigt wird, um die Punkte im Diagramm zu markieren. Damit die Anomalien mit dem korrekten Zeitstempel angezeigt werden, muss der Array dieselbe Länge haben, wie der Array mit den generellen Auslastungsdaten. Beim Laden der Zeitseriendaten werden, die Anomalien direkt mitgeladen. Somit kann an allen Positionen ohne Anomalien der Array mit null befüllt werden. Ist ein Wert im Array null, so wird im Diagramm nichts an dieser Position angezeigt. Sobald ein Wert als Anomalie gekennzeichnet ist, wird der Wert der Auslastung, der zu dem Zeitpunkt gemessen worden ist, in dem Array gespeichert.

```
1    getAnomalies() {
2    let anomalyPositions: any[] = [];
3    let success: boolean = false;
4
5    this.cpu.time_series_list.forEach((data, index) => {
6      for (let anomaly of this.cpu.events_and_anomalies) {
7        if (data.measurement_time == anomaly.timestamp && anomaly.is_anomaly)
   {
8          anomalyPositions.push(this.cpuData.value[index]);
9          success = true;
10       }
11     }
12     if (!success) {
13       anomalyPositions.push(null);
14     }
15     success = false;
16   })
17   return anomalyPositions;
18 }
```

Listing 4.72: Anomalien-Dataset

Für die Visualisierung dieser beiden Arrays werden zwei unterschiedliche Diagrammarten verwendet. Für den Graphen für die Auslastungsdaten wird, wie bisher, ein Liniendia-

gramm verwendet, wie im Listing 4.73 zu sehen ist. Für das Einzeichnen der Anomalien wird ein Streudiagramm verwendet. Damit das Streudiagramm über dem Liniendiagramm angezeigt wird und die Anomalien erkennbar sind, wird im Dataset der Auslastungswerte 'order' auf 2 gesetzt. Damit die Anomalien für den Benutzer deutlich erkennbar sind, werden die Punkte rot eingezeichnet, während der Graph standardmäßig blau bleibt. Als Labels werden, wie im Abschnitt 4.6.7 Zeitseriendaten grafisch darstellen beschrieben, die Zeitstempel der gemessenen Daten verwendet. Die Methode für das Anzeigen der Tooltips wird im Abschnitt 4.6.10 Details zu einzelnen Datenpunkten beschrieben.

```
this.cpuChart = new Chart("usage", {
  data: {
    datasets: [{
      type: 'line',
      label: 'Line Dataset',
      data: this.cpuData.value,
      backgroundColor: 'rgba(62, 149, 205, 0.5)',
      borderColor: '#3e95cd',
      order: 2
    }, {
      type: 'scatter',
      label: 'Scatter Dataset',
      data: this.getAnomalies(),
      backgroundColor: '#e82546',
      borderColor: '#e82546'
    }],
    labels: this.cpuData.time
  },
  options: {
    responsive: true,
    scales: {
      y: {
        beginAtZero: true,
      },
    },
    plugins: {
      legend: {
```

```
            display: false
          },
          tooltip: {
            callbacks: {
              label: (context) => this.getEventMsg(context)
            }
          }
        }
      }
    })
```

Listing 4.73: CPU-Auslastungsdiagramm inklusive Anomalien

4.6.9 Ereignisse anzeigen

Die Ereignisse werden gemeinsam mit den Anomalien und den Auslastungswerten geladen. Für das Auslastungsdiagramm wird wie zuvor beschrieben ein blaues Liniendiagramm und für das Einzeichnen der Anomalien ein rotes Streudiagramm verwendet. Nachdem Ereignisse über einen längeren Zeitraum und unabhängig voneinander erkannt und markiert werden, besteht die Möglichkeit, dass sie sich überschneiden. Deswegen wird für jedes Ereignis ein eigener Graph definiert, wie in der *getEventDataSet()*-Methode im Listing 4.74 zu sehen ist. Somit benötigt jedes Dataset einen eigenen Array. Diese einzelnen Arrays werden in der *getEvents()*-Methode, die im Listing 4.75 angeführt ist, präpariert. Für das Anzeigen der Ereignisse im Diagramm wird wieder ein Liniendiagramm verwendet. Die violetten Graphen der Ereignisse werden über dem eigentlichen Auslastungsgrafen angezeigt.

```
  getEventDataSet() {
  let dataset: any[] = [{
    type: 'line',
    label: 'Line Dataset',
    data: this.cpuData.value,
    backgroundColor: 'rgba(62, 149, 205, 0.5)',
    borderColor: '#3e95cd',
    order: 3
  }, {
```

```
      type: 'scatter',
      label: 'Scatter Dataset',
      data: this.getAnomalies(),
      backgroundColor: '#e82546',
      borderColor: '#e82546'
    }];
    this.getEvents().forEach((data) => {
      dataset.push({
        type: 'line',
        data: data,
        fill: true,
        backgroundColor: 'rgba(179, 0, 255, 0.25)',
        borderColor: 'rgb(179, 0, 255, 0.5)',
        order: 2
      });
    });
    return dataset;
  }
```

Listing 4.74: Mehrere Datasets inklusive Anomalien-Dataset und Ereignis-Datasets

Im Listing 4.75 ist konkret zu sehen, wie die Ereignisse von den Anomalien gefiltert und jeweils in einem eigenen Array gespeichert werden. Der Array muss wieder die gleiche Länge haben wie der Array mit den Auslastungsdaten, damit die Ereignisse an den korrekten Positionen eingezeichnet werden und mit den Labels, den Zeitstempeln, übereinstimmen. Anhand des Start- und Ende-Zeitstempels der Ereignisse kann der Array an den jeweiligen Stellen und dazwischen befüllt werden. In die restlichen Felder wird null gespeichert.

```
    getEvents() {
    let events: any[] = [];
    let inEvent: boolean = false;
    this.cpu.events_and_anomalies.forEach((Ereignis) => {
      if (!Ereignis.is_anomaly) {
        let tmpEvent: any[] = [];
        this.cpu.time_series_list.forEach((data) => {
          if (this.datePipe.transform(data.measurement_time, 'yyyy-MM-ddTHH:mm
  ' ?? "")
          == this.datePipe.transform(Ereignis.till_timestamp,
```

```
          'yyyy-MM-ddTHH:mm' ?? "")) {
            inEvent = true;
            tmpEvent.push(this.roundDecimal(data.value * 100, 2));
          } else if (data.measurement_time == Ereignis.timestamp) {
            tmpEvent.push(this.roundDecimal(data.value * 100, 2));
            inEvent = false;
          } else if (inEvent) {
            tmpEvent.push(this.roundDecimal(data.value * 100, 2));
          } else {
            tmpEvent.push(null);
          }
        })
        events.push(tmpEvent);
      }
    })
    return events;
  }
```

Listing 4.75: Extraktion der Events

Die Datasets werden in einer externen Funktion definiert, da unterschiedlich viele benötigt werden und die Implementierung etwas komplexer ist. Als Labels werden wie gehabt die Zeitstempel der einzelnen Datensätze verwendet, die auch zu den Ereignisse und Anomalien passen, wie in dem Listing 4.76 zu sehen ist. Auch die restlichen Optionen sind wie schon zuvor, sodass die y-Achse bei null beginnt, keine Legende angezeigt wird, das Diagramm auf unterschiedliche Bildschirmgrößen automatisch reagiert und die Tooltips in einer eigenen Methode definiert sind, aufgrund deren aufwändigeren Implementierung.

```
    this.cpuChart = new Chart("usage", {
      data: {
        datasets: this.getEventDataSet(),
        labels: this.cpuData.time
      },
      options: {
        responsive: true,
        scales: {
          y: {
            beginAtZero: true,
```

```
        },
      },
      plugins: {
        legend: {
          display: false
        },
        tooltip: {
          callbacks: {
            label: (context) => this.getEventMsg(context)
          }
        }
      }
    }
  })
```

Listing 4.76: CPU-Auslastungsdiagramm inklusive Anomalien und Ereignisse

4.6.10 Details zu einzelnen Datenpunkten

Um Details zu einzelnen Datenpunkten in den Diagrammen anzuzeigen, bietet die Chart.js Bibliothek die Option, benutzerdefinierte Tooltips zu generieren. Im Listing 4.77 wird deutlich, dass mithilfe einer Funktion Informationen des Datenpunktes, über den der Benutzer seinen Maus-Zeiger bewegt, ausgelesen und in der gewünschten Kombination als Tooltip angezeigt werden kann.

```
  tooltip: {
          callbacks: {
            label: function (context) {
              let label = context.dataset.label || '';
              if (label) {
                label += ' ';
              }
              label += context.parsed.y + ' %';
              return label;
            }
          }
```

```
          }
```

Listing 4.77: Tooltip

Beim Anzeigen von Daten, für die eine umfangreichere Implementierung benötigt wird, gibt es auch die Möglichkeit, die Funktionalität in einer eigenen TypeScript-Funktion zu definieren und diese aufzurufen. Bei Ereignisse und Anomalien werden nicht nur die Werte, sondern auch Begründungen zu diesen Ereignissen und Anomalien im Tooltip angezeigt. Um dies zu ermöglichen, werden mehr Informationen und somit eine komplexere, eigene Funktion benötigt. In der *getEventMsg()*-Methode im Listing 4.78 werden die Statistiken zu Ereignisse und Anomalien für die Anzeige im Tooltip präpariert.

```
  getEventMsg(context: TooltipItem<keyof ChartTypeRegistry>): string[] {
  let msg: string = "";
  if (context.dataset.borderColor != "#3e95cd") {
    this.cpu.events_and_anomalies.forEach((data) => {
      if (this.datePipe.transform(data.till_timestamp, 'MM-dd HH:mm') ==
      context.label || this.datePipe.transform(data.timestamp, 'MM-dd HH:mm
  ')
      == context.label) {
        msg = data.justification_message;
      }
    });
  } else {
    msg += context.parsed.y + "%";
  }
  return msg.split("\n");
}
```

Listing 4.78: Statistiken der Ereignisse und Anomalien

Wie im Listing 4.77 zu sehen ist, muss der context als Parameter übergeben werden, damit der Datensatz identifiziert werden kann. Die Zeitstempel werden verglichen und die dazugehörigen Informationen werden als Tooltip ausgegeben.

4.6.11 Benutzerdefinierte Benachrichtigungen

Es gibt die Option, eigene Warnungen anzulegen und zu bearbeiten. Mithilfe von einfachen Eingabe-Feldern und Dropdowns kann der Benutzer Kriterien für Warnungen auswählen und angeben. Wie schon im Abschnitt 4.6.1 Benutzeroberfläche beschrieben, werden für die Eingabefelder einfache HTML-Inputs verwendet, wie im Listing 4.79 zu sehen ist. Mithilfe von ngModel werden die eingegebenen Werte im TypeScript-Dokument verwertet.

```
1     <input [(ngModel)]="selectedUserAlert.type" type="text" placeholder="Name:
     ">
```

Listing 4.79: Texteingabefeld

Die neue oder überarbeitete Warnung wird, wie im Abschnitt 4.6.5 Kommunikation mit der API beschrieben, in die Datenbank gespeichert.

