HAW HAMBURG

Bachelorarbeit

Andre Brand

Bewertung von Architekturalternativen für Big Data Systeme

Fakultät Technik und Informatik
Department Informatik

Faculty of Computer Science and Engineering
Department Computer Science

Andre Brand

Bewertung von Architekturalternativen für Big Data Systeme

Bachelorarbeit eingereicht im Rahmen der Bachelorprüfung
im Studiengang Bachelor of Science Angewandte Informatik
am Department Informatik
der Fakultät Technik und Informatik
der Hochschule für Angewandte Wissenschaften Hamburg

Betreuender Prüfer: Prof. Dr. Olaf Zukunft
Zweitgutachter: Prof. Dr. Stefan Sarstedt

Eingereicht am: 19. Juli 2019

Andre Brand

Thema der Arbeit

Bewertung von Architekturalternativen für Big Data Systeme

Stichworte

Big Data, Architekturen, Bewertung, Alternativen

Kurzzusammenfassung

Big Data ist in vielen Unternehmen ein maßgebliches Werkzeug zur intelligenten Erstellung von wirtschaftlichen Kennziffern und wird zur Steuerung der Geschäftsprozesse genutzt. Es gibt viele Ausarbeitungen zu den unterschiedlichsten Big Data Architekturen. Diese Arbeit beinhaltet die Bewertung von unterschiedlichen Big Data Architekturen durch die Implementierung der Architekturen und das Anwenden einer Methode zur Bewertung von Softwarearchitekturen. Zuerst werden die zu bewertenden Architekturen vorgestellt und Methoden zur Bewertung erläutert. Danach wird das Vorgehen bei der Implementierung der Architekturen dargestellt. Hierbei wird ebenfalls der Aufwand der einzelnen Implementierungen erfasst. Darauf folgend wird eine Methode zur Bewertung der Architekturen gewählt und das weitere Vorgehen aufgezeigt. Die Bewertung aller vorgestellten Architekturen wird vorgenommen und zum Schluss in einem Fazit zusammengefasst.

Andre Brand

Title of Thesis

Evaluation of architecture alternatives for Big Data systems

Keywords

Big Data, Architectures, Evaluation, Alternatives

Abstract

In many companies Big Data is a decisive tool for the intelligent creation of economic indicators and is used to control business processes. There are many elaborations of Big Data architectures. This work includes the evaluation of different big data architectures by implementing the architectures and applying a method to evaluate software architectures. First, the architectures to be evaluated are presented and methods for evaluation are explained. Then the procedure for the implementation of the architectures is presented. Here also the expenditure of the individual implementations is seized. Afterwards a method for the evaluation of the architectures is selected and the further procedure is pointed out. The evaluation of all presented architectures is made and summarized at the end in a conclusion.

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

Abkürzungen

API Application Programming Interface.

ATAM Architecture Tradeoff Analysis Method.

DSL Domain Specific Language.

ETL Extract, Transform, Load.

GUI Graphical User Interface.

HTTP Hypertext Transfer Protocol.

HTTPS Hypertext Transfer Protocol Secure.

NFA nichtfunktionale Anforderungen.

SAAM Software Architecture Analysis Method.

TSU Telematic Service Units.

UML Unified Modeling Language.

z.B. zum Beispiel.

1 Einleitung

1.1 Motivation

Big Data ist ein Schlagwort, das seit einiger Zeit sowohl in der Fachliteratur als auch in der Tagespresse breit diskutiert wird. Immer mehr Unternehmen versuchen aus Big Data einen wirtschaftlichen Nutzen zu ziehen. [13, S. 4] Die anfallenden Daten unterscheiden sich in Struktur, sind global verteilt und teilweise nur unzuverlässig verfügbar. Herkömmliche Systeme wie Relationale Datenbanken stoßen dabei an ihre Grenzen. [13, S. 7] Um diese hohen Datenaufkommen verarbeiten und einen Nutzen daraus ziehen zu können, müssen Systeme geschaffen werden, die skalierbar sind und verteilt arbeiten können. Hierbei ist eine intelligente Architektur maßgeblich.

1.2 Zielsetzung

Unterschiedliche Anwendungszwecke machen es unumgänglich, dass verschiedene Architekturen für Big Data Systeme erstellt werden. Obwohl diese Architekturen vielfältige Aufgaben bewältigen sollen, bleibt die Kernaufgabe immer dieselbe und es müssen gewisse Anforderungen an Big Data Systeme eingehalten werden. Um zu überprüfen, ob Architekturalternativen die Anforderungen einhalten, werden in dieser Arbeit unterschiedliche Methoden zur Bewertung der Architekturen vorgestellt und auf insgesamt vier Architekturen angewendet.

1.3 Aufbau der Ausarbeitung

In Kapitel 2 wird zunächst der Begriff Big Data und die Anforderungen an ein solches Big Data System erläutert. Danach werden die zu bewertenden Architekturen vorgestellt.

Im Anschluss werden Methoden zur Bewertung von Architekturen eingeführt.
In Kapitel 4 wird eine Implementierung der einzelnen Architekturen beschrieben und der dabei entstehende Aufwand näher betrachtet.
In Kapitel 5 wird zunächst eine Methode zur Bewertung der Architekturen ausgewählt. Anschließend wird das Vorgehen bei der Bewertung vorgestellt und zum Schluss werden alle Architekturen danach bewertet.
In Kapitel 6 werden die Erkenntnisse, die in dieser Arbeit gewonnen wurden, zusammenfassend wiedergegeben und interpretiert. Abschließend wird ein Ausblick gegeben.

2 Grundlagen zu Big Data

Im täglichen Geschäft fallen für immer mehr Unternehmen eine stetig wachsende Menge an Daten an. Dabei kann es sich um Daten handeln, die durch Transaktionen, Sensoren in Mobiltelefonen, Autos oder Industriemaschinen oder als Nebenprodukt durch andere Aktivitäten, wie der Nutzung von Social Media entstehen. Die Menge dieser Daten lässt sich nur noch selten mit herkömmlichen Systemen verarbeiten und speichern. Daher musste für die Verwaltung dieser Datenmengen ein neues Verfahren entwickelt werden[17].

Ein Big Data System muss vor allem mit folgenden 5 V's umgehen können:

- **Volume** (Menge): Die Daten sind in einer umfangreichen Menge, zumeist im Tera- bis Zettabytebereich, vorhanden.
- **Velocity** (Geschwindigkeit): Datenströme sollen möglichst in Echtzeit ausgewertet und analysiert werden.
- **Variety** (Vielfalt): Darunter versteht man das Vorhandensein von strukturierten, semi-strukturierten sowie unstrukturierten Daten.
- **Veracity** (Verlässlichkeit): Da die Genauigkeit der Daten oft nicht garantiert werden kann, muss diese durch Algorithmen überprüft und gesichert werden.
- **Value** (Wert): Da die Bereitstellung der Infrastruktur und das Beschäftigen von ausgebildetem Personal hohe Kosten verursacht, muss durch die Nutzung eines Big Data Systems ein Mehrwert für das Unternehmen geschaffen werden.

Die Nutzung von Big Data Systemen ist nicht nur für profitorientierte Unternehmen geeignet, sondern auch für Regierungen, öffentliche Verwaltungen, NGOs (Non Governmental Organizations) und NPOs (Non Profit Organizations) [20, S. 13].

2.1 Nutzen von Big Data

Durch das Wissen, das aus den gesammelten Daten gezogen werden kann, ist es für Unternehmen möglich, Vorteile im Wettbewerb zu erzielen. Hierzu muss das Unternehmen die gewonnenen Informationen für Zwecke der Unternehmenssteuerung nutzen [27].

Ein Beispiel für die Nutzung von Big Data zeigt der Energiekonzern EWE in seinem enera-Projekt. Anhand der Verbrauchsdaten wird versucht, Strom und Gas zu regulieren, um so die vorhandenen Ressourcen effizienter nutzen zu können. Zudem wird mittels Verkehrsdaten geplant, wo Ladestationen für Elektroautos sinnvoll sind. Weitere Daten spielen zum Beispiel (z.B.) für die Infrastruktur und die vorhandenen Stromleitungen eine Rolle. Dadurch kann die Elektromobilität für den Verbraucher attraktiver gestaltet werden [12].

Die Bundespolizei konnte mit Hilfe von Big Data die Erstellung der polizeilichen Eingangsstatistiken automatisieren. Die Daten aus 160 Revieren, 80 Inspektionen, zehn Direktionen und dem Präsidium werden in einem Data Warehouse aufbereitet, qualitätsgesichert und stehen für die Beantwortung von Anfragen ohne große Zeitverzögerung zur Verfügung. Vorher wurden die Daten in mehreren Excel Dateien gespeichert, manuell gepflegt und an übergeordnete Direktionen versandt. Anschließend wurden die Daten erneut geprüft und zusammengeführt, um dann an die nächste Dienststelle übergeben zu werden, bis die Daten letztlich bundesweit ausgewertet werden konnten. Die Beantwortung einer Anfrage dauerte so einen Monat. Nun kann die Auswertung nahezu in Echtzeit erstellt werden [21].

Der E-Commerce Bereich und die Anbieter von Streamingdiensten können ebenfalls durch die Nutzung von Big Data Systemen profitieren. Durch die Speicherung von Kundendaten können personalisierte Empfehlungen für Produkte erstellt und an den Kunden weitergereicht werden. Hierdurch können weitere Umsätze generiert oder der Kunde durch eine zufriedenstellende Benutzererfahrung gebunden werden [22].

2.2 Kritik

Bei allen Chancen, die die Nutzung von Big Data bietet, bestehen auch Risiken, die betrachtet werden müssen.

Ein Beispiel für mögliche Risiken ist die Analyse durch Menschen. Bestehende Korrelationen können möglicherweise als wichtig erachtet werden, obwohl diese fragwürdig sind. Dies zeigt sich unter anderem darin, dass die ansteigenden iPhone-Verkäufe seit der Markteinführung mit der Anzahl an Morden korrelierte. Bei dieser Korrelation ist eine Kausalität offensichtlich ausgeschlossen, jedoch können Sachverhalte vorliegen, bei denen dies nicht so offensichtlich ist [14].

Ein Praxisbeispiel ist der Versuch, die Manipulation der Kilometerstände von gebrauchten Fahrzeugen zu erkennen. Diese Manipulation kann erhebliche Schäden für den Kunden und die Marke darstellen, daher lohnt sich das Erkennen solcher Manipulationen. Hierbei ist es wichtig, dass Falschaussagen mit sehr geringer Wahrscheinlichkeit getroffen werden dürfen, da es im Zweifel zu Gerichtsprozessen führen kann. Jedoch führte unter anderem das Fehlen von Daten zu Schwierigkeiten. Dies konnte durch unterschiedliche Ereignisse verursacht werden. Das Löschen der Speicher in der Werkstatt oder die tatsächliche Manipulation am Kilometerstand sind mögliche Ursachen. Zudem sind die neusten Tricks der Betrüger nicht bekannt und können somit nicht untersucht werden [16, S. 117].

2.3 Anforderungen an ein Big Data System

Ein Big Data System sollte gewisse Anforderungen erfüllen, damit es in unterschiedlichen Bereichen erfolgreich nutzbar ist.

2.3.1 Belastbarkeit und Fehlertoleranz

Da Big Data Systeme grundsätzlich verteilt implementiert werden, muss mit Ausfällen von Rechnern oder sogar ganzen Clustern gerechnet werden. Diese Ausfälle müssen kompensiert werden können.
Des Weiteren muss das System gegen menschliche Fehler, z.B. durch fehlerhafte Programmierung weitestgehend geschützt sein [18, S. 24].

2.3.2 Lesen und Aktualisieren mit geringen Latenzzeiten

Falls erforderlich, muss das Big Data System in der Lage sein, bei Abfragen und Aktualisierungen eine geringe Latenzzeit zu erzielen. Hierbei darf die Verfügbarkeit und Belastbarkeit des Systems nicht gefährdet werden [18, S. 25].

2.3.3 Skalierbarkeit

Skalierbarkeit bezeichnet die Fähigkeit eines Systems, eine steigenden Menge an Objekten, Elementen oder eine steigenden Arbeitslast verarbeiten zu können oder erweiterungsfähig zu sein [4].
Das Big Data System muss sowohl bei einer wachsenden Datenmenge als auch bei einer steigenden Arbeitslast weiterhin uneingeschränkt zur Verfügung stehen können. Deshalb müssen dem System durch das Hinzufügen von Rechenkapazität durch Erweiterung eingesetzter oder Hinzufügen neuer Rechner mehr Ressourcen zur Verfügung gestellt werden können [18, S. 25].