4.6.12 PC-Auswahl

Sollte noch kein PC ausgewählt sein, wird eine Warnung angezeigt, die den Benutzer auf die Ansicht für die PC-Auswahl weiterleitet. Dafür wird eine vordefinierte Komponente aus der CoreUI-Bibliothek verwendet, wie im Listing 4.80 beispielhaft dargestellt ist.

```
1 <c-alert *ngIf="showPcIdAlert" color="danger">
2     <div>No PC is selected. Please select one here:
3     <a cAlertLink [routerLink]="['/pc_selection']">PC Selection</a>
4     </div>
5 </c-alert>
```

Listing 4.80: Warnung der CoreUI

Auf der Ansicht für die PC-Auswahl an sich werden alle Geräte eines Nutzers aufgelistet. Er kann einen davon auswählen und auf der restlichen Webapplikation Informationen einholen. Natürlich gibt es auch die Möglichkeit die Geräte wieder zu entfernen oder neue Rechner hinzuzufügen. Dafür wird nur die Hardware-UUID und eine beliebige Bezeichnung für das Gerät benötigt.
Für das Hinzufügen eines Rechners wird aus den Informationen, die der Benutzer angegeben hat, ein JSON-String gebaut. Dieser wird, wie in Kapitel 4.6.5 Kommunikation mit der API bereits erläutert, an die API übergeben. Danach wird die Liste der Rechner erneut

geladen, damit der eben hinzugefügt Rechner ebenfalls aufscheint. Im Listing 4.81 in der *addPC()*-Methode ist die konkrete Implementierung zu sehen.

```
        addPC() {
        let userPC = {
            user_id: userID + "",
            hardware_uuid: this.newHardwareUUID,
            client_name: this.newClientName
        };

        this.pcService.addPCToUser(userPC).subscribe((response) => {
            this.pcService.getPCsOfUser(userID).subscribe((userPCs) => {
                this.userPcs = userPCs.pcs;
            });
        });
    }
```

Listing 4.81: Hinzufügen eines PCs

5 Ergebnis

Die nachfolgenden Abschnitte stellen das Ergebnis der vorliegenden Diplomarbeit mit den einzelnen Ansichten und Funktionalitäten der Webanwendung dar.

5.1 Startseite

Die Startseite ist jede Ansicht, die beim Aufrufen der Webanwendung als erstes angezeigt wird. In Abbildung 5.1 ist ersichtlich, dass die Startseite die wichtigsten Informationen über den ausgewählten Rechner übersichtlich darstellt. Dazu zählen die Marke, das Model und die Laufzeit des Rechners, die im oberen Teil der Ansicht angeführt werden. Darunter werden die CPU-, RAM und Festplatten-Auslastung inklusive relevanter Daten über die jeweilige Ressource angegeben.
Die wichtigste Funktionalität dieser Ansicht sind die Zeitstatistiken der gestarteten Prozesse, die in einem Balkendiagramm dargestellt werden. Dafür werden entweder alle gestarteten Prozesse oder die 10 Prozesse mit der höchsten Gesamtlaufzeit in dem ausgewählten Zeitraum in Erwägung gezogen. Den Zeitraum kann der Benutzer durch die Einstellungsmöglichkeiten im Dropdown über dem Diagramm nach Belieben konfigurieren.
Der letzte Bestandteil der Startseite sind die benutzerdefinierten Warnungen, die für den ausgewählten Rechner erstellt worden sind. Sobald die festgelegten Kriterien einer Warnung überschritten worden sind, wird die Warnung zur Liste hinzugefügt, beziehungsweise die Anzahl der Überschreitungen erhöht.
Da es sich bei den angezeigten Daten um rechnerspezifische Informationen handelt, werden diese nur angezeigt, wenn bereits ein Rechner ausgewählt worden ist. Das kann der Benutzer der Webanwendung in der Rechner-Auswahl-Ansicht erledigen, die im Abschnitt 5.3 angeführt ist.

Abbildung 5.1: Startseite

5.2 Prozesse

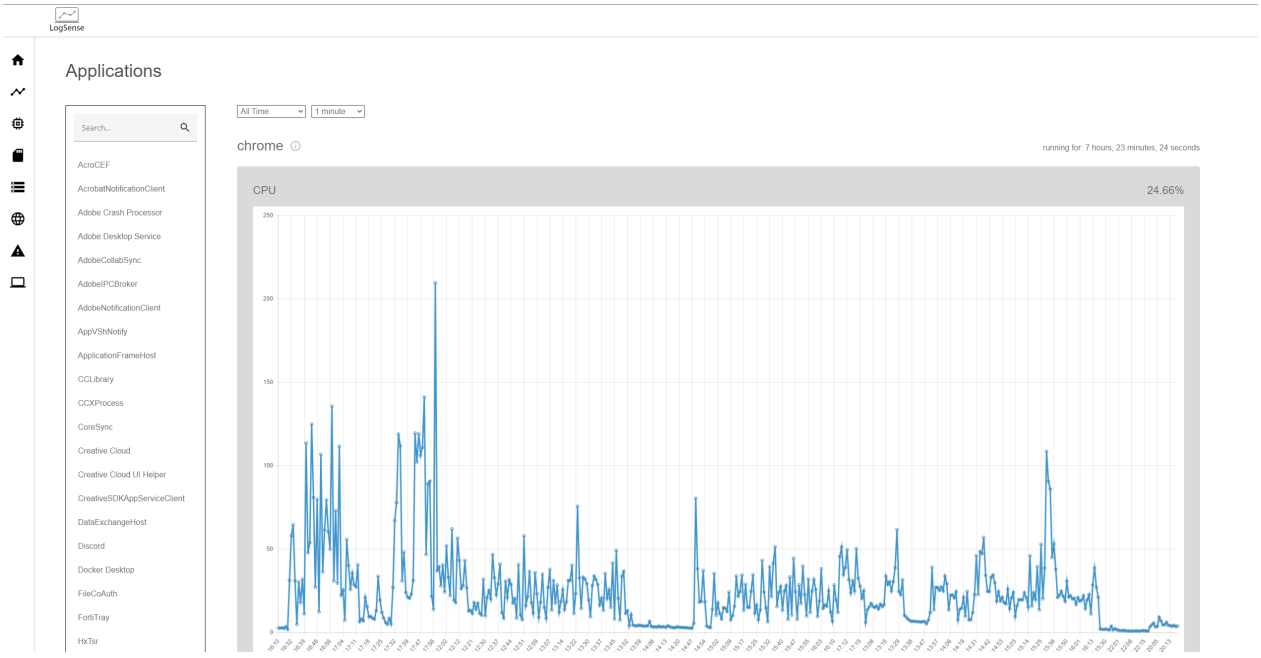


Abbildung 5.2: Applikationsseite mit Chrome als ausgewählte Applikation

Die Applikationsseite ist eine Ansicht, um den Verbrauch von Ressourcen und ausgelösten benutzerdefinierten Benachrichtigungen von einzelnen Applikationen anzusehen. Wie an der Abbildung 5.2 erkannt werden kann, werden bei der Applikationsseite eine Liste von den verfügbaren Applikationen angezeigt. Wie bei anderen Seiten gibt es die Möglichkeit, für diese Applikationen einen bestimmten Zeitrahmen anzugeben und die Datensätze nach Minuten zu gruppieren, um die Anzahl der Datensätze zu reduzieren.

Bei der linken Leiste gibt es die Möglichkeit, die Applikationen nach Namen zu filtern beziehungsweise zu durchsuchen. Die Suche wird nicht automatisch durchgeführt, sondern durch eine Benutzerinteraktion wie durch das Drücken der Enter-Taste oder das Klicken des Lupen-Icons. Bei Auswahl einer Applikation durch Klicken auf den Namen werden die gemessenen und ausgewählten Daten auf dem Graphen rechts angezeigt.

Über den Graphen steht der Name der Applikation und die gemessene Laufzeit nach *”running for:”* im angeforderten Zeitrahmen. Wie an der angeführten Abbildung 5.2 erkannt

werden kann, befindet sich neben dem Namen ein Info-Icon. Beim Schweben des Mauszeigers über dem Info-Icon werden zur jeweiligen Applikation weitere Details angezeigt wie der System-Pfad, unter dem die Applikation gespeichert ist. Außerdem wird zum Graphen immer der Durchschnittswert im Zeitrahmen auf der rechten Seite angezeigt, in diesem Fall ist der Durchschnittswert der CPU-Auslastung 24.66 Prozent gewesen.

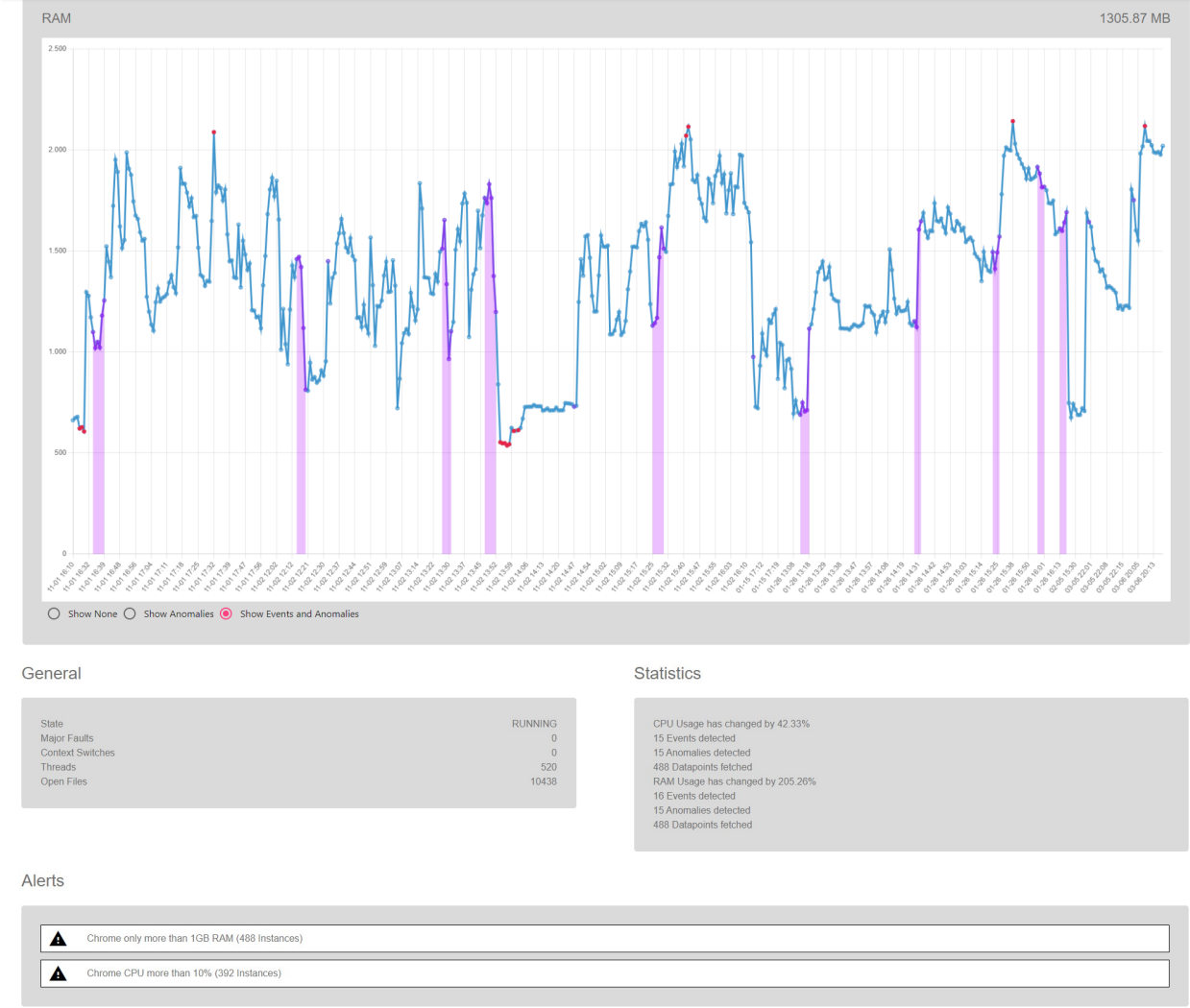


Abbildung 5.3: RAM-Graph und Statistiken

In der Abbildung 5.3 sieht man einen Graphen des RAM-Verbrauchs inklusive den vorkommenden Anomalien und Ereignisse, diese werden mit der Auswahl der Radio-Buttons eingeblendet. Darunter sieht man die Statistiken und generelle Informationen zu den RAM- und CPU-Verbrauch der jeweiligen Applikation. In diesen Statistiken wird Folgendes angeführt:

- Änderung vom ersten Datenpunkt zum letzten Datenpunkt
- Anzahl der Ereignisse
- Anzahl der Anomalien
- Anzahl der angeforderten Datenpunkte

Außerdem wird die Liste der aufgetretenen benutzerdefinierten Benachrichtigungen angezeigt. Diese beinhaltet die Nachricht der Benachrichtigung und die Anzahl, wie oft diese aufgetreten ist.

5.3 Benutzerdefinierte Benachrichtigungen

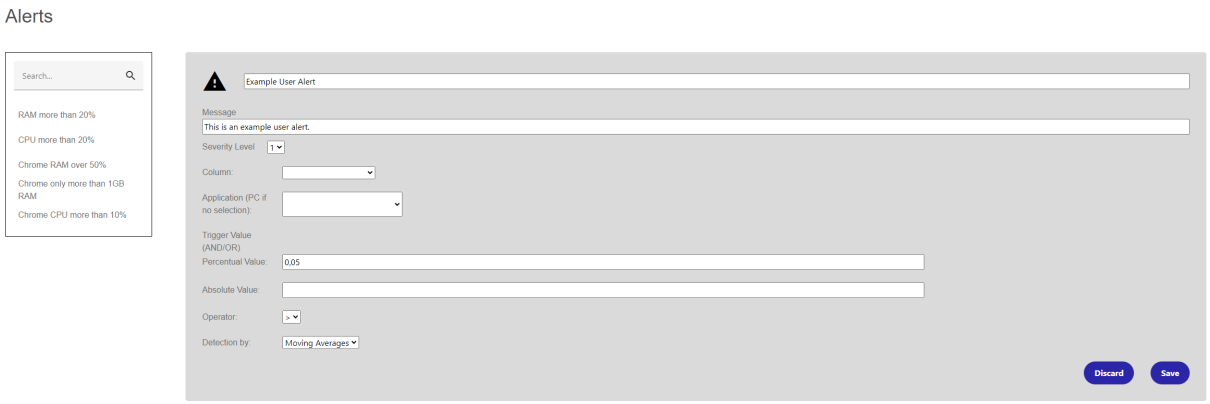


Abbildung 5.4: Seite für das Anlegen und Bearbeiten von benutzerdefinierten Benachrichtigungen

Die Seite für benutzerdefinierte Benachrichtigungen dient zum Anlegen, Bearbeiten und Löschen von benutzerdefinierten Benachrichtigungen, diese wird in der Abbildung 5.4 angeführt. Ähnlich wie bei der Applikationsseite wird eine Liste von den angelegten Benachrichtigungen angezeigt, bei dem der Benutzer in der Lage ist, diese nach einer Nachricht zu filtern und zu durchsuchen. Bei Aufrufen der Seite ist standardmäßig ein Beispiel für eine benutzerdefinierte Benachrichtigung ausgewählt, damit der Benutzer diese leicht umändern und anlegen kann. Außerdem wird ihm dadurch angezeigt, wie das Format einer Benachrichtigung aussehen sollte.

Eine Benachrichtigung besteht aus einem eindeutigen Namen und eine Nachricht die beim Auftreten angezeigt werden sollte. Außerdem wird ein Schweregrad gesetzt, der angibt, wie ernstzunehmend die aufgetretene Warnung ist. Weiters werden verschiedene Bedingungen gesetzt, die angeben, unter welchen Bedingungen die Benachrichtigung ausgelöst werden sollte. Diese Bedingungen bestehen aus folgenden Elementen:

- `Percentual Value`: Der prozentuale Schwellenwert, der überschritten werden muss, um eine Benachrichtigung auszulösen.
- `Absolute Value`: Der absolute Schwellenwert, der überschritten werden muss, um eine Benachrichtigung auszulösen.
- `Operator`: Der Operator, der angibt, wie die Bedingungen ausgewertet werden sollen. Möglichkeiten sind >, = und <
- `Column`: Gibt die Spalte an, auf die sich die Bedingungen beziehen, zu den möglichen Optionen gehören:
  - `cpu`
  - `ram`
  - `partition_major_faults`
  - `partition_minor_faults`
  - `available_memory`
  - `open_files`
- `Application`: Dies gibt die Anwendung an, auf die sich die Bedingungen beziehen. Beim Klicken auf das Dropdown-Menü kann eine Applikation von einer Liste ausgewählt werden, wenn im Menü nichts ausgewählt wird, gilt die Bedingung für den gesamten Rechner.
- `Detection by`: Gibt an, ob die Bedingungen über den gleitenden Durchschnitt oder über die einzelnen Datenpunkte erkannt werden. Meist sind Bewegungsdurchschnitte sinnvoll, da eine Benachrichtigung nur relevant ist, wenn dieser über eine längere

Zeitdauer andauert. Durch die Verwendung vom gleitenden Durchschnitt kann das Anschlagen von nicht-relevanten Benachrichtigungen vermieden werden.

Die Auswahl für die Bedingungen erfolgt größtenteils durch Dropdown-Menüs, mit Ausnahme der Eingaben für den prozentualen und absoluten Wert, bei denen Textboxen verwendet werden. Zwischen den *Percentual Value* und *Absolute Value* gibt es eine ODER-Verknüpfung, daher muss nur eine der beiden Bedingungen eintreten, um die Benachrichtigung auszulösen. Es ist auch nicht notwendig beide anzugeben, es wird lediglich einer der beiden Werte benötigt. Durch das Klicken des Save-Buttons wird die Benachrichtigung in der Datenbank angelegt und in zukünftige Anfragen von Daten überprüft werden. Außerdem wird die bisherige Liste aktualisiert.
Beim Klicken des Discard-Buttons werden alle bisherige Änderungen und Auswahl verworfen und erneut das Standard-Beispiel ausgewählt. Bei der Auswahl einer bereits angelegten Benachrichtigung

5.4 Statistiken und Charts

fix the chaos :)

CPU

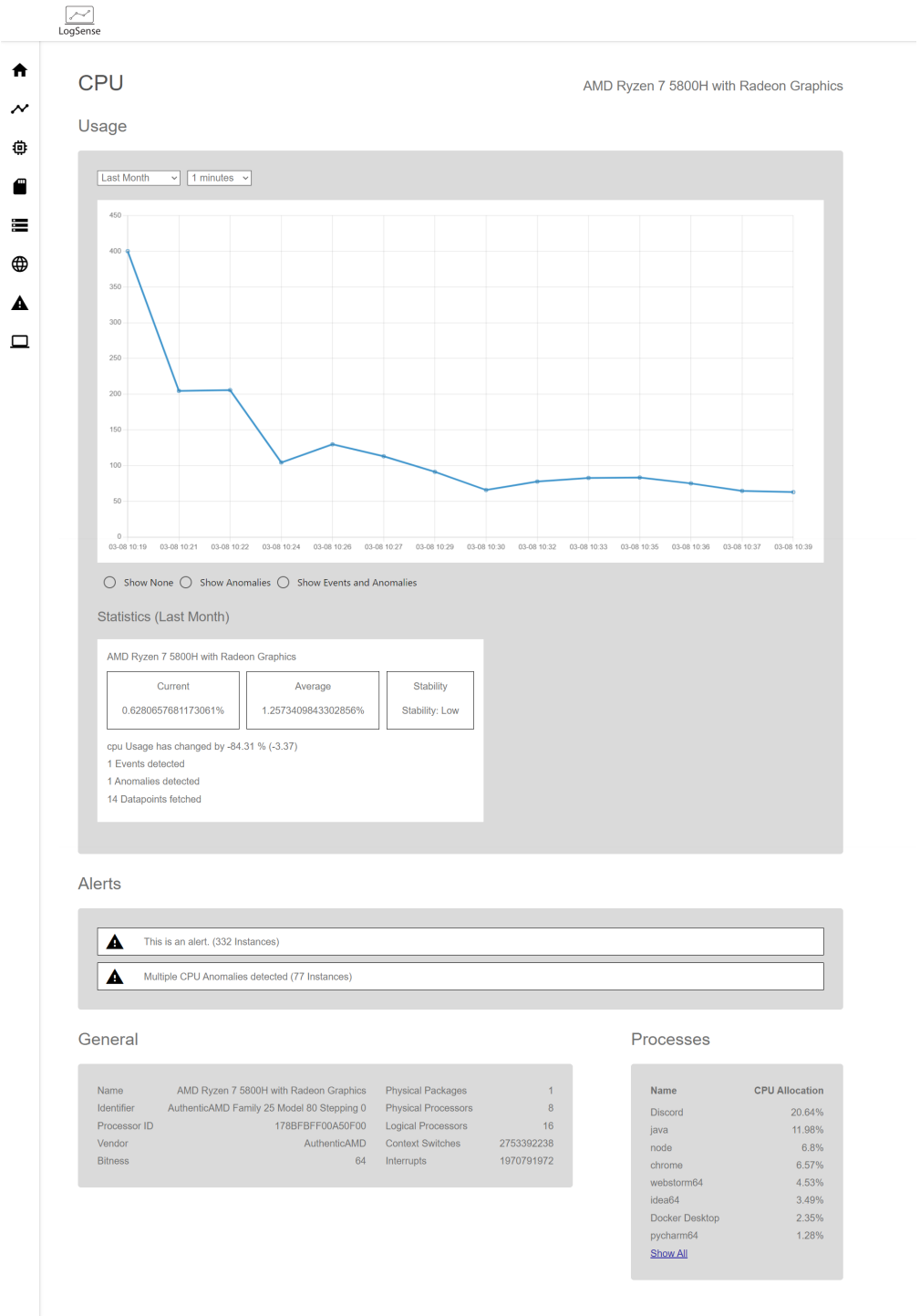


Abbildung 5.5: Ansicht der CPU

In der Abbildung 5.5 sieht man CPU-spezifische Details. Der wichtigste Aspekt ist das Liniendiagramm mit den Auslastungsdaten der CPU in einem angegebenen Zeitraum. Über dem Diagramm kann der Benutzer in einem Dropdown auswählen, von welchem Zeitraum die Daten angezeigt werden sollen. Weiteres kann der Benutzer in einem weiteren Dropdown den Zeitraum zwischen den gemessenen Daten auswählen.
Es besteht auch die Möglichkeit unter dem Diagramm über Radio-Buttons auszuwählen, ob Anomalien und Ereignisse in dem Grafen eingezeichnet werden sollen. Sollte diese Option gewählte werden, werden Anomalien als rote Punkte im Grafen markiert. Ereignisse werden als violette Bereiche im Diagramm eingezeichnet. Beim Hovern über einen markierten Datenpunkt werden Statistiken zu dem jeweiligen Ereignis oder der jeweiligen Anomalie angezeigt, wie in Abbildung 5.6 zu sehen ist.