2.3.4 Allgemeingültigkeit

Ein Big Data System darf nicht nur für ein bestimmtes Konzept erstellt werden, sondern muss für unterschiedliche Zwecke anwendbar sein [18, S. 25].

2.3.5 Erweiterbarkeit

Das Big Data System soll Erweiterungen, z.B. durch Ergänzung der Funktionalität, bei einem minimalen Entwicklungsaufwand ermöglichen. Hierzu ist das konvertieren der Daten in ein anderes Format oftmals zwingend erforderlich. Dies soll ebenfalls mit minimalem Aufwand möglich sein [18, S. 25].

2.3.6 Ad-hoc-Abfragen

Durch das gezielte Abfragen von Daten können unvermutete Datensätze entdeckt und ausgewertet werden. Dadurch werden wiederum neue Möglichkeiten zur Optimierung der

Geschäftsvorgänge eröffnet und es können neue Arten von Anwendungen entstehen [18, S. 25 f.].

2.3.7 Minimaler Wartungsaufwand

Unter dem Wartungsaufwand eines Big Data Systems versteht man die Arbeit, die für den reibungslosen Betrieb des Systems notwendig ist. Hierzu zählt unter anderem die Erweiterung um neue Maschinen zur Skalierung, die Fehlerbehebung im laufenden Betrieb, sowie die Überwachung aller laufenden Prozesse. Je komplexer das System ist, desto schwieriger und aufwendiger sind die Aufgaben zur Wartung. Aus diesem Grund sollte das System so einfach wie Möglich gehalten werden, um eben diesen Wartungsaufwand möglichst gering zu halten [18, S. 25].

2.3.8 Fehlerbehebung

Fehler können z.B. durch falsche Eingaben oder fehlerhaft implementierte Algorithmen entstehen. Um sie zu beheben, muss bei jedem Wert nachvollzogen werden können, wie dieser zustande gekommen ist. Andernfalls muss mit fehlerhaften Daten weitergearbeitet werden, die dann alle Ergebnisse mehr oder weniger falsch beeinflussen können [18, S. 25].

3 Vorstellung der Architekturen

3.1 Lambda Architektur

In der Lambda Architektur werden Abfragen als Funktionen auf sämtliche Daten behandelt. Jedoch ist das direkte Ausführen der Abfragen aufgrund der enormen Menge an Daten innerhalb eines Big Data Systems sehr rechenaufwändig und würde enorm viele Ressourcen in Anspruch nehmen. Zudem wäre die Latenzzeit deutlich zu hoch. Darum ist eine Vorabberechnung der benötigten Abfragen sinnvoll.
Die vorab berechneten Abfragen werden so gespeichert, dass man wahlfrei darauf zugreifen kann. Diese gespeicherten Abfragen werden Batch Views genannt [18, S. 32 f.].

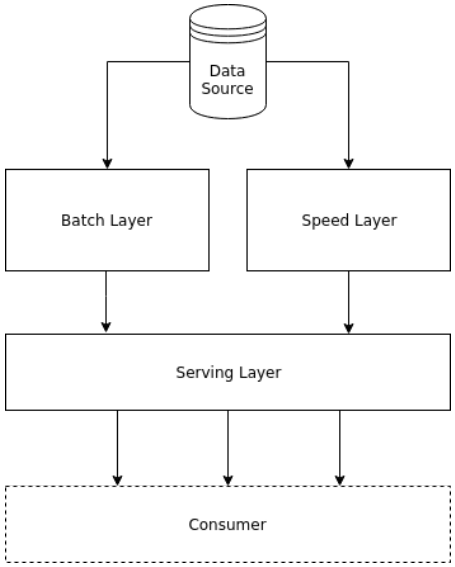

Abbildung 3.1: Bausteine einer Lambda Architektur

Die Lambda Architektur wird als etablierte Architektur gewählt, da sie die meist genannte Big Data Architektur ist und in zahlreichen Implementierungen wirtschaftlich genutzt

wird.

3.1.1 Batch Layer

Der Batch Layer der Lambda Architektur hat zwei Aufgaben. Zum einen muss er eine ständig wachsende Sammlung an Daten halten. Zum anderen muss er ständig Abfragen auf diesem Datensatz ausführen. Diese Abfragen sind in der Regel sehr aufwendig und haben eine hohe Latenzzeit. Eine Berechnung kann oft mehrere Stunden oder Tage dauern. Wenn die Abfragen ausgeführt wurden, werden die Ergebnisse an den Serving Layer weitergeleitet [18, S. 33 ff.].

3.1.2 Serving Layer

Der Serving Layer hat die Aufgabe, die Ergebnisse der Abfragen aus dem Batch Layer zu veröffentlichen. Hat der Batch Layer eine Abfrage erledigt, wird das Ergebnis im Serving Layer gespeichert und ersetzt die jeweils vorangegangenen Ergebnisse. Diese gespeicherten Ergebnisse werden als Batch Views bezeichnet [18, S. 35].

3.1.3 Speed Layer

Der Batch Layer aktualisiert nach Berechnung der Abfragen die Batch Views im Serving Layer. Die Views beinhalten jedoch nicht die Datensätze, die während der Berechnung eingegangen sind. Für eine Berechnung der Ergebnisse in Echtzeit fehlen die Daten der letzten Stunden, welche aber durch den Speed Layer berücksichtigt werden. Da dieser seine Berechnungen jedoch schneller durchführen muss, werden hier nicht alle Daten genutzt. Neuberechnungen wie im Batch Layer werden nicht durchgeführt, sondern die bisher vorhandenen Ergebnisse werden anhand der neuen Daten aktualisiert. Die erzeugten Views werden ebenfalls durch den Serving Layer veröffentlicht. Wenn der Batch Layer eine Berechnung ausgeführt und das Ergebnis an den Serving Layer weitergegeben hat, kann die dazugehörige, durch den Speed Layer berechnete View, verworfen werden [18, S. 36 f.].

3.2 Learning Analytics Architecture

Durch Daten, die innerhalb des Hochschulprozesses gesammelt werden, können Studenten, die Gefahr laufen zu scheitern oder auszusteigen, besser identifiziert und unterstützt werden. Da die Daten in einer umfangreichen Menge zur Verfügung stehen, ist eine Verarbeitung mit klassischen RDBMS nicht zielführend. Es fehlt an geeigneten konzeptionellen Big Data Architekturen für diesen Anwendungszweck. Das Paper "A Big Data Architecture for Learning Analytics in Higher Education" befasst sich mit dieser Thematik und stellt eine Architektur vor. Die Architektur wird im Folgenden erläutert [19].

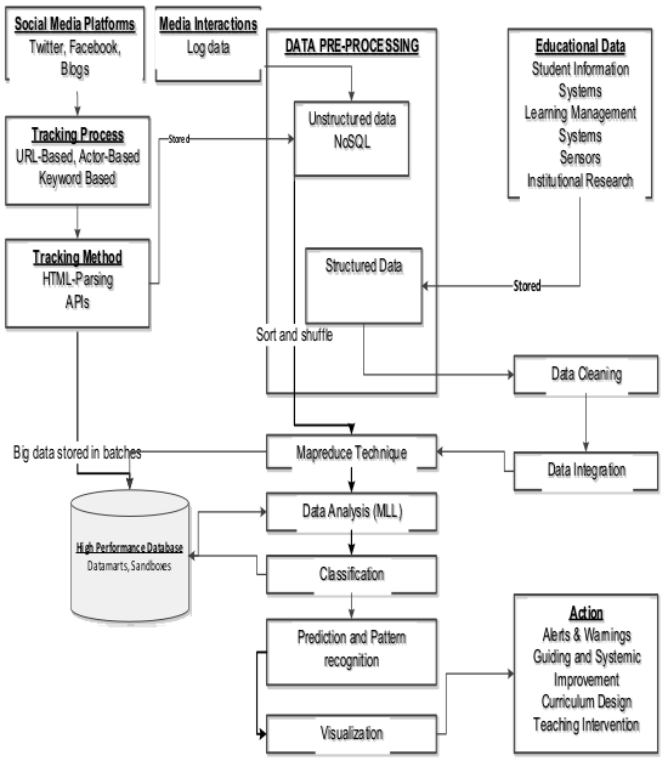

Abbildung 3.2: Bausteine der Learning Analytics Architektur [19]

Die Learning Analytics Architecture wird als Alternativarchitektur vorgestellt und bewertet, da sie sich mit der Förderung von Studenten auseinandersetzt. Da die Quote von Studienabbrechern in einigen Studiengängen bei 54 % liegt, könnte solch ein System die Bedingungen für Studenten verbessern.

3.2.1 Datenerhebungsdienst

Der Datenerhebungsdienst umfasst alle Datenquellen, die innerhalb des Hochschulprozesses Daten über die Studierenden sammeln. Dies können z.B. Studierendeninformationssysteme, Lernmanagementsysteme, soziale Netzwerke, Schülerkarten oder Sensoren sein. Die erzeugten, strukturierten und unstrukturierten Daten werden zur Speicherung an das Datenspeicher- und Managementsystem geleitet und von dort aus an das Datenanalysesystem weitergereicht [19].

3.2.2 Datenspeicher- und Managementsystem

Im Datenspeicher- und Managementsystem werden die Daten der Studenten gespeichert und für das Datenanalysesystem bereitgestellt. Es übernimmt auch die Aufgabe, die eingehenden Rohdaten zu verarbeiten und in eine Form zu konvertieren, die von dem Datenanalysesystem gut verarbeitet werden kann.
Das Datenbankmanagementsystem muss alle Funktionen, wie Pufferung und Echtzeit-Abfrageoptimierung beinhalten [19].

3.2.3 Datenanalysesystem

Im Datenanalysesystem werden die zuvor gesammelten und bereinigten Daten mittels intelligenter Verarbeitungsalgorithmen ausgewertet, um aussagekräftige Informationen zu extrahieren [19].

3.2.4 Datenvisualisierungssystem

Das Datenvisualisierungssystem stellt die im Datenanalysesystem erzielten Ergebnisse visuell dar. Durch die Darstellung soll das Treffen von Entscheidungen einfacher und schneller ermöglicht werden [19].

3.3 Reference Architecture for Big Data Systems in the National Security Domain

Big Data Architekturen der nationalen Sicherheitsbehörden haben ein einzigartiges Ausmaß an Anwendungsfällen. Unter anderem müssen strategische Georaumanalysen erstellt, Videoanalysen durchgeführt oder Open-Source Dateien von z.B. Webseiten und Social Media Kanälen gesammelt und gespeichert werden. Um diesen Anforderungen gerecht zu werden, wird eine Architektur benötigt, die unter anderem mit der Fülle der unterschiedlichen Datenmengen arbeiten kann [15]. Dafür wurde die Reference Architecture for Big Data Systems in the National Security Domain entwickelt.

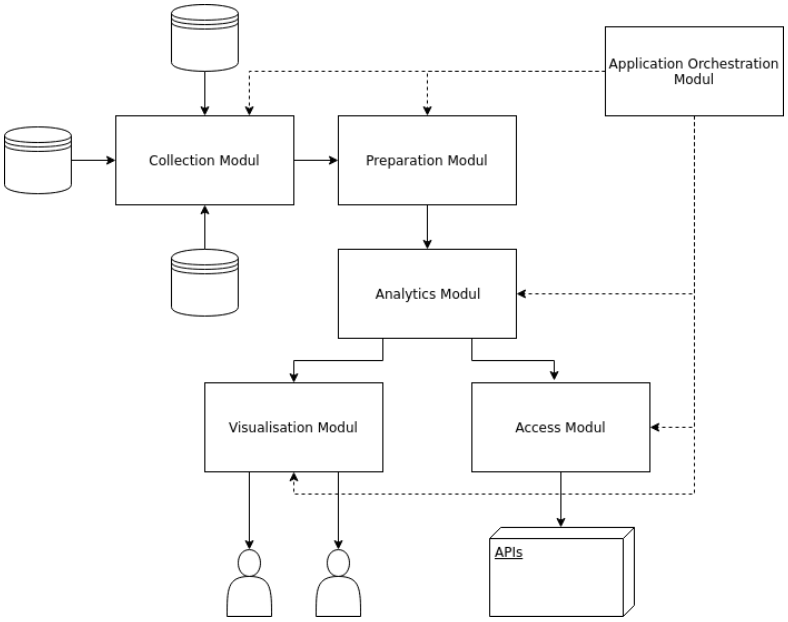

Abbildung 3.3: Bausteine der Reference Architecture for Big Data Systems in the National Security Domain

Die Reference Architecture for Big Data Systems in the National Security Domain wird als Architekturalternative vorgestellt, da sie eine gut beschriebene Architektur ist, die auch in der Praxis von den australischen Sicherheitsbehörden übernommen wurde [28].