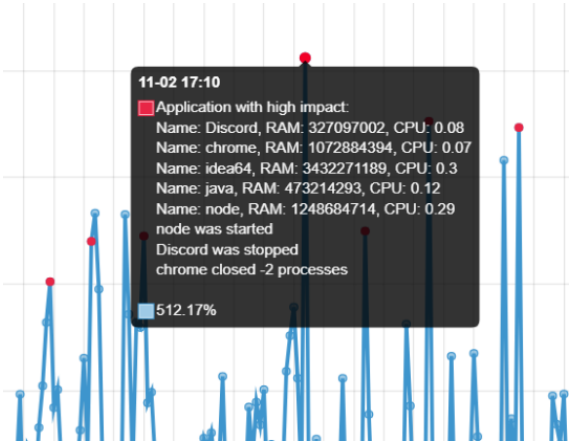


Abbildung 5.6: Informationen zu einer Anomalie

Außerdem werden für den ausgewählten Zeitraum werden unter dem Auslastungsgrafen weitere Statistiken angezeigt, wie die aktuelle und die durchschnittliche Auslastung der CPU, aber auch die Stabilität und die Änderung der Auslastung. Die Anzahl der Datenpunkte, die im Grafen angezeigt werden, wird ebenfalls mit der Menge der Anomalien und Ereignisse aufgelistet. Weiteres werden die benutzerdefinierten Warnungen angezeigt. Generelle Informationen wie der Name der CPU und Prozessor ID werden ebenfalls gemeinsam mit der CPU-Auslastung einzelner Prozesse aufgelistet. Dabei kann der Benutzer entscheiden, ob er alle Prozesse anzeigen möchte, die in dem ausgewählten Zeitraum erfasst worden sind oder nur die acht Prozesse mit der höchsten Auslastung.

RAM

Ähnlich wie bei der Detailansicht für die CPU ist die Ansicht für RAM-spezifische Details aufgebaut. Es gibt wieder die Möglichkeit über zwei Dropdowns den Zeitraum für die anzuzeigenden Daten festzulegen, sowie den Zeitraum zwischen den erfassten Datenpunkten auszuwählen. Im Liniendiagramm unterhalb sieht man in Abbildung 5.7 die RAM-Auslastung. Wie schon bei der CPU-Ansicht kann der Benutzer unterhalb des Diagramms mithilfe von Radio-Buttons auswählen, ob er Ereignisse und Anomalien im Grafen angezeigt haben möchte. Diese werden wieder als rote Punkte und violette Flächen eingezeichnet, wie schon in Abbildung 5.3 zu sehen ist.

Außerdem werden noch Informationen und Statistiken zum RAM angezeigt, wie der gesamte Speicher, der aktuell freie Speicher und die Page Size. Die restlichen Statistiken gleichen sich denen der CPU. Die benutzerdefinierten Warnungen werden neben den Auslastungswerten der Prozesse angezeigt. Dabei kann der Benutzer wieder auswählen, ob er alle Prozesse angezeigt haben möchte, die in diesem Zeitraum erfasst worden sind oder nur die acht Prozesse mit der höchsten RAM-Auslastung.

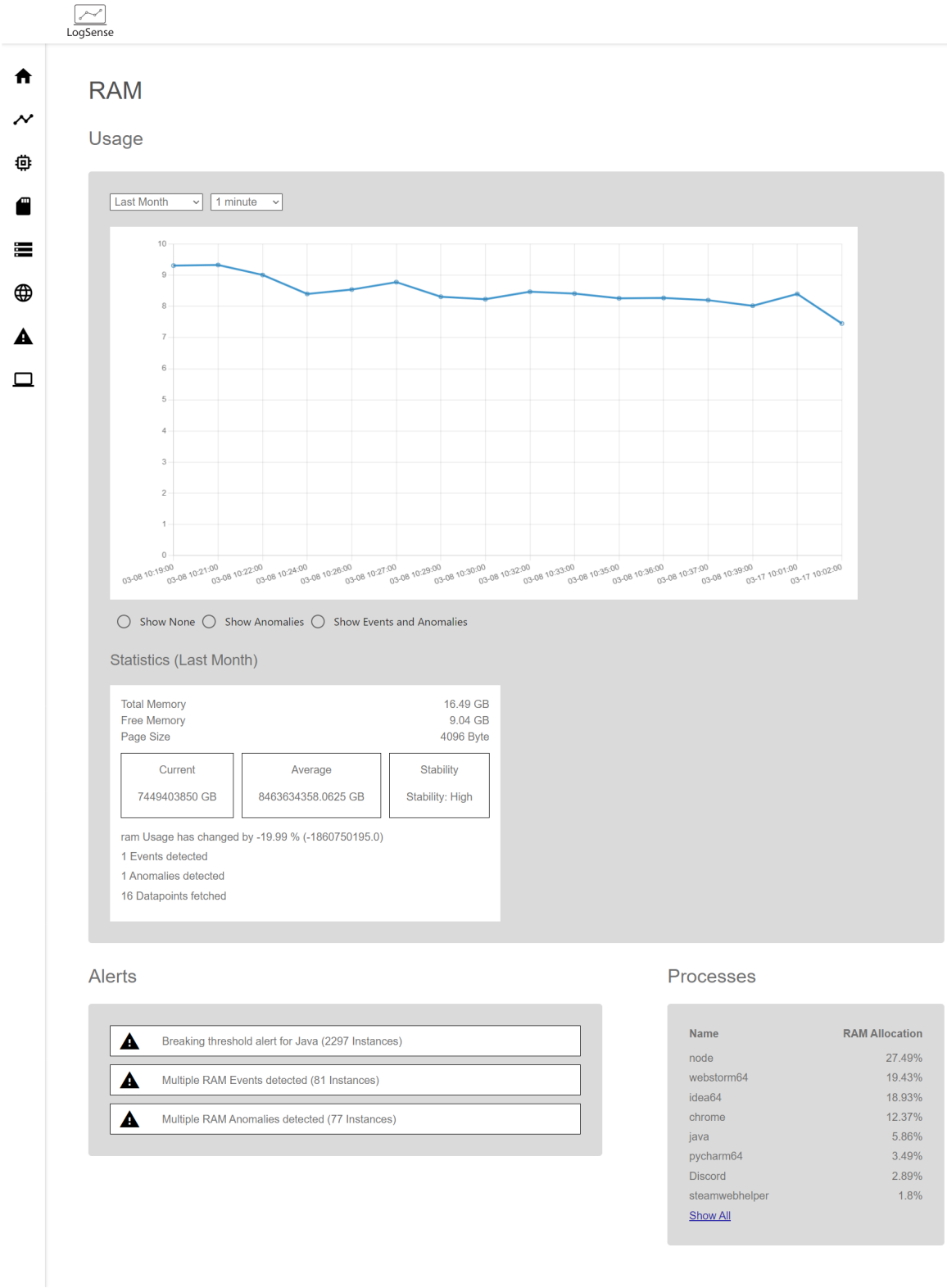


Abbildung 5.7: Ansicht des RAM

Festplatte

Auf der Seite der Festplatte wird hauptsächlich festgehalten, wie viel Gesamtspeicher der Festplatten noch zur Verfügung steht. Man kann mithilfe des Dropdown die Zeitperiode auswählen, in welcher man die Daten anzeigen möchte. Außerdem wird ein linearer Regressionsalgorithmus verwendet, um anhand der vergangenen Daten den zukünftigen Verlauf vorherzusehen. Die Vorhersage wird dann in rot am Ende des bisherigen Graphen angezeigt. Dieser Vorgang ist in der Abbildung 5.8 dargestellt.

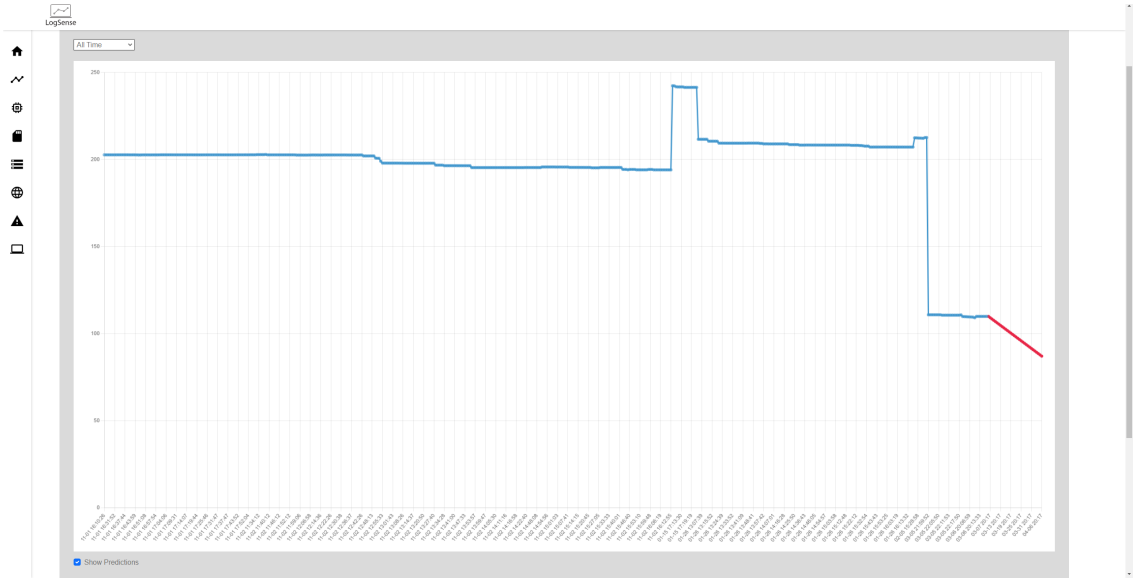


Abbildung 5.8: Ansicht der Festplatten

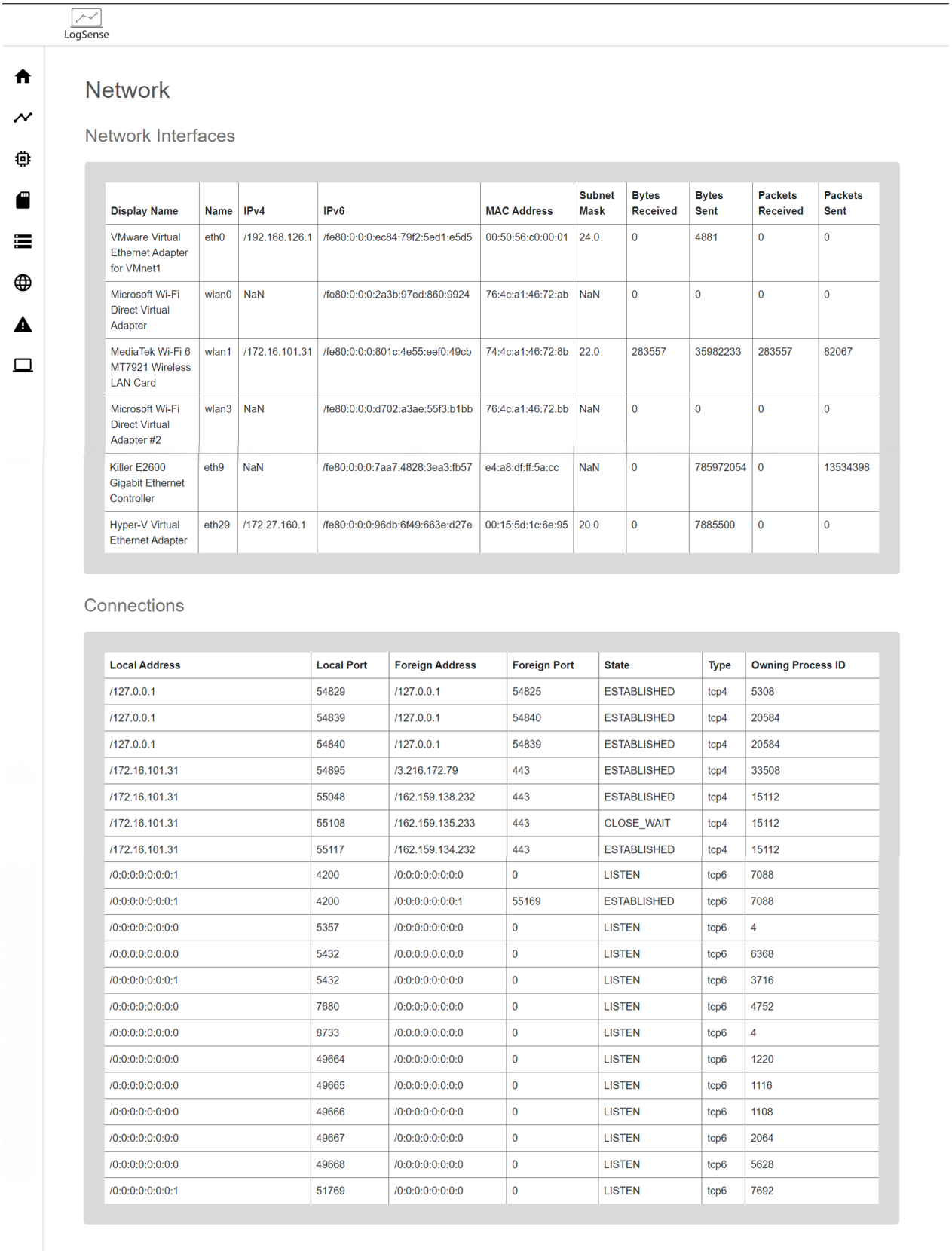
Netzwerkschnittstelle

Bei der Netzwerkseite werden sowohl die Netzwerkinterfaces, als auch die Verbindungen dargestellt.