3.3.1 Big Data Application Provider Modul

Das Big Data Application Provider Modul beinhaltet alle Module, die Datentransformationen und -analysen durchführen und die Geschäftslogik auf Anwendungsebene umfassen [15].

3.3.1.1 Application Orchestration Modul

Im Application Orchestration Modul werden andere Module des Big Data Application Providers konfiguriert und kombiniert. Zusätzlich werden integrierte Aktivitäten in eine zusammenhängende Anwendung integriert. Eine Anwendung ist in diesem Fall die komplette Datenverarbeitung durch das System.
Dies kann durch Menschen, Software oder einer Kombination aus beidem vorgenommen werden und entweder zur System-Designzeit oder durch eine Graphical User Interface (GUI) beziehungsweise eine Domain Specific Language (DSL) erfolgen [15].

3.3.1.2 Collection Modul

Das Collection Modul bildet eine Schnittstelle zu externen Datenanbietern. Es kann an die Merkmale und Vorgaben dieser Anbieter angepasst werden, um Datenverlust zu vermeiden [15].

3.3.1.3 Preparation Modul

Im Preparation Modul werden die Daten gemäß des Extract, Transform, Load (ETL) Prozesses für die Analyse vorbereitet und verarbeitet.
Dies beinhaltet folgende Arbeitsschritte:

- Datenvalidierung (z.B. mittels Checksumme)
- Datenbereinigung (z.B. Entfernen oder Korrigieren von fehlerhaften Daten)
- Optimierung (z.B. Entfernen von Duplikaten)
- Schema-Transformation und Standardisierung
- Indizierung zur Unterstützung einer schnelleren Suche

Zusätzlich kann das Modul die Daten mit Informationen aus anderen Quellen anreichern. Dabei handelt es sich in der Regel um eine einfache Verarbeitung der Daten, wie z.B. das Erstellen von Satzzählungen oder das Hinzufügen von Standortnamen auf Basis von Koordinaten. Eine komplexere Anreicherung der Daten wird durch das Analytics Modul vorgenommen [15].

3.3.1.4 Analytics Modul

Das Analytics Modul extrahiert das Wissen aus den Daten. Darüber hinaus trägt es zum weiteren ETL Zyklus bei, indem fortgeschrittene Datenanreicherungen und -transformationen vorgenommen werden [15].

3.3.1.5 Visualisation Modul

Im Visualisation Modul werden die verarbeiteten Daten und die aus der Analyse resultierenden Ergebnisse visuell dargestellt, sodass die Darstellung Bedeutung und Wissen vermittelt. Je nach Bedarf können die Visualisierungen in einem statischen Dokument erzeugt oder mittels On-Demand-Generierung in einer interaktiven Oberfläche dargestellt werden. Die interaktive Oberfläche kann unter anderem die Filterung der Daten beinhalten und zur Erstellung von neuen Daten oder der Bestätigung beziehungsweise der Korrektur von vorhandenen Daten dienen [15].

3.3.1.6 Access Modul

Das Access Modul bildet eine Schnittstelle für alle Data Consumer. Data Consumer sind Programme, die außerhalb des Systems angesiedelt werden. Die Schnittstelle kann mittels Application Programming Interface (API) oder über Webservices angesprochen werden. Daten können je nach Anwendung mit Hilfe dieses Moduls angefragt und auch hinzugefügt werden [15].

3.3.2 Big Data Framework Provider Modul

Das Big Data Framework Provider Modul umfasst alle vom Big Data Application Provider Modul verwendeten Dienste. Mehrere Instanzen des Big Data Application Provider Moduls können sich eine Instanz des Big Data Framework Provider Moduls teilen.

3.3.2.1 Processing Modul

Das Processing Modul befasst sich mit der effizienten, skalierbaren und zuverlässigen Durchführung der Analysen. Die Infrastruktur mit 10 bis 1000 Knoten, die für die Ausführung der Analysen notwendig ist, wird durch das Modul bereitgestellt. Darüber hinaus wird die Verteilung der Berechnung und Verarbeitung koordiniert [15].

3.3.2.2 Messaging Modul

Mit Hilfe des Messaging Moduls werden Nachrichten zur Bereitstellung und Übertragung der Daten, sowie für Steuerungsfunktionen zwischen den Komponenten in einer zuverlässigen Warteschlange verwaltet. Das Messaging Modul muss eine Vielzahl von Clients und Programmiersprachen unterstützen.
Je nach Größe und Durchsatz der Nachrichten muss entschieden werden, ob ein verteiltes Messaging Framework genutzt werden soll. Zudem kann entschieden werden, ob die Nachrichten dauerhaft gespeichert werden [15].

3.3.2.3 Data Storage Modul

Das Data Storage Modul befasst sich in erster Linie damit, persistente Daten zuverlässig und effizient bereitzustellen. Dies beinhaltet die logische Datenorganisation, Datenverteilungs- und Zugriffsmethoden und die Datenbeschaffung.

Die Datenorganisation beschreibt das Datenspeicherformat und die Zugriffsmethoden behandeln die vom Big Data Application Provider Modul geforderten Zugriffsarten (z.B. über API). Es ist üblich, Datensätze in unterschiedlichen Repräsentationen bereitzustellen, um eine effiziente analytische Ausführung von unterschiedlichen Anwendungsfällen zu unterstützen.

Sollten die Daten über ein Cluster verteilt gespeichert werden, behandelt das Data Storage Modul eventuelle Verfügbarkeits- und Konsistenzprobleme der Daten und befasst sich mit der Toleranz der Partitionen innerhalb des Clusters [15].

3.3.2.4 Infrastructure Modul

Das Infrastructure Modul stellt die Ressourcen zur Verfügung, die von den anderen Modulen benötigt werden. Hierzu zählen folgende Dienste:

- **Networking:** Ressourcen, die zum Übertragen der Daten von einem Modul zum anderen genutzt werden.
- **Computing:** Physikalische Prozessoren und Speicher, die zur Ausführung von Software dienen.
- **Storage:** Ressourcen, die die Persistenz der Daten gewährleisten.
- **Environmental:** Physische Ressourcen, die beim Aufbau einer Instanz eines Big Data Systems berücksichtigt werden müssen (z.B. Strom, Kühlung).

Infrastruktur- und Rechenzentrumsdesign können ein wichtiger Faktor beim Erreichen der gewünschten Leistung sein. Eine Big Data Infrastruktur muss skalierbar und zuverlässig sein [15].

3.3.3 Cross-Cutting Modul

Alle Module des Cross-Cutting Moduls befassen sich mit Problemen, die sich auf fast alle anderen Module auswirken [15].

3.3.3.1 Security Modul

Das Security Modul behandelt die Zugriffsregelung auf die Anwendungen und die vorhandenen Daten. Außerdem kümmert es sich um die Erkennung und Verhinderung von Eindringversuchen. Beispielsweise können Zugriffsmuster analysiert werden.
Gerade Big Data Systeme stellen ein attraktives Ziel für Angreifer dar und da die Daten mehrere Module durchlaufen, muss Sorge getragen werden, dass in jedem dieser Module alle Sicherheitsvorkehrungen eingehalten werden [15].

3.3.3.2 Management Modul

Das Management Modul teilt sich in zwei Hauptaufgaben auf.
Einerseits muss es sich um das Systemmanagement kümmern, zu dem Aktivitäten wie Überwachung, Konfiguration, Bereitstellung und Kontrolle der Infrastruktur und der Anwendungen zählt.
Andererseits zählt hierzu das Datenmanagement, das Aktivitäten rund um den Lebenszyklus der Daten von der Erhebung über die Vorbereitung, Analyse, Visualisierung und den Zugriff umfasst [15].

3.3.3.3 Federation Modul

Das Federation Modul koordiniert das Zusammenspiel der angeschlossenen Instanzen der Architektur. Die Anforderungen sind ähnlich wie bei einem typischen System of Systems, jedoch unterstützen diese möglicherweise nicht das Ausmaß eines Big Data Systems [15].

3.3.3.4 Common Concerns Modul

Im Common Concerns Modul werden Bedenken über Skalierbarkeit, Verfügbarkeit, Datenorganisation, verwendete Technologien und Akkreditierung zusammengefasst.

3.4 Big Data Architecture for Automotive Applications

Die Zahl eingebetteter Sensoren, mit denen riesige Datenmengen aufgezeichnet werden, steigt stetig. Mit Hilfe dieser Daten können unter anderem Sicherheitssysteme der Autos weiter verbessert werden. Die anfallenden Datenmengen können durch herkömmliche Berechnungs- und Speicherkonzepte nicht effizient verwaltet werden. Für diesen Zweck hat die PSA-Gruppe, der zweitgrößte europäische Autohersteller, eine Big Data Architektur geschaffen. Die Architektur ist ein Schichtenmodell und besteht im Wesentlichen aus fünf Layern. Jeder dieser Layer wird im Folgenden vorgestellt [8].

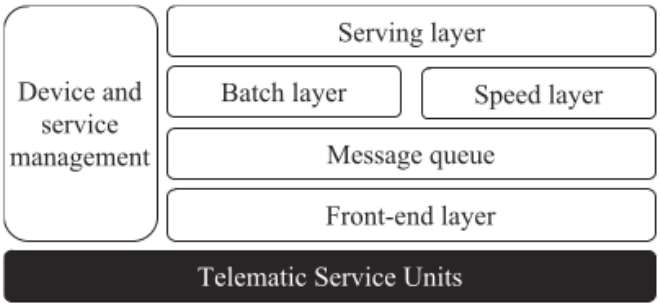

Abbildung 3.4: Bausteine der Big Data Architecture for Automotive Applications [8]

Die Architektur wird als Alternativarchitektur vorgestellt und anschließend bewertet, weil sie unter anderem durch Mitglieder der PSA-Gruppe erstellt wurde und demnach einen hohen Praxisbezug besitzt.

3.4.1 Device and Service Management Layer

Der Device and Service Management Layer beinhaltet alle Managementfunktionen, die als Referenzinformationen dienen und die Art der Verarbeitung und Transformation beeinflussen. Ferner ist eine Datenbank, die die Referenzinformationen enthält, und Webschnittstellen, die eine Interaktion mit der Datenbank ermöglichen, vorhanden. Fahrzeughalter, Administratoren und Partner erhalten Zugriff zu ihren jeweiligen Schnittstellen [8].

3.4.2 Frontend Layer

Je nach Kommunikationsprotokoll nimmt der Frontend Layer die Daten von den Telematic Service Units (TSU) entgegen und dekodieren diese. Als Kommunikationsprotokoll kann unter anderem Hypertext Transfer Protocol (HTTP) oder Hypertext Transfer Protocol Secure (HTTPS) verwendet werden. Anschließend werden die Daten in ein Format umgewandelt, das die nachfolgende Schicht, der Message Queue Layer, verarbeiten kann. Die Daten werden dann zur weiteren Verarbeitung an den Message Queue Layer weitergeleitet [8].

3.4.3 Message Queue Layer

Der Message Queue Layer dient der Bereitstellung eines belastbaren Datenfeeds. Es ist eine nachrichtenorientierte Middleware, die als Producer - Consumer Queue agiert. Dadurch soll eine asynchrone Kommunikation ermöglicht werden. Nachrichten werden vom Sender in die Queue gegeben, gespeichert und auf Abruf an den Empfänger weitergeleitet. Dies gewährleistet eine Abdeckung von Nichtverfügbarkeit des Senders oder Empfängers [8].

3.4.4 Speed Layer

Der Speed Layer ist für die Onlineverarbeitung der kontinuierlichen Datenflüsse zuständig. Um die Daten möglichst in Echtzeit zu verarbeiten, wird eine Stream Verarbeitung verwendet.
Des Weiteren werden die Rohdaten zur Speicherung und weiteren Analyse im Batch Layer vorbereitet. Hierzu zählt z.B. die Filterung und Anonymisierung von personenbezogenen Daten.
Da der Speed Layer das Stream Verarbeitungsparadigma verwendet und die funktionale Architektur der Stream Verarbeitung aus Microservices besteht, werden diese auch im Speed Layer verwendet.
Es folgt eine Auflistung der Microservices sowie ihrer jeweiligen Aufgaben.

- **Erfassung**
  Abrufen neuer Daten vom Message Queue Layer.
- **Dispatcher**
  Empfangen von Kunden- und Referenzdaten und Vorbereitung mittels Anonymisierung und Rollenzuweisungen.
- **Dienstleistungen**
  Erbringen von Mikrodienstleistungen für Kunden und Drittpartner.
- **Speicherung**
  Speicherung der Daten - jeder andere Microservice hat Zugriff auf den Speicher-Microservice.