Bei den Netzwerkinterfaces handelt es sich um physische oder virtuelle Netzwerkkarten, die die Verbindungen ans Internet ermöglichen. Auf dieser Seite werden wichtige gespeicherte Informationen über diese Netzwerkkarte dargestellt, wie beispielsweise der Anzeigename, die IP-Adresse oder die Anzahl der versendeten Bytes.

Die Verbindungen zeigen eine Liste zu und von welchen IP-Adressen und Ports Verbindungen aufgebaut wurden. Auch welcher Prozess die Verbindung aufgebaut hat und der Status der Verbindung kann hier eingesehen werden.

Die Seite wird in Abbildung 5.9 dargestellt.



Display Name	Name	IPv4	IPv6	MAC Address	Subnet Mask	Bytes Received	Bytes Sent	Packets Received	Packets Sent
VMware Virtual Ethernet Adapter for VMnet1	eth0	/192.168.126.1	/fe80:0:0:0:ec84:79f2:5ed1:e5d5	00:50:56:c0:00:01	24.0	0	4881	0	0
Microsoft Wi-Fi Direct Virtual Adapter	wlan0	NaN	/fe80:0:0:0:2a3b:97ed:860:9924	76:4c:a1:46:72:ab	NaN	0	0	0	0
MediaTek Wi-Fi 6 MT7921 Wireless LAN Card	wlan1	/172.16.101.31	/fe80:0:0:0:801c:4e55:eef0:49cb	74:4c:a1:46:72:8b	22.0	283557	35982233	283557	82067
Microsoft Wi-Fi Direct Virtual Adapter #2	wlan3	NaN	/fe80:0:0:0:d702:a3ae:55f3:b1bb	76:4c:a1:46:72:bb	NaN	0	0	0	0
Killer E2600 Gigabit Ethernet Controller	eth9	NaN	/fe80:0:0:0:7aa7:4828:3ea3:fb57	e4:a8:df:ff:5a:cc	NaN	0	785972054	0	13534398
Hyper-V Virtual Ethernet Adapter	eth29	/172.27.160.1	/fe80:0:0:0:96db:6f49:663e:d27e	00:15:5d:1c:6e:95	20.0	0	7885500	0	0

Local Address	Local Port	Foreign Address	Foreign Port	State	Type	Owning Process ID
/127.0.0.1	54829	/127.0.0.1	54825	ESTABLISHED	tcp4	5308
/127.0.0.1	54839	/127.0.0.1	54840	ESTABLISHED	tcp4	20584
/127.0.0.1	54840	/127.0.0.1	54839	ESTABLISHED	tcp4	20584
/172.16.101.31	54895	/3.216.172.79	443	ESTABLISHED	tcp4	33508
/172.16.101.31	55048	/162.159.138.232	443	ESTABLISHED	tcp4	15112
/172.16.101.31	55108	/162.159.135.233	443	CLOSE_WAIT	tcp4	15112
/172.16.101.31	55117	/162.159.134.232	443	ESTABLISHED	tcp4	15112
/0:0:0:0:0:0:0:1	4200	/0:0:0:0:0:0:0:0	0	LISTEN	tcp6	7088
/0:0:0:0:0:0:0:1	4200	/0:0:0:0:0:0:0:1	55169	ESTABLISHED	tcp6	7088
/0:0:0:0:0:0:0:0	5357	/0:0:0:0:0:0:0:0	0	LISTEN	tcp6	4
/0:0:0:0:0:0:0:0	5432	/0:0:0:0:0:0:0:0	0	LISTEN	tcp6	6368
/0:0:0:0:0:0:0:1	5432	/0:0:0:0:0:0:0:0	0	LISTEN	tcp6	3716
/0:0:0:0:0:0:0:0	7680	/0:0:0:0:0:0:0:0	0	LISTEN	tcp6	4752
/0:0:0:0:0:0:0:0	8733	/0:0:0:0:0:0:0:0	0	LISTEN	tcp6	4
/0:0:0:0:0:0:0:0	49664	/0:0:0:0:0:0:0:0	0	LISTEN	tcp6	1220
/0:0:0:0:0:0:0:0	49665	/0:0:0:0:0:0:0:0	0	LISTEN	tcp6	1116
/0:0:0:0:0:0:0:0	49666	/0:0:0:0:0:0:0:0	0	LISTEN	tcp6	1108
/0:0:0:0:0:0:0:0	49667	/0:0:0:0:0:0:0:0	0	LISTEN	tcp6	2064
/0:0:0:0:0:0:0:0	49668	/0:0:0:0:0:0:0:0	0	LISTEN	tcp6	5628
/0:0:0:0:0:0:0:1	51769	/0:0:0:0:0:0:0:0	0	LISTEN	tcp6	7692

Abbildung 5.9: Ansicht der Netzwerke

5.5 Rechner-Auswahl

Bevor die Daten über den Rechner, die laufenden Prozesse oder die Ressourcen der dazugehörigen Ansicht angezeigt werden, muss zuerst ein Rechner ausgewählt werden. Solange das nicht erledigt ist, weist die in der Abbildung 5.10 dargestellte Fehlermeldung den Benutzer darauf hin, dass er noch eine Auswahl treffen muss. Der Verweis auf die Ansicht mit der Rechner-Auswahl ermöglicht eine einfache Navigation zu dieser Seite.

Abbildung 5.10: Fehler: kein Rechner wurde ausgewählt

Wie bereits erwähnt, kann die Rechner-Auswahl durch die Verlinkung in der oben beschriebenen Fehlermeldung oder durch den Notebook-Button in der Toolbar erreicht werden. Hier werden alle bereits hinzugefügten Rechner mit ihrer Hardware UUID und einem ausgewählten Namen aufgelistet. Die Liste verfügt über die Möglichkeiten Rechner zu markieren beziehungsweise auszuwählen und mit dem Entfernen-Button aus dem System zu entfernen. Sobald ein Rechner ausgewählt worden ist, kann mithilfe der Toolbar auf eine andere Ansicht navigiert werden und die Daten des ausgewählten Rechners werden angezeigt.

Soll ein neuer Rechner überwacht und dessen Daten in dieser Webanwendung angezeigt werden, kann in dieser Ansicht die Hardware UUID des Rechners eingegeben und ein Name für diesen Rechner festgelegt werden. Der neu hinzugefügte Rechner wird dann ebenfalls in der Liste oben angeführt. Sofern der Agent mindestens einmal Daten von diesem Rechner erfasst hat und dieser Rechner ausgewählt worden ist, besteht die Möglichkeit, die Daten des neuen Rechners einzusehen.

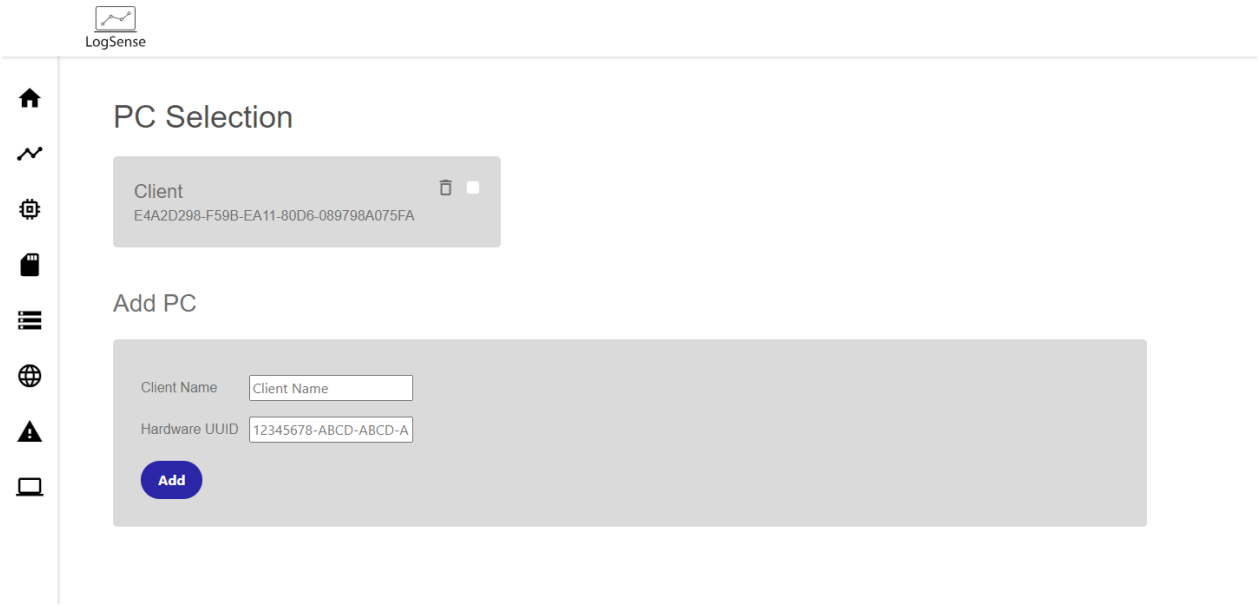


Abbildung 5.11: Ansicht der Rechner-Auswahl

5.6 Ausblick

LogSense als Projekt ist für das Projektteam fertig abgeschlossen, allerdings besteht die Möglichkeit, dass LogSense von Dynatrace im Rahmen von Praktikanten weiter entwickelt wird. Insbesondere ist hierbei LogSense interessant, da es einige Feature wie Ressource Monitoring der Dynatrace One Plattform aufweist, allerdings auf einem weitaus eingeschränkteren Rahmen. Durch diese Weiterentwicklung von LogSense sind Entwickler in der Lage, wertvolle Erkenntnisse und Einblicke in Überwachungslösungen zu bekommen.