Jeder eingehende Datensatz wird sofort entgegengenommen und durch alle Microservices verarbeitet. Damit es zu keiner Überlastung kommt, muss der Datensatz verarbeitet werden, bevor der nächste eintrifft. [8].

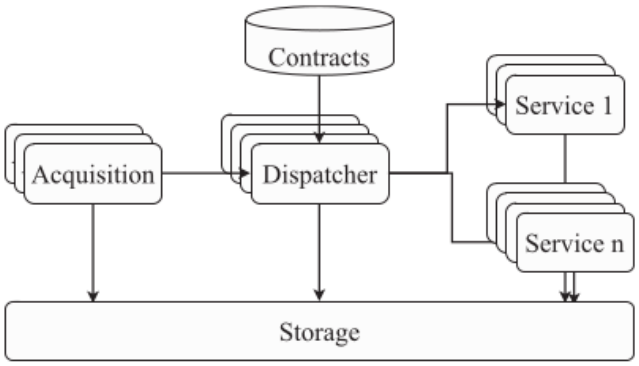

Abbildung 3.5: Funktionale Architektur des Speed Layers [8]

3.4.5 Batch Layer

Die Hauptaufgaben des Batch Layers sind die Batch Verarbeitung und die Speicherung der Daten. Hierbei bestehen nicht die Echtzeit-Verarbeitungseinschränkungen, die im Speed Layer vorhanden sind, weshalb nicht das Stream Verarbeitungsparadigma verwendet werden muss. Für die Implementierung eignet sich unter anderem das Open-Source-Framework Hadoop.
Durch die Nutzung von personalisierten Daten können diese nicht direkt gespeichert werden, sondern müssen erst vorverarbeitet werden. Die Vorverarbeitung findet im Speed Layer statt. Die Daten werden danach als unveränderlich abgespeichert. Da die Daten in hoher Menge vorliegen, ist die Nutzung eines skalierbaren Datensystems notwendig.
Während der Speed Layer neue Daten verarbeitet, sobald diese eingehen, ist der Batch Layer für die Verarbeitung bereits gespeicherter Daten verantwortlich. Die Verarbeitung ist häufig sehr zeitaufwändig. Deshalb ist die Verwendung eines nebenläufigen Programmiermodells nötig, um die Berechnungen so effizient wie möglich zu gestalten [8].

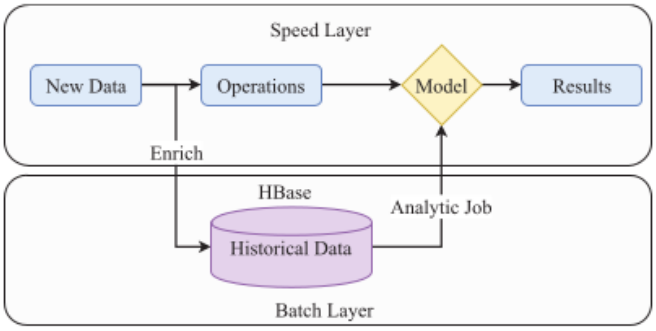

Abbildung 3.6: Zusammenarbeit von Speed- und Batch Layer [8]

3.4.6 Serving Layer

Um die verarbeiteten Daten aus dem Speed und Batch Layer zur Verfügung zu stellen, wird ein Webservice verwendet. Der Webservice bietet den Nutzern die APIs an, die benötigt werden, um die Daten abzufragen. Er leitet die Anfragen an die jeweiligen Datenbanken weiter. Zudem regelt der Serving Layer die Zugriffsrichtlinien, also welche Benutzergruppe auf welche Daten zugreifen darf. Bei der Speicherung der Daten ist eine Trennung zwischen den im Speed Layer erzeugten und den im Batch Layer gespeicherten Daten möglich und sinnvoll [8].

4 Implementierung der vorgestellten Big Data Architekturen

4.1 Allgemeines Vorgehen bei der Implementierung

Alle Implementierungen der vorgestellten Big Data Architekturen behandeln die Auswertung von Temperatur- und Luftfeuchtigkeitsdaten des Deutschen Wetterdienstes. Hierdurch soll eine gute Vergleichbarkeit der Architekturen gegeben werden. Auf diese Weise muss der Prozess der Analyse nur ein einziges Mal implementiert werden und kann danach wiederverwendet werden.
Alle Teile der Architekturen werden in der funktionalen Programmiersprache Elixir verfasst, da sie eine verteilte Implementierung und deshalb eine horizontale Skalierung ermöglicht. Darüber hinaus lassen sich die Anwendungen auch sehr gut vertikal skalieren, da problemlos hunderttausende leichtgewichtige Prozesse gleichzeitig auf einem Rechner ausgeführt werden können und systemweite Pausen auf ein Minimum reduziert werden [24]. Die Temperaturdaten werden in einer Apache Cassandra Datenbank gespeichert. Die Cassandra Datenbank wurde gewählt, da sie eine sehr gute Skalierbarkeit und Fehlertoleranz bietet [3] [23].
Zur Auswertung der Temperaturdaten werden ein Batch Processing Modul und ein Stream Processing Modul implementiert. Die Module werden in allen Architekturen verwendet, sodass der Aufwand möglichst gering bleibt.

4.1.1 Genutzte Daten

Die in den Big Data Architekturen ausgewerteten Daten sind Temperatur- und Luftfeuchtigkeitsdaten des Deutschen Wetterdienstes. Für die Auswertung werden folgende Attribute der Daten verwendet:

- Station
- Temperatur
- Luftfeuchtigkeit
- Jahr
- Monat
- Tag
- Stunde

Anhand der Station kann man den Ort der Temperaturmessung bestimmen. Die auszuwertenden Daten sind der Temperatur- und der Luftfeuchtigkeitswert. Jahr, Monat, Tag und Stunde beschreiben den Zeitpunkt der Messung. Zur einfacheren Abfrage werden sie getrennt voneinander gespeichert. Fehlende Messwerte sind mit -999 angegeben und werden in den Berechnungen nicht berücksichtigt [9].

4.1.2 Batch Processing Modul

Das Batch Processing Modul besitzt eine Schnittstelle zur Datenbank, in der die eingehenden Temperaturdaten eingefügt werden. Die Daten können einzeln oder als Liste hinzugefügt werden. Es gibt auch Schnittstellen zu einem oder mehreren Map Reduce Prozessen, in denen die Daten ausgewertet werden. Befinden sich Daten in der Datenbank, werden diese nach den Stationen abgefragt und alle Daten einer Station durch einen Map Reduce Prozess ausgewertet. Die Auswertung erfolgt verteilt. Ein Prozess kann mehrere Nodes ansprechen, die eine Map Funktion auf die Daten anwenden. Der Reduce Vorgang erfolgt in einer weiteren Node. Eine Node kann auf demselben Rechner oder auf einem anderen Rechner innerhalb des Netzwerks laufen, sodass eine Verteilung der Arbeitslast möglich ist. Hierfür muss auf dem Rechner jeweils nur der Code für die Auswertung und eine Client Version des Map Reduce Prozesses vorhanden sein. Die Node meldet sich am Map Reduce Prozess an und bekommt von diesem die Daten zur Berechnung zugewiesen. Sobald die Berechnung abgeschlossen ist, sendet die Node das Ergebnis zurück. Alle Ergebnisse werden zusammengefasst und an das Batch Processing Modul gesendet. Dieser fasst wiederum alle Ergebnisse zusammen und sendet sie an alle als Empfänger registrierten Prozesse.

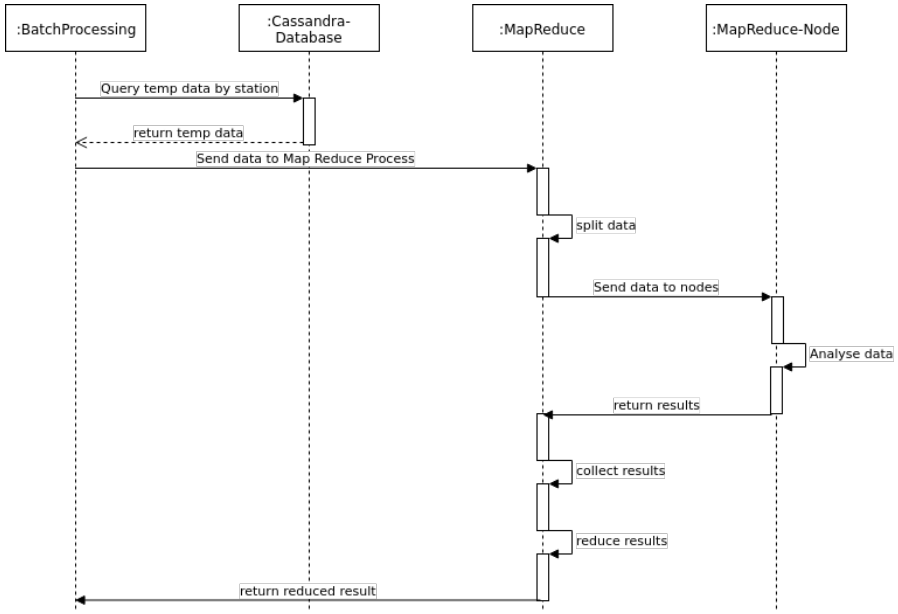

Abbildung 4.1: Sequenzdiagramm für die Analyse von Temperaturdaten mit dem Batch-Processing Modul

4.1.3 Stream Processing Modul

Im Stream Processing Modul werden die Daten einzeln verarbeitet und das Ergebnis direkt berechnet, ohne die Daten zu speichern. Das Ergebnis wird nach jedem Hinzufügen von Daten an die Prozesse weitergeleitet, die sich als Empfänger registriert haben. Die Daten können wie auch im Batch Processing Modul sowohl einzeln als auch als Liste hinzugefügt werden.

4.1.4 Mnesia Datenbank zur Speicherung der Ergebnisse der Analysen

Die Ergebnisse der Analysen werden in einer Mnesia Datenbank gespeichert. Diese in Erlang implementierte Datenbank erlaubt eine schnelle Suche der Daten. Außerdem kann das Schema bei laufendem System konfiguriert werden. Die Ergebnisse können sowohl persistent auf der Festplatte als auch im Hauptspeicher gesichert werden [11].

4.2 Implementierung der Lambda Architektur

Im Folgenden wird die Implementierung der Lambda Architektur beschrieben.
Batch und Speed Layer empfangen Daten aus den Datenquellen. Die Daten werden in beiden Schichten analysiert und die Ergebnisse werden an den Serving Layer weitergeleitet. Im Serving Layer werden beide Ergebnisse gespeichert und zur Abfrage bereitgestellt.

4.2.1 Batch Layer

Der Batch Layer verwendet das Batch Processing Modul. Er besitzt eine eigene Verbindung zur Datenquelle, die die Daten in einer geeigneten Form einliest. Die Daten werden über das Batch Processing Modul in die Cassandra Datenbank eingefügt. Durch das Batch Processing Modul werden die Daten analysiert. Die Ergebnisse werden direkt an den Serving Layer weitergeleitet. Ist die Analyse aller Daten abgeschlossen, wird sie sofort wiederholt, um auch neu hinzugefügte Daten zu berücksichtigen.

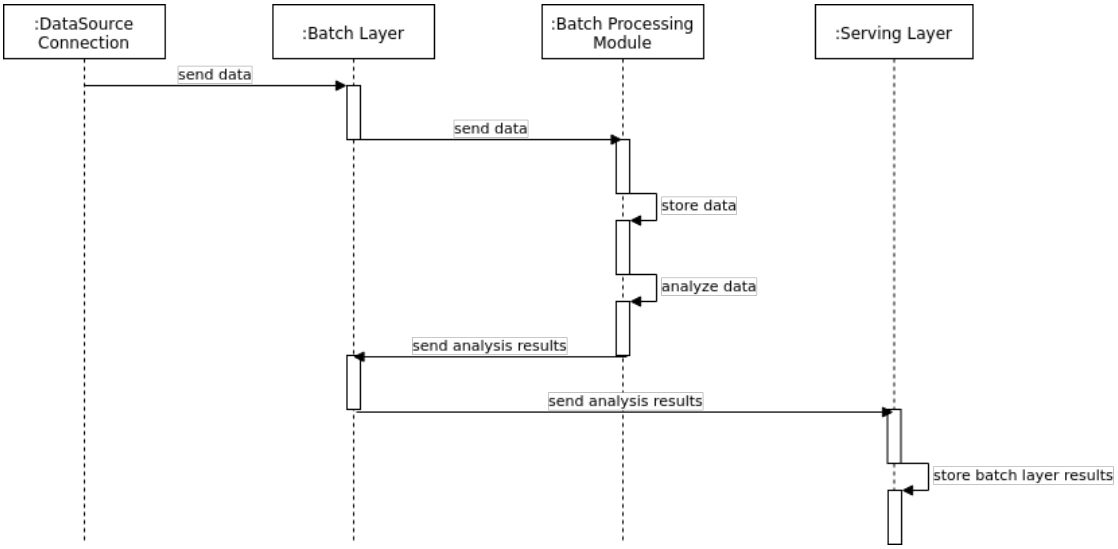

Abbildung 4.2: Sequenzdiagramm für den Batch Layer der Lambda Architektur

4.2.2 Speed Layer

Der Speed Layer nutzt das Stream Processing Modul zur Analyse der Daten. Neue Daten werden aus einer eigenen Verbindung zur Datenquelle eingelesen und in geeigneter Form

an das Stream Processing Modul weitergeleitet. Die Daten werden analysiert und das Ergebnis wird an den Serving Layer weitergegeben. Eine Berechnung findet nur statt, wenn neue Daten eingefügt werden.