6 Resümee

Philipp Borbely

Im Rahmen der Entwicklung dieser Diplomarbeit habe ich viele neue Technologien kennengelernt und habe ein Einblick in die Datenwissenschaft und Datenverarbeitung bekommen. Da ich bis zu Beginn der Diplomarbeit hauptsächlich nur Erfahrung mit Programmieren hatte, musste ich mich anfangs mit vielen neuen Konzepten und Themen der Datenwissenschaft beschäftigen. Besonders in der Anfangsphase der Diplomarbeit wurden viele Tests und Demo-Applikationen in Jupyter erstellt, damit ich die Themen verstehe und die Implementierung reibungslos verläuft. Auch die Programmiersprache Python und dessen verwendete Bibliotheken wie Pandas sind für mich neu gewesen, allerdings waren diese durch zahlreiche Tutorials und gute Dokumentationen mit Beispielen leicht zu verstehen. Besonders bei der Umsetzung hat unsere Betreuung im Rahmen des Praktikums geholfen, da sie Verbesserungsvorschläge bei der bisherigen Umsetzung nannten und uns Feedback lieferten. Insgesamt ist die Implementierung der Features enorm interessant, wenn auch manchmal herausfordernd gewesen, doch besonders gefallen hat es mir, dass es uns frei war, eigene Ideen und Features einzubringen, wodurch ich viel experimentieren konnte.

Sarah Ettlinger

Durch die Arbeit an dieser Diplomarbeit, habe ich die Möglichkeit erhalten, in verschiedenen Bereichen wertvolle Erfahrungen zu sammeln. Von der Programmierung einer Datenerfassungssoftware mit einer mir unbekannten Bibliothek bis zum Deployment einer Java-Anwendung konnte ich viele neue Technologien, Konzepte und Best Practices erlernen. Der Entwurf eines Softwaresystems mit mehreren Komponenten hat viel Zeit in Anspruch genommen, jedoch habe ich dadurch festgestellt, dass man die Kommunikation zwischen den Komponenten und das Definieren von Schnittstellen in solchen Projekten nicht vernachlässigen darf.

Zusätzlich hat mir die Zusammenarbeit im Team und mit dem Auftraggeber gezeigt, wie

wichtig effektive Kommunikation und gemeinsame Ziele für den Erfolg eines Projekts sind.

Thomas Jilek

Die Diplomarbeit war eine großartige Möglichkeit für mich, mich mit neuen Technologien auseinanderzusetzen. Denn sowohl mit Python als auch mit Fastapi oder auch TimescaleDB hatte ich bis dahin noch nicht die Möglichkeit gehabt, mich intensiv zu beschäftigen. Außerdem konnte ich in Erfahrung bringen, welche Bereiche der Softwareentwicklung mich wirklich interessieren: vor allem an der Planung der Architektur habe ich ein besonderes Interesse entwickelt.
Dieses Projekt hat mir nicht zuletzt verdeutlicht, wie entscheidend eine effektive Planung und Dokumentation ist. Darüber hinaus hat es mir die Wichtigkeit klarer Kommunikation innerhalb des Teams vor Augen gebracht, um ein Projekt erfolgreich abzuschließen.

Emily Stadlbauer

Durch das Verfassen dieser Diplomarbeit sowie auch das Entwickeln von LogSense, habe ich einen guten Einblick in unterschiedliche Technologien und Konzepte bekommen, aber auch im Bereich Projektentwicklung und Teamarbeit habe ich wichtige Erfahrungen gesammelt. Speziell durch die Entwicklung der Webapplikation mit Angular habe ich mein Interesse an der Frontend-Entwicklung entdeckt. Da diese Aspekte der Programmierung in der HTL Perg eher nachrangig sind, war LogSense eine ausgezeichnete Möglichkeit mich intensiver mit dem Entwerfen und Implementieren von Benutzeroberflächen auseinander zu setzen. Weiteres ist mir durch die Arbeit an der Diplomarbeit geworden, wie wichtig gute Kommunikation im Team sowie eine sorgfältige Planung für den Erfolg eines Projektes ist.

Aufgabenverteilung

Philipp Borbely

- Einleitung
  - Problemstellung
  - Zielsetzung
- Theoretische Grundlagen
  - Datenverarbeitung
    - Python
    - Pandas
    - Numpy
    - Scikit-learn
    - Ruptures
    - Matplotlib
  - Entwicklungssysteme
    - PyCharm
    - Jupyter
- Planung und Realisierung
  - Funktionalität
  - Architektur
    - Agent
    - REST-API
    - Datenbank
    - Algorithmen
    - Webapplikation

- Implementierung
  - Datenverarbeitung
    - Zusammenführung
    - Datenbeschaffung
    - Anomalieerkennung
    - Change Point Detection
    - Forecast
    - Benutzerdefinierte Benachrichtigungen
    - Begründungen
    - Statistiken berechnen
- Ergebnis
- Glossar

Sarah Ettlinger

- Einleitung
  - Projektinhalt
    - Agent
    - Analyse
    - Schnittstellen
    - Datenspeicherung
    - Benutzeroberfläche
  - Projektumfeld
    - Projektteam
    - Auftraggeber
- Theoretische Grundlagen
- Datenerfassung

    - Agent
    - Java
    - Oshi
    - java.util.concurrent Package
- Entwicklungssysteme
    - IntelliJ IDEA
- Planung und Realisierung
    - Entwurf
        * Startseite
        * Ressourcenverbrauch
        * Prozesse
        * Netzwerkschnittstellen und Verbindungen
        * Benutzerdefinierte Benachrichtigungen
        * Rechner-Auswahl
        * Gruppenanzeige
    - Informationsfluss
- Implementierung
    - Allgemeine Rechner-Daten
    - Prozesse und deren Ressourcenverbrauch
    - Zusammenführen der gemessenen Daten
    - Ressourcenverbrauch vom Rechner
    - CSV Konverter
    - REST Client
    - Programmablauf
- Ergebnis

Thomas Jilek

- Einleitung
  - Problemstellung
  - Zielsetzung
- Theoretische Grundlagen
  - Datenhaltung
    - PostgreSQL
    - TimescaleDB
    - Psycopg
    - SQL
    - Docker
  - Schnittstellen
    - Fast API
    - Flask
    - REST
  - Server
    - Ubuntu
    - nginx
  - Entwicklungssysteme
    - DataGrip
  - Sonstige Software
    - LucidChart
- Planung und Realisierung
  - Architektur
    - Agent

Emily Stadlbauer

- Einleitung
  - Projektinhalt
    - Agent
    - Analyse
    - Schnittstellen
    - Datenspeicherung
    - Benutzeroberfläche
  - Projektumfeld
    - Projektteam
    - Auftraggeber
- Theoretische Grundlagen
  - Visualisierung
    - Angular
    - BootStrap
    - Chart.js
    - Angular Material
    - CoreUI
    - Flexbox
  - Entwicklungssysteme
    - WebStorm
  - Sonstige Software
    - GitLab
    - AdobeXD
    - StarUML
  - Planung und Realisierung

    * Funktionalität
    * Entwurf
        - Startseite
        - Ressourcenverbrauch
        - Prozesse
        - Netzwerkschnittstellen und Verbindungen
        - Benutzerdefinierte Benachrichtigungen
        - Rechner-Auswahl
        - Gruppenanzeige
- Implementierung
    * Visualisierung
        - Benutzeroberfläche
        - Toolbar
        - Responsive Design
        - Hover-Effekte
        - Kommunikation mit der API
        - Grafik der Zeitaufzeichnung einzelner Prozesse
        - Zeitseriendaten grafisch darstellen
        - Anomalien anzeigen
        - Ereignisse anzeigen
        - Details zu einzelnen Datenpunkten
        - Benutzerdefinierte Benachrichtigungen
        - PC-Auswahl
- Ergebnis

Literaturverzeichnis

[1] URL https://logowik.com/adobe-xd-vector-logo-4409.html.

[2] URL https://iconduck.com/icons/101775/file-type-angular.

[3] Breakpoints, . URL https://getbootstrap.com/docs/5.0/layout/breakpoints/. Accessed: 05.03.2024.

[4] . URL https://getbootstrap.com/docs/5.0/about/brand/.

[5] URL https://brandfetch.com/chartjs.org.

[6] How does coreui angular dashboard template cut development time? URL https://coreui.io/angular/. Accessed: 05.03.2024.

[7] URL https://commons.wikimedia.org/wiki/File:DataGrip.svg.

[8] URL https://www.stickpng.com/img/icons-logos-emojis/tech-companies/docker-full-logo.

[9] URL https://fastapi.tiangolo.com/.

[10] URL https://flask.palletsprojects.com/en/3.0.x/.

[11] URL https://iconduck.com/icons/27408/gitlab.

[12] URL https://en.m.wikipedia.org/wiki/File:IntelliJ_IDEA_Icon.svg.

[13] Package java.util.concurrent, . URL https://docs.oracle.com/javase/8/docs/api/index.html?java/util/concurrent/package-summary.html. Accessed: 09.02.2024.

[14] . URL https://1000logos.net/java-logo/.

[15] URL https://slack.com/apps/A0G8S1LSC-lucidchart.

[16] URL https://www.logo.wine/logo/Nginx.

[17] URL https://en.m.wikipedia.org/wiki/File:Postgresql_elephant.svg.

[18] URL https://github.com/psycopg.

[19] URL https://en.m.wikipedia.org/wiki/File:PyCharm_Icon.svg.

[20] Interface scheduledexecutorservice. URL https://docs.oracle.com/javase/8/docs/api/java/util/concurrent/ScheduledExecutorService.html. Accessed: 27.02.2024.

[21] URL https://github.com/staruml.

[22] URL https://www.timescale.com/brand.

[23] URL https://freebiesupply.com/logos/ubuntu-logo/.

[24] URL https://de.wikipedia.org/wiki/WebStorm.

[25] The json data interchange syntax, 2017. URL https://www.ecma-international.org/wp-content/uploads/ECMA-404_2nd_edition_december_2017.pdf. Accessed: 26.03.2024.

[26] What is a web server?, 2017. URL https://www.ionos.com/digitalguide/server/know-how/web-server-definition-background-software-tips/. Accessed: 26.03.2024.

[27] How to identify and remove seasonality from time series data with python, 2020. URL https://machinelearningmastery.com/time-series-seasonality-with-python/. Accessed: 07.03.2024.

[28] Comparative analysis of time series databases in the context of edge computing for low power sensor networks, 2020. URL https://link.springer.com/chapter/10.1007/978-3-030-50426-7_28. Accessed: 25.03.2024.

[29] Psycopg 2.9.9 documentation, 2021. URL https://www.psycopg.org/docs/index.html. Accessed: 25.03.2024.

[30] Css flexbox: Mehr möglichkeiten beim webdesign, 2022. URL https://www.ionos.at/digitalguide/websites/webseiten-erstellen/css-flexbox/. Accessed: 11.02.2024.

[31] Bootstrap (framework), 2023. URL https://getbootstrap.com/. Accessed: 14.01.2024.

[32] Bootstrap (framework), 2023. URL https://de.wikipedia.org/wiki/Bootstrap_(Framework). Accessed: 14.01.2024.

[33] Chart.js, 2023. URL https://en.wikipedia.org/wiki/Chart.js. Accessed: 14.01.2024.

[34] A guide to concurrent queues in java, 2023. URL https://anytech.medium.com/understand-synchronizers-in-java-concurrency-library-in-depth-1-reentrantlock-cf43ba3a8377. Accessed: 27.02.2024.

[35] Cascading style sheets, 2023. URL https://de.wikipedia.org/wiki/Cascading_Style_Sheets. Accessed: 11.02.2024.

[36] An illustration of a decision tree based isolation forest, 2023. URL https://www.researchgate.net/figure/An-Illustration-of-a-decision-tree-based-isolation-forest-Assume-a-training-dataset-X_fig5_352121703. Accessed: 07.03.2024.