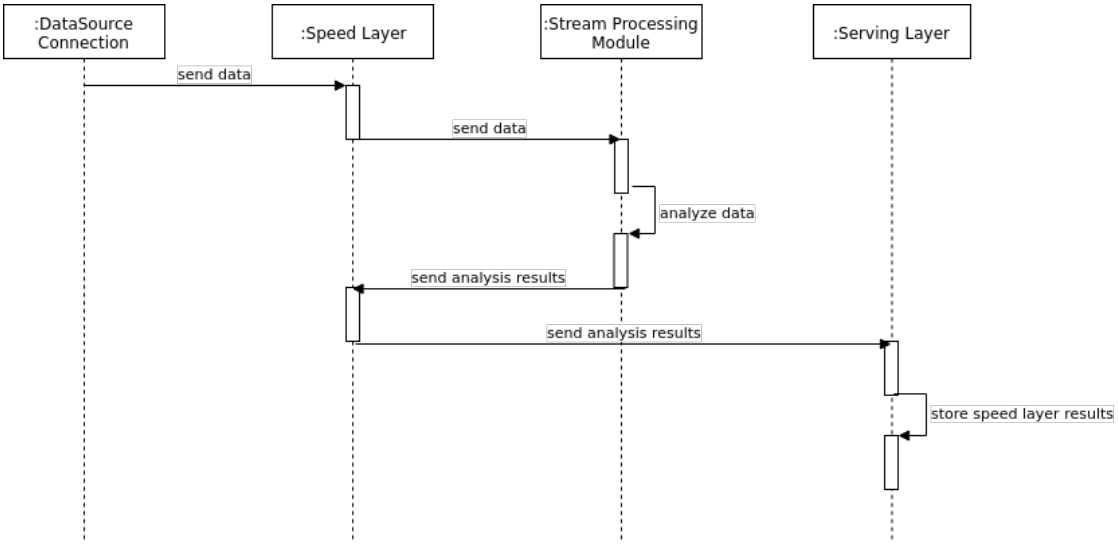

Abbildung 4.3: Sequenzdiagramm für den Speed Layer der Lambda Architektur

4.2.3 Serving Layer

Im Serving Layer gehen die Ergebnisse der Berechnungen vom Batch und Speed Layer ein. Die Ergebnisse werden je nach Layer, aus dem sie stammen, in einer Mnesia Datenbank gespeichert. Sollten neue Ergebnisse eintreffen, werden die alten überschrieben. Fragt ein Benutzer die Ergebnisse ab, wird zuerst geprüft, ob ein Ergebnis aus dem Batch Layer vorhanden ist. Ist keines vorhanden, wird das Ergebnis des Speed Layers zurückgegeben. Sollte auch hier kein Ergebnis vorhanden sein, wird ein NULL-Wert zurückgesendet. Ist ein Ergebnis aus dem Batch Layer vorhanden, wird geprüft, ob die Anzahl der berücksichtigten Elemente geringer als die Anzahl der Elemente des Ergebnisses aus dem Speed Layer sind. Sollte das der Fall sein, werden die Ergebnisse des Speed Layers zurückgegeben, da der Speed Layer die aktuelleren Werte liefern kann. Nur wenn im Ergebnis des Batch Layers mehr oder gleich viele Elemente berücksichtigt wurden, wird sein Ergebnis zurückgeliefert.

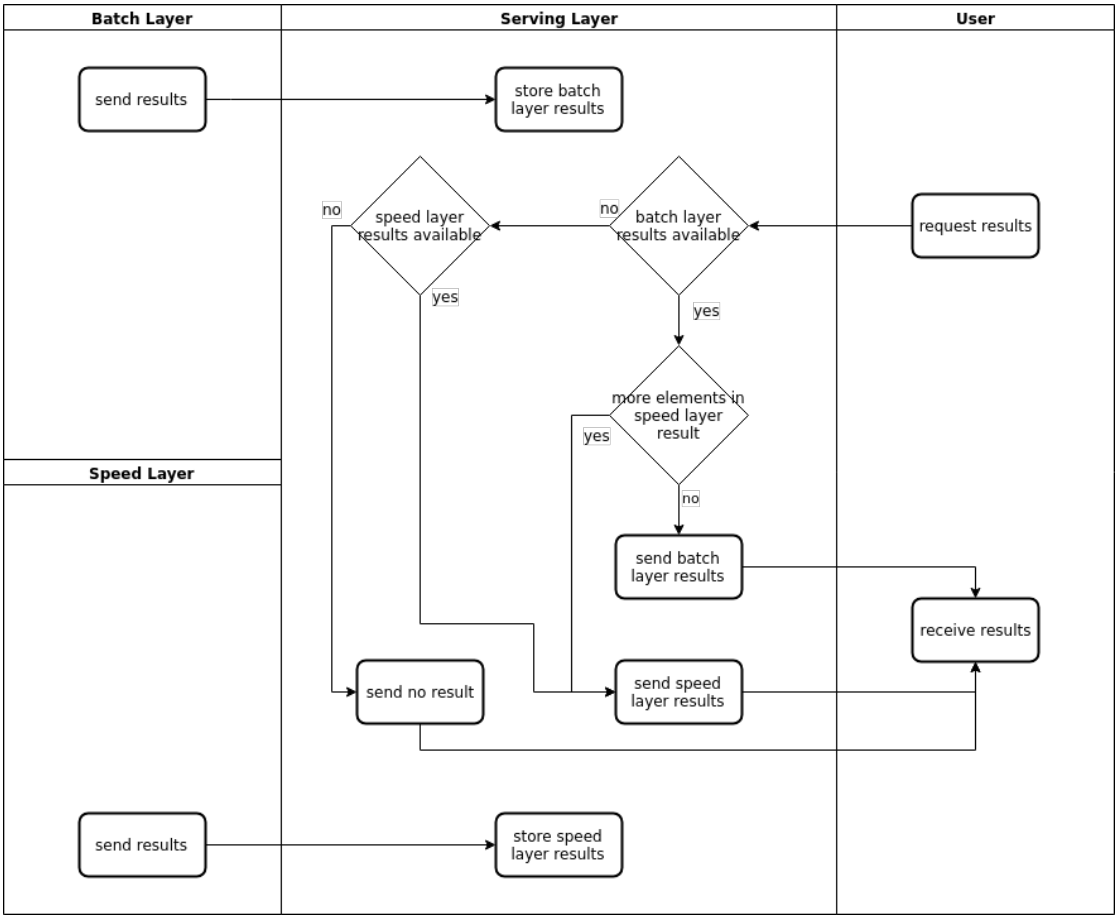

Abbildung 4.4: Ablaufdiagramm für den Serving Layer der Lambda Architektur

4.2.4 Aufwand der Implementierung

Der höchste Aufwand zur Implementierung lag im Batch Processing Modul, das zur Analyse der Daten im Batch Layer dient. Dieses Modul musste so implementiert werden, dass eine verteilte Berechnung auf mehreren Rechnern innerhalb eines Netzwerkes möglich ist. Durch die Möglichkeit der Prozesskommunikation auf einem oder mehreren Rechnern in einem Netzwerk, die durch Elixir erreicht wird, war dies unkompliziert umzusetzen [25]. Jedoch ist für die Analyse der Daten ein doppelter Aufwand entstanden, da sowohl im Batch Layer als auch im Speed Layer die Ergebnisse unabhängig voneinander berechnet werden. Die Speicherung im Serving Layer war mit etwas mehr Aufwand verbunden, da die Ergebnisse der Layer unterschieden werden müssen.

4.3 Implementierung der Learning Analytics Architecture

Die Learning Analytics Architecture nutzt ebenfalls das Batch Processing Modul zur Analyse der Daten.

4.3.1 Datenerhebungsdienst

Der Datenerhebungsdienst liest die Dateien, die in einem vorher angegebenen Ordner liegen, ein und sendet diese an das Datenspeicher- und Managementsystem.

4.3.2 Datenspeicher- und Managementsystem

Im Datenspeicher- und Managementsystem gehen die im Datenerhebungsdienst erfassten Daten ein. Die Daten werden über das Batch Processing Modul in die Cassandra Datenbank eingefügt. Vom Batch Processing Modul aus wird ebenfalls die Analyse der Daten mit dem Map Reduce Prozess angestoßen. Die Ergebnisse der Analysen werden entgegengenommen und in die Mnesia Datenbank eingefügt.

4.3.3 Datenanalysesystem

Das Datenanalysesystem wird vollständig durch den Map Reduce Prozess des Batch Processing Moduls übernommen. Die Daten werden aufgeteilt und durch mehrere Nodes analysiert. Die Teilergebnisse der Nodes werden am Ende zusammengefügt und an das Batch Processing Modul gesendet. Da die Daten durch das Batch Processing Modul ebenfalls aufgeteilt wurden, müssen die Ergebnisse wieder zusammengefasst werden.

4.3.4 Datenvisualisierungssystem

Das Datenvisualisierungssystem wird nicht implementiert. Die Ergebnisse können stattdessen direkt über das Hauptmodul abgefragt werden.

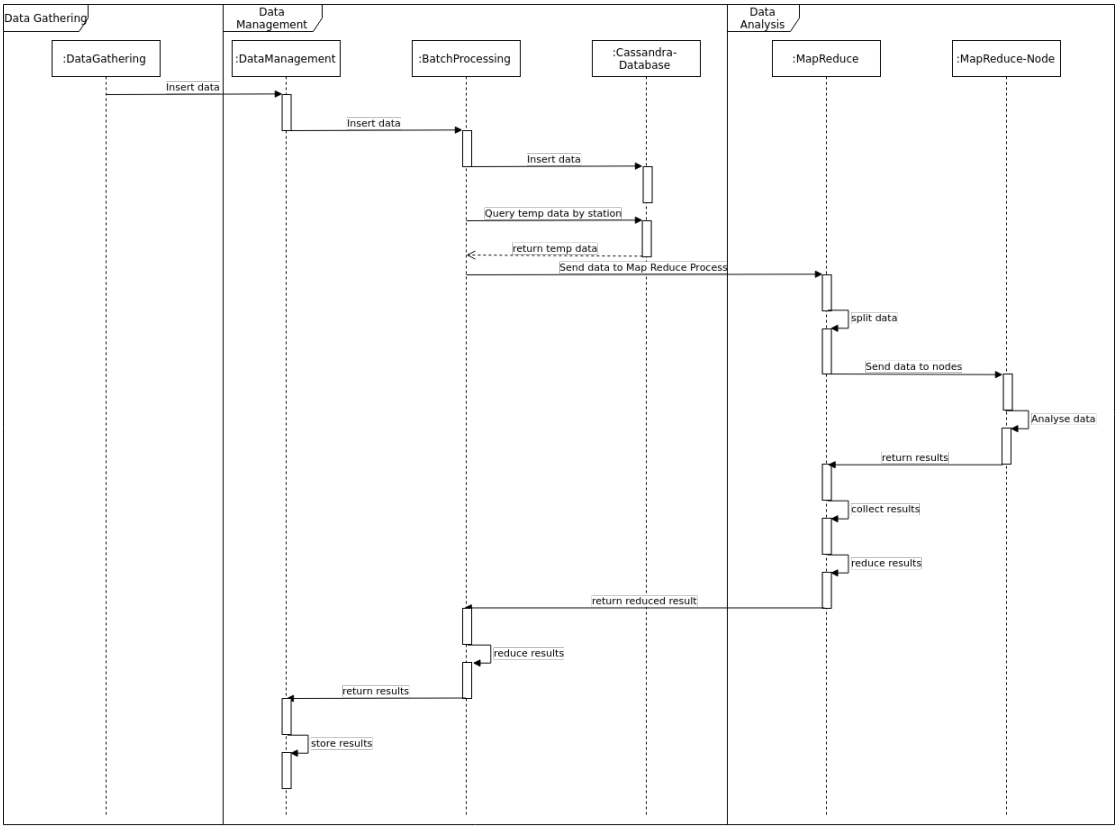

Abbildung 4.5: Sequenzdiagramm für die Learning Analytics Architecture

4.3.5 Aufwand der Implementierung

In der Gesamtbetrachtung der Architektur war die Implementierung des Analyseteils am aufwendigsten. Da dieser Teil bereits durch die Implementierung des Batch Processing Moduls vorhanden war, konnte die Architektur relativ schnell umgesetzt werden.

4.4 Implementierung der Reference Architecture for Big Data Systems in the National Security Domain

Die Reference Architecture for Big Data Systems in the National Security Domain besteht aus insgesamt drei Hauptmodulen. Das Big Data Application Provider Modul bildet den gesamten Prozess des Big Data Systems von der Datengewinnung über die Analyse bis hin zur Bereitstellung der Ergebnisse für Benutzer ab.

Das Big Data Framework Provider Modul stellt die dazu benötigten Hilfsfunktionen, wie z.B. eine Datenbankanbindung oder ein Modul für einen zuverlässigen Austausch der Nachrichten zwischen den Modulen.
Das Cross-Cutting Modul sorgt unter anderem für die Sicherheit im Big Data System und kümmert sich um das System- und Datenmanagement, indem es die Bereitstellung und die Kontrolle der Infrastruktur übernimmt und alle Aktivitäten rund um den Lebenszyklus der Daten umfasst. Dieses Modul wird nicht implementiert.

4.4.1 Big Data Application Provider Modul

Das Big Data Application Provider Modul besteht aus sechs Teilmodulen, die ähnlich einer Schichtenarchitektur angeordnet sind und zusammenarbeiten. Die Teilmodule kommunizieren über das Message Modul des Big Data Framework Provider Moduls.

4.4.1.1 Application Orchestration Modul

Im Application Orchestration Modul werden die unterschiedlichen Teilmodule konfiguriert, gestartet und die jeweiligen Nachfolger mit den Teilmodulen verknüpft. Es erfolgt die Konfiguration im Quellcode.

4.4.1.2 Collection Modul

Das Collection Modul ist die Schnittstelle zu den Datenquellen. Bei der Datenquelle handelt es sich um einen Ordner mit Zip-Dateien, die eingelesen werden. Nachdem eine Datei eingelesen wurde, werden alle Zeilen der Datei in einer Liste an das Preparation Modul gesendet.

4.4.1.3 Preparation Modul

Im Preparation Modul gehen die eingelesenen Daten aus dem Collection Modul ein. Eine eingelesene Datei wird durch eine Liste mit allen Zeilen dargestellt. Jede Zeile in dieser Liste wird in einen Datensatz umgewandelt, der von dem Data Storage Modul des Big Data Framework Provider Moduls verarbeitet werden kann. Die umgewandelten Datensätze werden als Liste weitergeleitet.

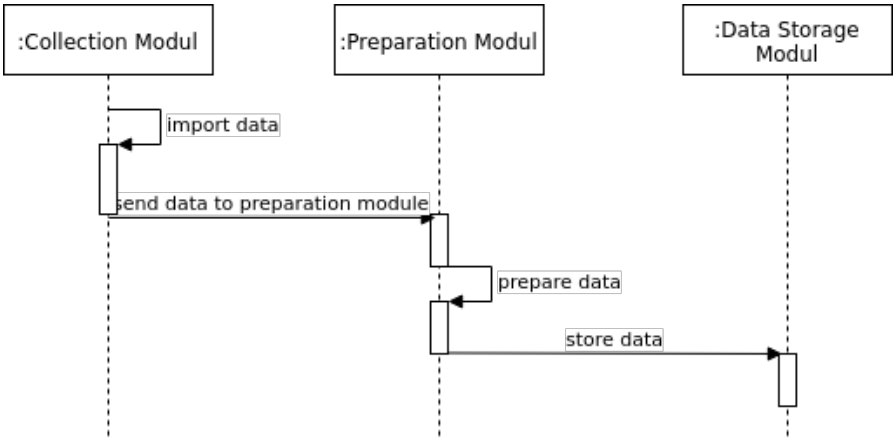

Abbildung 4.6: Sequenzdiagramm für das Collection und Preparation Modul der Reference Architecture for Big Data Systems in the National Security Domain

4.4.1.4 Analytics Modul

Das Analytics Modul startet das Batch Processing Modul und nimmt die Ergebnisse entgegen. Sie werden an das Data Storage Modul gesendet, um schließlich in der Mnesia Datenbank gespeichert zu werden.

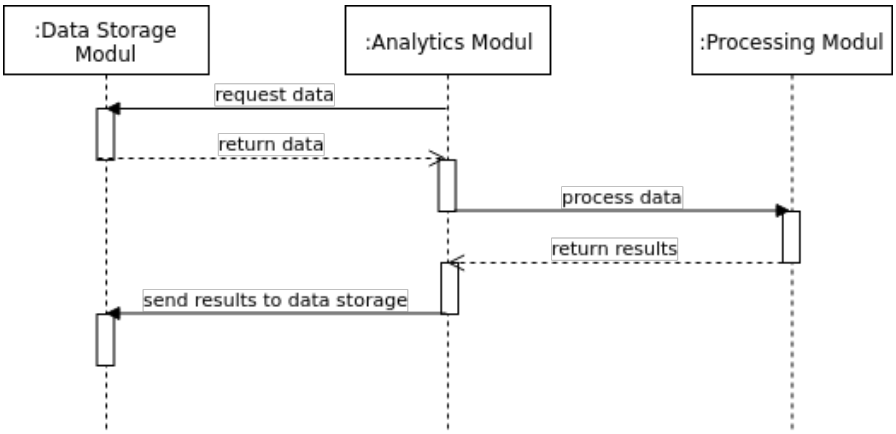

Abbildung 4.7: Sequenzdiagramm für das Analytics Modul der Reference Architecture for Big Data Systems in the National Security Domain

4.4.1.5 Visualisation Modul

Das Visualisation Modul wird nicht implementiert.

4.4.1.6 Access Modul

Das Access Modul bildet eine Schnittstelle zum Data Storage Modul. Der Benutzer kann die Ergebnisse der Analysen abfragen. Die Ergebnisse werden direkt und nicht über das Message Modul vom Data Storage Modul abgefragt. Dadurch ist eine direkte Bereitstellung der Ergebnisse möglich.

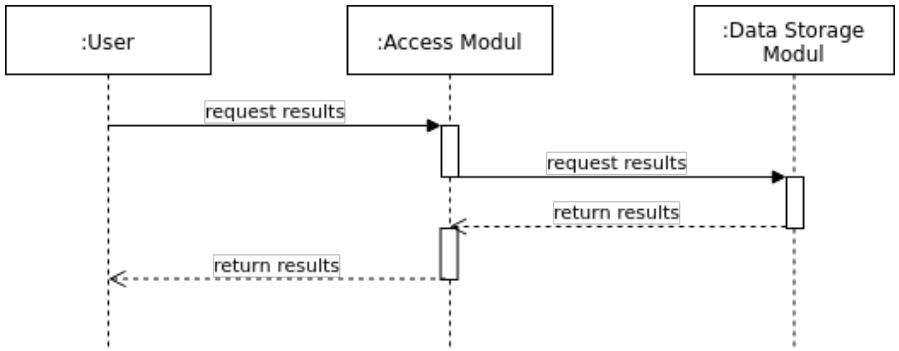

Abbildung 4.8: Sequenzdiagramm für das Access Modul der Reference Architecture for Big Data Systems in the National Security Domain

4.4.2 Big Data Framework Provider Modul

Das Big Data Framework Provider Modul stellt Module bereit, um die Daten einzulesen, zu speichern und zu verarbeiten.

4.4.2.1 Processing Modul

Das in Kapitel 4.1.2 beschriebene Batch Processing Modul wird als Processing Modul verwendet.

4.4.2.2 Messaging Modul

Alle Module, die Nachrichten über das Messaging Modul empfangen sollen, müssen sich mit einer ID und einem Secret registrieren. Sind die Module registriert, können andere Module Nachrichten hinterlegen. Die Nachrichten müssen von den empfangenden Modulen abgerufen werden. Nachrichten sind nach einem Abruf nicht mehr verfügbar. Deshalb

wird das Secret benötigt, damit die Nachrichten nicht von anderen Modulen abgerufen werden können.

4.4.2.3 Data Storage Modul

Das Data Storage Modul besteht aus zwei Teilen. Ein Teil kümmert sich um die Temperaturdaten, die vom Preparation Modul vorbereitet und in einer Cassandra Datenbank gespeichert werden. Die Daten werden vom Processing Modul für die Berechnung der Analysen genutzt.
Der zweite Teil speichert die Ergebnisse, die vom Analytics Modul zugesendet werden und im Access Modul für die Benutzer abrufbar sind. Die Ergebnisse werden in einer Mnesia Datenbank gespeichert.
Da hier ein erhöhtes Nachrichtenaufkommen besteht, hat jeder Teil eine eigene Verbindung zum Message Modul.

4.4.2.4 Infrastructure Modul

Das Infrastructure Modul wird nicht implementiert.

4.4.3 Cross-Cutting Modul

Das Cross-Cutting Modul wird in dieser Implementierung nicht weiter berücksichtigt.

4.4.4 Aufwand der Implementierung

Obwohl dieses Modul in der Architektur am umfangreichsten erschien, war der Aufwand für die Implementierung nicht höher. Da alle Module ähnlich wie in einer Microservice-Architektur nur kleine Aufgaben wahrnehmen und die Daten sofort an das nächste Modul weiterleiten, war die Implementierung sehr gut umsetzbar. Die Wartbarkeit des Systems ist gut, da jede Aufgabe von einem Modul übernommen wird und man auftretende Fehler schneller finden kann. Allerdings ist die Implementierung aufwendiger, wenn die fehlenden Teile des Cross-Cutting Moduls hinzugefügt werden und z.B. Eindringversuche im Security Modul erkannt und analysiert werden sollen.

4.5 Implementierung der Big Data Architecture for Automotive Applications

Die Big Data Architecture for Automotive Applications nutzt wie die Lambda Architektur sowohl einen Batch als auch einen Speed Layer. Des Weiteren gibt es Module, die das Einlesen der Daten, einen Nachrichtenaustausch der Layer und die Abfrage von Daten durch Stakeholder ermöglichen.

4.5.1 Device and Service Management Layer

Im Device and Service Management Layer sollen Fahrzeughalter, Administratoren und Partner Zugriff auf in einer Datenbank gespeicherten Referenzinformationen erhalten. Da keine Fahrzeugdaten analysiert werden, ist diese Datenbank nicht notwendig und es werden keine weiteren Daten gespeichert. Der Device and Service Management Layer dient dazu, alle anderen Layer zu starten. Außerdem bildet er die Schnittstelle, die die Abfrage der Ergebnisse aus den Analysen ermöglicht.

4.5.2 Frontend Layer

Der Frontend Layer liest die Temperaturdatensätze aus Dateien in einem angegebenen Ordner ein und leitet sie als Liste von Zeilen der Datei über die Message Queue an den Speed Layer weiter.

4.5.3 Message Queue Layer

Die Layer der Architektur melden sich mit einer ID und einem Passwort an der Message Queue an. Nach erfolgreicher Anmeldung können dort Nachrichten für die Layer mittels der ID hinterlegt werden. Um die Nachrichten abfragen zu können, wird das richtige Passwort benötigt. Nachdem eine Nachricht abgefragt wurde, wird diese aus dem System entfernt. Die Nachrichten werden so gespeichert, dass die älteste vorhandene Nachricht als erstes an den anfragenden Layer gesendet wird. Ist keine Nachricht mehr vorhanden, wird bei der Abfrage eine Meldung gesendet, dass sich keine weiteren Nachrichten in der Queue befinden.