[37] Fastapi – introduction, 2023. URL https://www.geeksforgeeks.org/fastapi-introduction/. Accessed: 07.03.2024.

[38] java.util.concurrent package, 2023. URL https://www.geeksforgeeks.org/java-util-concurrent-package. Accessed: 09.02.2024.

[39] Software-agent, 2023. URL https://de.wikipedia.org/wiki/Software-Agent#:~:text=Adaptive%20Agenten%20verwalten%20ein%20Modell,der%20Ressourcen%20optimale%20Ausf%C3%BChrung%20m%C3%B6glich. Accessed: 09.12.2023.

[40] What is an add-on?, 2024. URL https://www.computerhope.com/jargon/a/addon.htm. Accessed: 26.03.2024.

[41] Adobe xd, 2024. URL https://de.wikipedia.org/wiki/Adobe_XD. Accessed: 31.01.2024.

[42] Was ist ein api-endpunkt?, 2024. URL https://www.cloudflare.com/de-de/learning/security/api/what-is-api-endpoint/. Accessed: 26.03.2024.

[43] Arima models for time series forecasting, 2024. URL https://people.duke.edu/~rnau/411arim.htm. Accessed: 07.03.2024.

[44] Autoarima dokumentation, 2024. URL https://alkaline-ml.com/pmdarima/modules/generated/pmdarima.arima.auto_arima.html. Accessed: 07.03.2024.

[45] Delta, 2024. URL https://www.rhetos.de/html/lex/delta_x.htm. Accessed: 26.03.2024.

[46] Docker overview, 2024. URL https://docs.docker.com/get-started/overview/. Accessed: 26.03.2024.

[47] What is docker?, 2024. URL https://www.ibm.com/topics/docker. Accessed: 26.03.2024.

[48] What is dynatrace, 2024. URL https://docs.dynatrace.com/docs/get-started/what-is-dynatrace. Accessed: 21.02.2024.

[49] Dynatrace, 2024. URL https://en.wikipedia.org/wiki/Dynatrace. Accessed: 21.02.2024.

[50] Gleitender durschschnitt, 2024. URL https://welt-der-bwl.de/Gleitender-Durchschnitt. Accessed: 26.03.2024.

[51] Hypertext markup language, 2024. URL https://de.wikipedia.org/wiki/Hypertext_Markup_Language. Accessed: 11.02.2024.

[52] Intellij idea overview, 2024. URL https://www.jetbrains.com/help/idea/discover-intellij-idea.html. Accessed: 30.01.2024.

[53] Java native access, 2024. URL https://en.wikipedia.org/wiki/Java_Native_Access. Accessed: 15.01.2024.

[54] Lineare regression, 2024. URL https://datatab.de/tutorial/lineare-regression. Accessed: 07.03.2024.

[55] Lucidchart, 2024. URL https://en.wikipedia.org/wiki/Lucidchart. Accessed: 25.03.2024.

[56] Nginx, 2024. URL https://en.wikipedia.org/wiki/Nginx. Accessed: 25.03.2024.

[57] Definition normalverteilung, 2024. URL https://de.statista.com/statistik/lexikon/definition/95/normalverteilung/. Accessed: 07.03.2024.

[58] The open source definition, 2024. URL https://opensource.org/osd. Accessed: 26.03.2024.

[59] Oshi - operating system hardware information, 2024. URL https://github.com/oshi/oshi. Accessed: 15.01.2024.

[60] A brief history of postgresql, 2024. URL https://www.postgresql.org/docs/current/history.html. Accessed: 25.03.2024.

[61] Postgresql, 2024. URL https://en.wikipedia.org/wiki/PostgreSQL. Accessed: 25.03.2024.

[62] Representational state transfer, 2024. URL https://de.wikipedia.org/wiki/Representational_State_Transfer. Accessed: 25.03.2024.

[63] Ruptures pelt documentation, 2024. URL https://centre-borelli.github.io/ruptures-docs/code-reference/detection/pelt-reference/#ruptures.detection.pelt.Pelt. Accessed: 05.02.2024.

[64] structured query language (db) (sql), 2024. URL https://itwissen.info/structured-query-language-DB-SQL.html. Accessed: 26.03.2024.

[65] Definition standardabweichung, 2024. URL https://de.statista.com/statistik/lexikon/definition/126/standardabweichung/. Accessed: 26.03.2024.

[66] 2024. URL https://staruml.io/. Accessed: 31.01.2024.

[67] Statistische methoden, 2024. URL https://www.strategischeumweltpruefung.at/sup-methoden/sup-umweltfolgenabschaetzung/analysemeth/statistisch. Accessed: 26.03.2024.

[68] Stationarity differencing: Definition, examples, types, 2024. URL https://www.statisticshowto.com/stationarity/. Accessed: 26.03.2024.

[69] Time series database, 2024. URL https://en.wikipedia.org/wiki/Time_series_database. Accessed: 25.03.2024.

[70] Timescale documentation, 2024. URL https://docs.timescale.com/. Accessed: 25.03.2024.

[71] Ml | underfitting and overfitting, 2024. URL https://www.geeksforgeeks.org/underfitting-and-overfitting-in-machine-learning/. Accessed: 26.03.2024.

[72] Definition varianz, 2024. URL https://de.statista.com/statistik/lexikon/definition/138/varianz/. Accessed: 26.03.2024.

[73] Version control concepts and best practices, 2024. URL https://homes.cs.washington.edu/~mernst/advice/version-control.html.

[74] Web application framework, 2024. URL https://web.archive.org/web/20150723163302/http://docforge.com/wiki/Web_application_framework. Accessed: 26.03.2024.

[75] Robin Böhm. Angular-tutorial für einsteiger, 2024. URL https://angular.de/artikel/angular-tutorial-deutsch/. Accessed: 17.01.2024.

[76] Edgar F. Codd. *A Relational Model of Data for Large Shared Data Banks*. Communications of the ACM, 1970.

[77] Chart.js Community. Chart.js, 2023. URL https://www.chartjs.org/docs/latest/. Accessed: 14.01.2024.

[78] Jupyter Dokumentation, 2024. URL https://jupyter.org/. Accessed: 07.03.2024.

[79] Matplotlib Dokumentation, 2024. URL https://matplotlib.org/. Accessed: 07.03.2024.

[80] Numpy Dokumentation, 2024. URL https://numpy.org/. Accessed: 07.03.2024.

[81] Pandas Dokumentation, 2024. URL https://pandas.pydata.org/. Accessed: 07.03.2024.

[82] Python Dokumentation, 2024. URL https://www.python.org/. Accessed: 07.03.2024.

[83] Ruptures Dokumentation, 2024. URL https://centre-borelli.github.io/ruptures-docs/. Accessed: 07.03.2024.

[84] Scikit-Learn Dokumentation, 2024. URL https://scikit-learn.org/stable/. Accessed: 07.03.2024.

[85] J. Fielding, R.; Reschke. Hypertext transfer protocol (http/1.1): Semantics and content - 4.3.3 post, 2014. URL https://www.rfc-editor.org/info/rfc7231.

[86] M.; Reschke J. Fielding, R.; Nottingham. Http semantics, 2022. URL https://www.rfc-editor.org/rfc/rfc9110.html.

[87] Roy Thomas Fielding. *"Chapter 5: Representational State Transfer (REST)". Architectural Styles and the Design of Network-based Software Architectures (Ph.D.)*. University of California, Irvine, 2000.

[88] Mathieu Fortin. A guide to concurrent queues in java, 2024. URL https://www.baeldung.com/java-concurrent-queues. Accessed: 27.02.2024.

[89] JetBrains, 2023. URL https://www.jetbrains.com/de-de/pycharm/. Accessed: 02.04.2024.

[90] JetBrains, 2023. URL https://www.jetbrains.com/de-de/pycharm/. Accessed: 02.04.2024.

[91] JetBrains, 2023. URL https://www.jetbrains.com/de-de/webstorm/. Accessed: 27.01.2024.

[92] M.; Salz R. Leach, P.; Mealling. *A Universally Unique IDentifier (UUID) URN Namespace*. Internet Engineering Task Force, 2005.

[93] Microsoft, 2024. URL https://www.typescriptlang.org/. Accessed: 11.02.2024.

[94] Eamonn; Zhu Qiang; Cash Sydney; Westover Brandon Mueen, Abdullah; Keogh. *Exact Discovery of Time Series Motifs*. University of California, 2009.

[95] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python, 2011. URL https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.IsolationForest.html. Accessed: 05.02.2024.

[96] ArcGIS Pro. How change point detection works, 2024. URL https://pro.arcgis.com/en/pro-app/latest/tool-reference/space-time-pattern-mining/how-change-point-detection-works.htm. Accessed: 05.02.2024.

[97] Martin Reddy. *API Design for C++*. Elsevier Science, 2011.

[98] SASS-Team. Sass basics, 2024. URL https://sass-lang.com/guide/. Accessed: 11.02.2024.

[99] Y Shafranovich. Http semantics, 2005. URL https://www.ietf.org/rfc/rfc4180.txt.

[100] Angular Entwickler Team. Attribute directives, 2022. URL https://angular.io/guide/attribute-directives. Accessed: 11.02.2024.

[101] Angular Entwickler Team. Built-in directives, 2022. URL https://angular.io/guide/built-in-directives. Accessed: 11.02.2024.

[102] Angular Entwickler Team. Cli overview and command reference, 2022. URL https://angular.io/cli. Accessed: 11.02.2024.

[103] Angular Entwickler Team. Angular routing, 2022. URL https://angular.io/guide/routing-overview. Accessed: 11.02.2024.

[104] Angular Entwickler Team. Structural directives, 2022. URL https://angular.io/guide/structural-directives. Accessed: 11.02.2024.

[105] Angular Entwickler Team. Introduction to services and dependency injection, 2022. URL https://angular.io/guide/architecture-services. Accessed: 11.02.2024.

[106] Angular Entwickler Team. Angular components overview, 2023. URL https://angular.io/guide/component-overview. Accessed: 11.02.2024.

[107] Angular Entwickler Team. Understanding dependency injection, 2023. URL https://angular.io/guide/dependency-injection. Accessed: 11.02.2024.

[108] Angular Entwickler Team. Understanding communicating with backend services using http, 2023. URL https://angular.io/guide/understanding-communicating-with-http. Accessed: 11.02.2024.

[109] Angular Entwickler Team, 2023. URL https://material.angular.io/. Accessed: 17.01.2024.

[110] ASGI Team, 2018. URL https://asgi.readthedocs.io/en/latest/. Accessed: 25.03.2024.

[111] GitLab Entwickler Team. 10 reasons why enterprises choose gitlab, 2024. URL https://about.gitlab.com/why-gitlab/. Accessed: 31.01.2024.

[112] Uvicorn Team, 2024. URL https://www.uvicorn.org/. Accessed: 25.03.2024.

[113] Bex Tuychiev. A comprehensive introduction to anomaly detection, 2023. URL https://www.datacamp.com/tutorial/introduction-to-anomaly-detection. Accessed: 07.03.2024.

[114] Iveta Vistorskyte. Concurrency vs parallelism: The main differences, 2021. URL https://oxylabs.io/blog/concurrency-vs-parallelism. Accessed: 21.02.2024.

Abbildungsverzeichnis

Code Listings

Tabellenverzeichnis

Glossar

RAM

RAM (Random Access Memory) bezeichnet elektronischen Computerspeicher zum Speichern von Arbeitsdaten und Maschinencode, der in beliebiger Reihenfolge gelesen und geändert werden kann. Dieser wird normalerweise mit flüchtigen Speichertypen assoziiert.

CPU

Die CPU (Central Processing Unit) ist der Hauptprozessor in einem Computer.