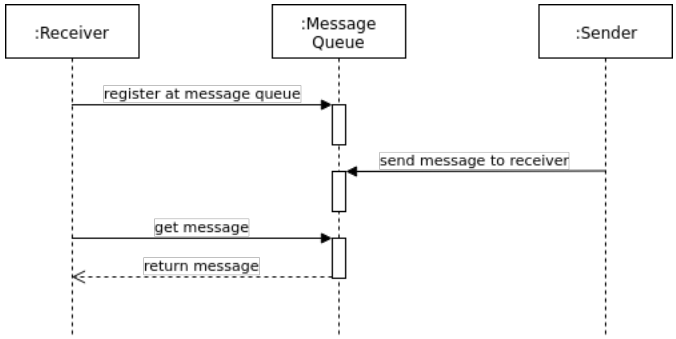

Abbildung 4.9: Sequenzdiagramm für die Messsage Queue der Big Data Architecture for Automotive Applications

4.5.4 Speed Layer

Der Speed Layer fragt permanent bei der Message Queue nach neuen Nachrichten. Es werden Daten als Liste von Zeilen durch den Frontend Layer gesendet. Die Zeilen werden in ein Format umgewandelt, das weiterverarbeitet werden kann. Die vorverarbeiteten Daten werden an den Batch Layer und an das Stream Processing Modul weitergeleitet. Die Ergebnisse werden über die Message Queue an den Serving Layer gesendet.

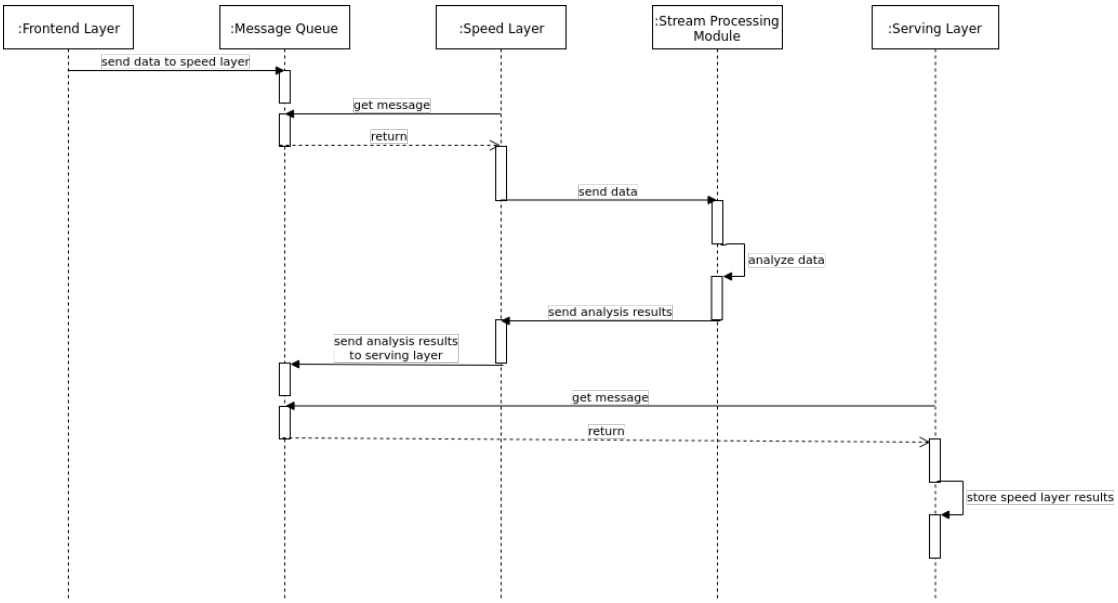

Abbildung 4.10: Sequenzdiagramm für den Speed Layer der Big Data Architecture for Automotive Applications

4.5.5 Batch Layer

Vom Speed Layer eingehende Daten werden zur Analyse an das Batch Processing Modul gesendet, wo sie in der Cassandra Datenbank gespeichert und verteilt analysiert werden. Die Ergebnisse dieser Analysen werden mittels Message Queue an den Serving Layer weitergegeben.

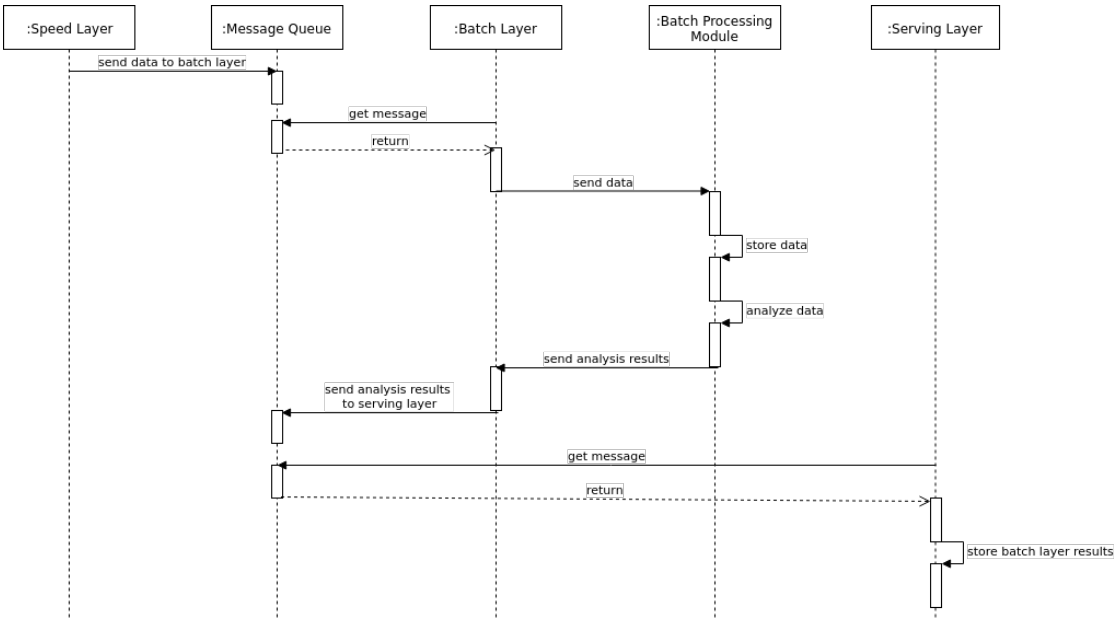

Abbildung 4.11: Sequenzdiagramm für den Batch Layer der Big Data Architecture for Automotive Applications

4.5.6 Serving Layer

Die Ergebnisse vom Batch und Speed Layer werden durch die Message Queue an den Serving Layer übertragen, an die Mnesia Datenbank weitergeleitet und dort gespeichert. Sobald die Ergebnisse abgefragt werden, wird zuerst überprüft, ob ein Ergebnis aus dem Batch Layer vorhanden ist. Sollte ein Ergebnis vorhanden sein, wird das äquivalente Ergebnis des Speed Layers überprüft. Das Ergebnis mit den meisten berücksichtigten Elementen wird zurückgeliefert. Ist das Ergebnis des Batch Layers nicht vorhanden, wird das Ergebnis vom Speed Layer zurückgegeben. Sollte auch hier kein Ergebnis vorhanden sein, wird eine Fehlermeldung gesendet.

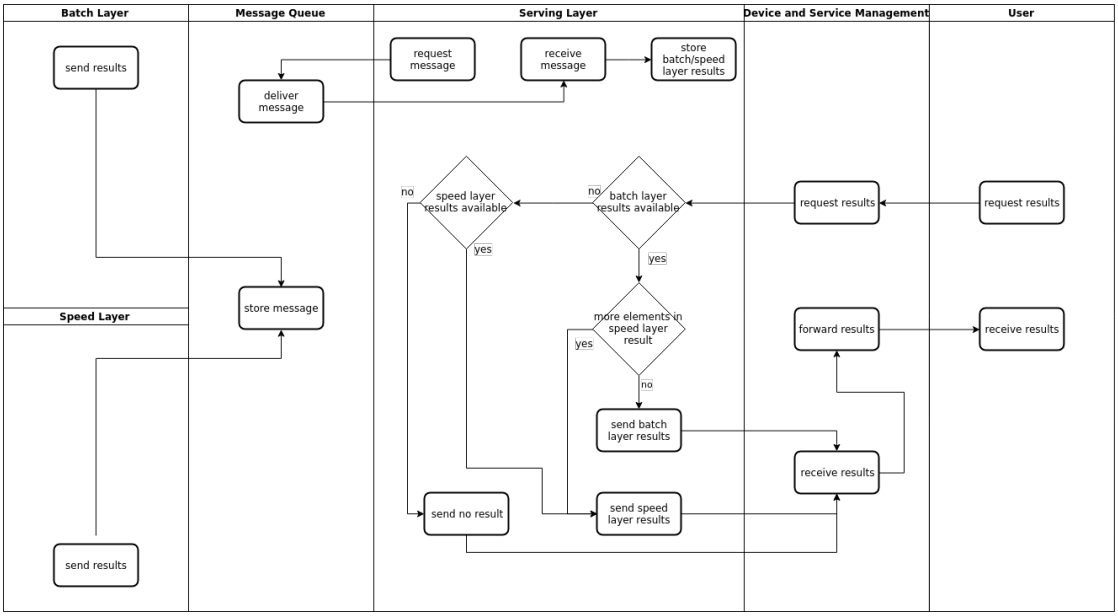

Abbildung 4.12: Ablaufdiagramm für den Serving Layer der Big Data Architecture for Automotive Applications

4.5.7 Aufwand der Implementierung

Der Aufwand der Implementierung dieser Architektur war im Vergleich zu der Lambda Architektur ähnlich. Es musste lediglich eine Importfunktion für die Daten geschrieben werden.

5 Bewertung der vorgestellten Big Data Architekturen

In diesem Kapitel erfolgt die Bewertung der Big Data Architekturen. Zunächst werden Methoden vorgestellt und danach die geeignetste ausgewählt. Anhand dieser Methode wird jede einzelne Architektur bewertet. Das Ergebnis der Bewertung fließt in das Endergebnis ein, das in Kapitel 6 dargelegt wird.

5.1 Vorstellung von Bewertungsmethoden für Softwarearchitekturen

Nachfolgend werden ausgewählte Methoden zur Bewertung von Softwarearchitekturen vorgestellt.

5.1.1 Architecture Tradeoff Analysis Method

Die Architecture Tradeoff Analysis Method (ATAM) besteht aus neun Schritten, die der Reihe nach mit allen Teilnehmern durchgeführt werden. Teilnehmer beim ATAM sind Vertreter aus allen Projektparteien - Kunde, Geschäftsführung, Marketingabteilung, IT-Abteilung, Qualitätssicherungsabteilung sowie die Betreuer des Projekts. [1] Die Schritte gliedern sich folgendermaßen:

1. Präsentation der ATAM, damit alle Teilnehmer die Methode verstehen und akzeptieren.
2. Vorstellung der wichtigsten Projektziele aus geschäftlicher Sicht.

3. Präsentation über die Projektsoftwarekomponenten, die Teil der Softwarearchitektur sind und miteinander agieren.

4. Identifizierung von Softwarearchitekturansätzen anhand des vorliegenden Designs und den mit der Unified Modeling Language (UML) gebildeten Diagrammen zur Architektur.

5. Generieren des Utility Trees, der als eine Darstellungsform für die Einflussfaktoren dient. Der Utility Tree enthält in der ersten Ebene alle Faktoren, in der zweiten die Kategorien der einzelnen Faktoren und in der dritten Ebene als Blätter die Szenarien aus den einzelnen Kategorien. Diese Szenarien werden mittels zwei Dimensionen priorisiert. Die erste Dimension beschreibt die Wichtigkeit des Szenarios für den Erfolg des Systems. Die zweite Dimension beschreibt die Schwierigkeit, das Szenario im System umzusetzen. Alle Szenarien, die mindestens in einer Dimension mit hoch priorisiert wurden, müssen behandelt werden. Alle Szenarien mit mindestens einer niedrigen Priorisierung werden aus Kostengründen nicht weiter betrachtet.

6. Analyse von Softwarearchitekturansätzen, bei denen die in Schritt 5 gebildeten Szenarien mit den Architekturansätzen verlinkt werden. Das Ziel ist, zu belegen, dass das Szenario mit der Architektur umgesetzt werden kann. In diesem Schritt werden die Risiken der Softwarearchitektur herausgearbeitet. Ein Risiko ist hierbei ein Architekturaspekt, der sich während der Analyse als problematisch herausstellt.

7. In diesem Schritt werden nochmals Szenarien erstellt, diesmal jedoch in einem anderen, größeren Personenkreis. Diese Szenarien können mit denen in Schritt 5 erstellten übereinstimmen. Nach der Erstellung werden die Szenarien wieder durch die Ersteller mittels Abstimmung priorisiert. Unterscheidet sich die Priorisierung der Szenarien von gleichen, im Utility Tree ebenfalls enthaltenen Szenarien zu stark, ist das ein Indiz für ein weiteres Risiko.