UUID

Eine UUID (Universally Unique Identifier)[92] ist eine praktisch eindeutige 128-Bit-Kennzeichnung. Der Begriff Globally Unique Identifier (GUID) wird ebenfalls verwendet, meist in Microsoft-Systemen.

API

Ein API (Application Programming Interface) [97] ist eine Softwareschnittstelle zur Kommunikation zwischen zwei oder mehr Computerprogrammen oder -komponenten.

JSON

JSON (JavaScript Object Notation)[25] ist ein offenes, sprachunabhängiges Standarddateiformat zum Austausch von Daten(z.B.: zwischen einem Webserver und Webbrowser), einschließlich Webanwendungen, das Klartext zur Formatierung von Attribut-Werte-Paaren bzw. Arrays verwendet. JSON-Dateien verwenden die Namenserweiterung .json.

Time-series Data

Time Series Data (Zeitreihen Daten)[94] bezeichnet eine Reihe von Datenpunkten, die in zeitlicher Reihenfolge indiziert sind, also um eine Sequenz aus Datenpaaren bestehend aus Zeit und dazugehörendem Wert.

add-on

Add-on[40] bezeichnet Softwaremodule, die zugehörigen Anwendungen hinzugefügt werden, um die Funktionen der Anwendung zu erweitern.

Open-source

Open Source[58] bezeichnet Quellcode (inkl. Designdokumente), der zur Weiterverbreitung und möglichen Änderung frei verfügbar gemacht wird. Das Open-Source-Modell ist ein dezentrales Softwareentwicklungsmodell, das eine offene Zusammenarbeit fördert.

space-time partitioning

Space-time partitioning bezeichnet die Organisation bzw. Formatierung von Zeitreihen Daten (siehe Eintrag Time Series Data) im Speicher

web-framework

Ein Web Framework[74] (WF) oder Web Application Framework (WAF) ist ein Software-Framework, das die Entwicklung von Webanwendungen einschließlich Webdiensten, Webressourcen und Web-APIs unterstützen soll.

persistieren

Speichern von Daten, so dass die Daten selbst nach dem Beenden der Anwendung noch erhalten bleiben. Hierfür werden Speichermedien wie Datenbanken und Dateien verwendet.

Repository

Repository[73] bezeichnet in der Softwareentwicklung eine Datenstruktur in Versionskontrollsystemen, die Metadaten zur Verwaltung von Änderungen an Computerprogrammen, Dokumenten, großen Websites oder anderen Informationssammlungen zur Verfügung stellt.

HTTP

HTTP (HyperText Transfer Protocol) [86] ist ein Protokoll der Anwendungsschicht im Internet Protocol Suite-Modell und die Grundlage der Datenkommunikation für das World Wide Web, wo Hypertextdokumente Hyperlinks zu anderen Ressourcen enthalten.

REST

REST (Representational State Transfer)[87] ist ein Software-Architekturstil, der als Leitfaden für den Entwurf und die Entwicklung der Architektur für das World Wide Web entwickelt wurde.

REST Client

Eine Anwendung, welche HTTP-Anfragen an eine REST-API sendet und die erhaltenen Antworten verarbeitet.

CSV

CSV (Comma-Separated Values)[99] bezeichnet ein Dateiformat zum Speichern von tabellarischen Daten (Zahlen und Text) im Klartext, das Kommas zum Trennen von Werten und Zeilenumbrüche zum Trennen von Datensätzen verwendet. CSV-Dateien verwenden die Namenserweiterung .csv.

POST Request

POST[85] ist eine von HTTP unterstützte Anfragemethode zum Hochladen einer Datei oder zum Absenden eines ausgefüllten Webformulars im World Wide Web. Die POST-Anfragemethode verlangt von Natur aus, dass ein Webserver die im Textkörper der Anfragenachricht enthaltenen Daten akzeptiert.

Benutzerdefinierte Benachrichtigungen

Benutzerdefinierte Benachrichtigungen sind vom Benutzer erstellte Benachrichtigungen, welche es ihm ermöglichen, Meldungen zu potenziellen Problemen zu erhalten. Diese bekommt dieser nicht als Push-Benachrichtigung, sondern als Meldung in der Weboberfläche. Eine Benachrichtigung besteht aus einem eindeutigen Namen, einer Nachricht, einem Schwierigkeitsgrad und mehren Bedingungen, die zum Auslösen benötigt werden.

Anomalien

Mit Anomalien in dieser Arbeit sind Extremwertpunkte und Datenpunkte mit ungewöhnlichen Verhalten gemeint, welche bei den PC- und Anwendungsdaten auftreten. Die Erkennung von Anomalien zielt darauf ab, ungewöhnliches Verhalten von Applikationen aufzuzeigen, wodurch das Erkennen von Problemen beim Rechner vereinfacht wird. Herkömmlicherweise werden immer Datenpunkte als Anomalien gewertet, die im Vergleich zu den nächsten Datenpunkten enorm hohe oder niedri-

ge Werte aufweisen. [113]

Ereignisse

Mit Ereignisse sind Ereignisse im Trendverlauf gemeint, konkret handelt es sich dabei um Änderungspunkte. Diese sind Zeitpunkte in einer Zeitreihe, an denen sich das Verhalten der Daten oder statistische Eigenschaften signifikant □andern, beispielsweise durch einen enormen Anstieg oder Abfall der Daten. [96]

Begründungen

Begründungen sind Nachrichten, welche zusätzlichen Informationen zur Erklärung und Begründung von Ereignissen und Anomalien beinhalten. Die Begründungsnachricht enthält eine Liste von Applikationen und Informationen darüber, ob Applikationen und deren Prozesse zu einem Zeitpunkt geschlossen oder gestartet worden sind.

Stationarität

Stationarität von Daten bedeutet, dass statistische Eigenschaften wie Mittelwert und Varianz über die Zeit konstant bleiben. Viele statistische Methoden oder Modelle setzen die Stationarität von Daten voraus, da sie das Arbeiten mit Zeitreihendaten erleichtern. [68]

Normalverteilung

In der Statistik handelt es sich bei der Normalverteilung, als auch Gauß-Verteilung bekannt, um eine Wahrscheinlichkeitsverteilung. Daten werden als normalverteilt bezeichnet, wenn die Verteilung einer spezifischen Form folgt, welche als Glockenkurve gekennzeichnet wird. Diese ist symmetrisch um den Mittelwert der Daten und fällt nach beiden Seiten hin steil ab, Zweidrittel aller Messwerte liegen innerhalb der

Entfernung der Standardabweichung zum Mittelwert. [57]

Saisonalität

Mit Saisonalität sind wiederkehrende Muster oder Zyklen zu bestimmten Intervallen von Zeitreihen gemeint. Dies führt dazu, dass die Daten nicht-stationär sind. Statistische Modelle wie ARIMA sind in der Lage, Daten mit Saisonalität zu behandeln, allerdings treten wiederkehrende Muster wie das Installieren von Applikationen durch den Benutzer bei den Messdaten unregelmäßig auf. Zu Beginn des Projektes ist angenommen worden, dass es sich bei dem freien Speicherplatz um saisonale Daten handelt, allerdings ist beim Arbeiten mit Daten schnell erfasst worden, dass durch die irregulären Intervalle die Daten keinerlei Saisonalität aufweisen. [27]

Gleitender Durchschnitt

Ein gleitender Durchschnitt ist eine Methode zur Glättung von Zeitreihendaten, bei der der Mittelwert über eine bestimmte Anzahl von aufeinanderfolgenden Datenpunkten berechnet wird. Er wird verwendet, um kurzfristige Schwankungen zu reduzieren und langfristige Trends sichtbar zu machen. [50]

Varianz

Die Varianz ist ein Maß für die Streuung oder die Abweichung der Werte in einem Datenset um den Mittelwert. Sie wird verwendet, um die Verteilung der Daten zu quantifizieren und die Homogenität oder Heterogenität des Datensets zu bewerten. [72]

Statistische Methoden

Statistische Methoden sind Techniken zur Analyse von Daten, die darauf abzielen, Muster, Beziehungen oder Trends zu identifizieren. Dazu gehören ARIMA und die

lineare Regression. [67]

Delta

Delta bezieht sich auf die Veränderung oder Differenz zwischen zwei Werten, insbesondere in einem Zeitverlauf. [45]

Standardabweichung

Die Standardabweichung ist ein Maß für die durchschnittliche Abweichung der Werte in einem Datenset um den Mittelwert. Sie gibt an, wie weit die einzelnen Werte typischerweise von diesem Mittelwert entfernt sind und wird häufig zur Beschreibung der Streuung der Daten verwendet. [65]

Underfitting

Underfitting tritt auf, wenn ein Machine-Learning-Modell zu simpel ist, um komplexe Zusammenhänge in den Daten zu erfassen. Dies geschieht oft, wenn zu wenige Datensätze zum Trainieren des Modells verfügbar sind. [71]

Anhang

Diplomarbeitsplakat

Abnahmeformular

Dynatrace Austria GmbH
Am Fünfundzwanziger Turm 20, 4020 Linz

+43 732 908 208
austria@dynatrace.com
www.dynatrace.com

Dient zur Vorlage

Linz, 29.02.2024

Bestätigung

Wir, Dynatrace Austria GmbH, bestätigen, dass die Diploarbeit „LogSense“, erstellt von Philipp Borbely, Sarah Ettlinger, Thomas Jilek und Emily Stadlbauer, zu unserer Zufriedenheit erstellt wurde.

Inhalt der Diplomarbeit

1. SQL-Skript zur Erstellung des Datenbankschemas (TimescaleDB)
2. Agent - Java Projekt & Executable
3. User Interface - Angular Projekt
4. Backend & API - Python Projekt
5. Anleitung für das Setup des Systems (README.md Datei)

Funktionalitäten der Webanwendung

1. Overview: generelle Informationen über den Rechner, Ressourcenverbrauch von CPU, RAM und Disk, Übersicht über gesamte Laufzeit der gestarteten Prozesse, Alerts für den Rechner

Dynatrace Austria GmbH
Am Fünfundzwanziger Turm 20, 4020 Linz

+43 732 908 208
austria@dynatrace.com
www.dynatrace.com

2. Detail-Ansichten für alle Ressourcen (CPU, RAM und Disk): Diagramm über Ressourcenverbrauch, Statistiken bezüglich des Ressourcenverbrauchs, Alerts für die ausgewählte Ressource CPU: allgemeine Informationen über die CPU, CPU-Auslastung pro Prozess RAM: RAM-Auslastung pro Prozess Disk: Informationen über Speichermedien und deren Partitionen
3. Prozess-Ansicht: Liste mit allen gestarteten Prozessen für den ausgewählten Prozess: Ressourcenverbrauch von CPU und RAM, Alerts für den Prozess, allgemeine Informationen und Statisken über den Prozess
4. Custom Alerts erstellen: Liste mit allen bisher erstellten Custom Alerts, Möglichkeit bereits erstellte Custom Alerts zu bearbeiten bzw. neue Custom Alerts zu erstellen
5. PC Selecton: Liste mit allen hinzugefügten Rechnern und Möglichkeit einen Rechner auszuwählen, Möglichkeit neue Rechner im System zu registrieren

Mit freundlichen Grüßen

DocuSigned by:
25122BCAEACD4DC...
Alexander Scheran

DocuSigned by:
Veronika Leibetseder
B14466964EA646E...
Veronika Leibetseder

BofA Re: Dynatrace Austria GmbH
Unicredit Bank Austria AG
Swift Code: BKAUATWW
IBAN: AT431200010012534078

UID-Nr.: ATU 38604701
Firmenbuch: FN 91482h, LG Linz