8. Die in Schritt 7 erstellten Szenarien werden ähnlich wie in Schritt 6 untersucht. Im Idealfall wiederholt sich die Argumentation, die auch in Schritt 6 genutzt wurde. Treten neue Aspekte auf, müssen diese genauer untersucht werden. Das dient zur Kontrolle, ob in den vorangegangenen Schritten nichts vergessen, übersehen oder falsch eingeschätzt wurde.

9. Im letzten Schritt werden die Ergebnisse für alle beteiligten Personen zusammengefasst und präsentiert. Zudem wird in der Nachbearbeitung ein ausführlicher Abschlussbericht erstellt.

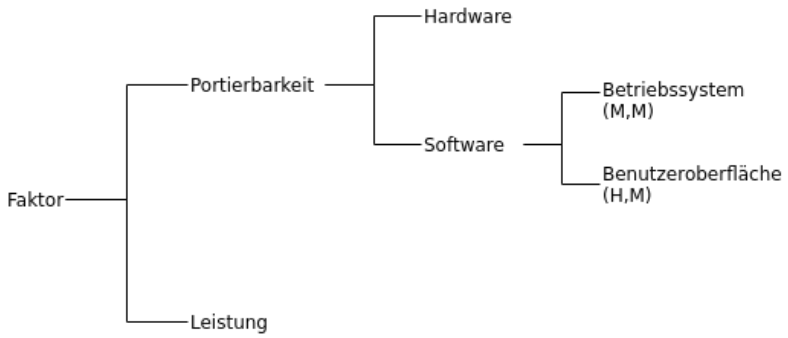

Abbildung 5.1: Beispiel eines Utility Trees [26]

Wenn die Methode früh genug angewendet wird, lassen sich Risiken innerhalb der Architektur zu einem Zeitpunkt bestimmen, an dem sie noch leicht und kostengünstig entfernt oder umgangen werden können [5, 26].

Das ATAM ist eine sehr aufwendige Methode, die vor allem viel Zeit und die Partizipation von Teilnehmern aus unterschiedlichen Bereichen erfordert [5].

5.1.2 Software Architecture Analysis Method

Die Software Architecture Analysis Method (SAAM) ist eine Methode zur Analyse von Softwarearchitektur, bei der der Fokus auf der Abschätzung von Wartbarkeit und funktionalen Anforderungen liegt. Andere nichtfunktionale Anforderungen (NFA) werden bei der Betrachtung auf Sicherheit, Modifizierbarkeit und Portierbarkeit beschränkt. Die Methode ist besonders gut geeignet, um Systeme mit vergleichbarer Funktionalität zu bewerten. Jedoch ist sie für die Bewertung anhand vieler NFA eher ungeeignet.

Die SAAM gliedert sich in folgende Schritte:

1. **Beschreibung der Architektur**
   Zuerst wird die zu bewertende Architektur so beschrieben, dass alle an der Bewertung teilnehmenden Personen diese verstehen. Die Beschreibung sollte folgende Elemente beinhalten:

   - Komponenten und Datenelemente
   - Verbindungen zwischen diesen

- Beschreibung des Systemverhaltens

Die Beschreibung der Architektur sollte die teilnehmenden Personen dazu animieren, sich mögliche Szenarien auszudenken.

2. **Erheben möglicher Szenarien**
   Szenarien sind die Tätigkeiten, die das System unterstützen soll. Deshalb ist es notwendig, dass die Personen, die diese Szenarien erstellen, möglichst aus dem Anwenderkreis des zu entwickelnden Produktes stammen (z.B Auftraggeber, Benutzer, Administrator, Entwickler, Wartungspersonal, Marketing). Die Szenarien werden in einem Meeting von den entsprechenden Personen gemeinsam erstellt. Wird für ein Szenario eine umfassendere Architekturbeschreibung benötigt, muss diese erstellt werden.

3. **Klassifizieren und Priorisieren der Szenarien**
   Die Einteilung der Szenarien erfolgt in zwei Kategorien:

   a) Direkte Szenarien
      Szenarien, die mit der aktuellen Architektur ausgeführt werden können, ohne das Änderungen vorgenommen werden müssen.

   b) Indirekte Szenarien
      Szenarien, die Änderungen an der Softwarearchitektur zur Folge haben, damit diese ausgeführt werden können.

   Darüber hinaus können die Szenarien priorisiert werden, damit die Architekturbewertung effizienter gestaltet werden kann. Nur die am höchsten priorisierten Szenarien werden genauer untersucht. Zur Priorisierung wird eine offene Abstimmung unter den teilnehmenden Personen vorgeschlagen.

4. **Bewertung jedes einzelnen Szenarios**
   Die vorher erstellten Szenarien werden nun den betroffenen Elementen der Architektur zugeordnet. Bei einem direkten Szenario wird die Ausführung durch das System beschrieben. Für ein indirektes Szenario müssen die für die Ausführung nötigen Änderungen an der Architektur beschrieben und der Aufwand dieser Änderungen abgeschätzt werden.

5. **Untersuchung der Szenariointeraktionen**
   Eine Szenariointeraktion besteht, wenn eine Änderung an einer Komponente der

Architektur durch mindestens zwei Szenarien erforderlich ist. Liegt eine hohe Szenariointeraktion vor, kann das auf folgende zwei Problemstellungen hindeuten:

a) Eine Komponente bildet nicht zusammengehörige Funktionsbereiche ab.

b) Die Architektur einer Komponente ist nicht hinreichend genau dokumentiert. Hier sollte eine Nachbesserung der Beschreibung erfolgen.

6. **Erstellen der Gesamtbewertung**
   Die bewerteten Szenarien können anhand ihres Bezugs zur relativen Bedeutung für den Erfolg des Systems gewichtet werden. Durch die gewichteten Szenarien kann eine Gesamtbewertung der Architektur erfolgen. Mittels der Gewichtung der Szenarien können Architekturen verglichen und so ein Ranking erstellt werden.

Die SAAM wurde ursprünglich entwickelt, um die Modifizierbarkeit von Architekturen zu überprüfen. Allerdings hat sich in der Anwendung der Methode gezeigt, dass sie sich sehr gut dafür eignet, die funktionale Abdeckung und andere Qualitätsattribute wie Portierbarkeit, Erweiterbarkeit und die Integrierbarkeit zu überprüfen [7].

5.1.3 Bewertung anhand der Priorisierung nichtfunktionaler Anforderungen

Die Bewertung anhand von NFA ist ein möglicher Ansatz zum Vergleich von Architekturen, die unterschiedliche funktionale Anforderungen besitzen. Typische NFA sind:

- Benutzbarkeit
- Zuverlässigkeit
- Korrektheit
- Sicherheit
- Wartbarkeit
- Wirtschaftlichkeit
- Skalierbarkeit

Diese Anforderungen sind gleichzeitig Qualitätsmerkmale der Architekturen.
Die Erfüllung einer NFA kann teilweise in Widerspruch zu der Erfüllung einer anderen Anforderung stehen. So steht die Erfüllung der Anforderung Sicherheit im Widerspruch zu der Anforderung Einfachheit. Daher ist es sinnvoll, die NFA nach den jeweiligen Erwartungen zu priorisieren und anhand dieser Priorisierung eine Scoring Tabelle zu erstellen. Hierzu muss jedoch nicht nur eine Priorisierung vorgenommen werden, sondern auch eine Unterteilung, wann eine Architektur eine Anforderung in welchem Grad erfüllt. Es kann ein prozentualer Werteanteil verwendet werden. Wichtig ist, dass die Kriterien zur Erfüllung der Anforderungen präzise messbar sind [2].

Tabelle 5.1: Beispiel für eine Scoring Tabelle [2]

Priorität	NFA	Erfüllungsgrad			
		100% - 85 %	84% - 50 %	49% - 25 %	< 24 %
		Hoch	Mittel	Gering	Nicht
1	Wartbarkeit	21	14	7	0
2	Benutzbarkeit	18	12	6	0
3	Betreibbarkeit	15	10	5	0
4	Performance	12	8	4	0
5	Testbarkeit	9	6	3	0
6	Stabilität	6	4	2	0
7	Portierbarkeit	3	2	1	0
	Score max.				84

5.2 Auswahl eines geeigneten Bewertungsschemas

Da die ATAM eine sehr aufwendige Methode ist und vor allem viel Zeit und Partizipation von unterschiedlichen Teilnehmern benötigt, ist sie für diese Arbeit ungeeignet. Eine Mischform aus der SAAM und der Bewertung anhand einer Priorisierung nichtfunktionaler Anforderungen ist am meisten geeignet, da die Abläufe anhand der Szenarien mit der SAAM und die in Kapitel 2.1.3 vorgestellten Anforderungen an ein Big Data System mit der Bewertung anhand der Priorisierung nichtfunktionaler Anforderungen sehr gut bewertet werden können.

5.2.1 Szenarien zur Bewertung der Architekturen

Der zweite Schritt der SAAM sieht vor, dass mögliche Szenarien erhoben werden sollen. Da sich die Architekturen in ihrer Funktionalität sehr stark ähneln, können zur Bewertung der einzelnen Architekturen dieselben Szenarien verwendet werden. Folgende Szenarien werden für diese Bewertung als geeignet angesehen:

1. Hinzufügen von Daten aus den Datenquellen in das Big Data System.
2. Korrigieren von fehlerhaften Daten und die entsprechende Korrektur der bisher erstellten Analysen.
3. Löschen von Daten innerhalb des Big Data Systems, z.B. aus Datenschutzgründen.
4. Abfragen von Ergebnissen aus den Auswertungen.
5. Abfragen von einzelnen Datensätzen, um beispielsweise Ausreißer zu identifizieren.
6. Hinzufügen einer neuen Methode zur Auswertung der Daten.
7. Ergänzen der Datensätze um neue Attribute.

Zu Beginn sollte die Klassifizierung der einzelnen Szenarien stattfinden. Ein Szenario wird entweder als direktes oder als indirektes Szenario klassifiziert. Direkte Szenarien können mit der vorhandenen Architektur umgesetzt werden, ohne dass eine Änderung erfolgen muss. Indirekte Szenarien benötigen eine Änderung an der Architektur. Zunächst muss für jedes Szenario der Ablauf in der Architektur überprüft werden. Das geschieht mit Hilfe von Ablauf- oder Sequenzdiagrammen. Die Diagramme werden in den jeweiligen Kapiteln zur Bewertung der einzelnen Architekturen erzeugt und kategorisiert. Anhand der Kategorisierung werden die indirekten Szenarien durch den Aufwand der Änderung an der Architektur, den ein Szenario verursacht, bewertet. Danach erfolgt eine Bewertung der Szenariointeraktionen und die Erstellung der Gesamtbewertung. Diese Schritte werden in den jeweiligen Kapiteln zur Bewertung der Architekturen dargestellt.

Für die Gesamtbewertung wird ein Punktesystem verwendet. Die Punkte werden für die Szenarien folgendermaßen verteilt:

Tabelle 5.2: Punkteverteilung für die Szenarien

Szenario	**Punkte**
Direktes Szenario	0
Indirektes Szenario niedriger Aufwand	1
Indirektes Szenario mittlerer Aufwand	3
Indirektes Szenario hoher Aufwand	5

5.2.2 Priorisierung der Anforderungen an ein Big Data System

Um eine Bewertung anhand der NFA vorzunehmen, müssen zuerst die NFA ausgewählt und priorisiert werden. Bei den genutzten NFA handelt es sich um die in Kapitel 2.1.3 vorgestellten:

- Belastbarkeit und Fehlertoleranz
- Lesen und Aktualisieren mit geringer Latenzzeit
- Skalierbarkeit
- Allgemeingültigkeit
- Erweiterbarkeit
- Ad-Hoc-Abfragen
- Minmaler Wartungsaufwand
- Fehlerbehebung

Die Anforderungen werden folgendermaßen priorisiert:

Tabelle 5.3: Scoring Tabelle für die Bewertung der Big Data Architekturen

Priorität	NFA	Erfüllungsgrad			
		100% - 85 %	84% - 50 %	49% - 25 %	< 24 %
		Hoch	Mittel	Gering	Nicht
1	Belastbarkeit und Fehlertoleranz	24	16	8	0
2	Skalierbarkeit	21	14	7	0
3	Geringe Latenzzeiten	18	12	6	0
4	Erweiterbarkeit	15	10	5	0
5	Allgemeingültigkeit	12	8	4	0
6	Ad-Hoc-Abfragen	9	6	3	0
7	Minimaler Wartungsaufwand	6	4	2	0
8	Fehlerbehebung	3	2	1	0
	Score max.				108

Die Skalierbarkeit steht über der geringen Latenzzeit, da eine gute Skalierbarkeit eine geringe Latenzzeit impliziert. Wenn ein System gut skalierbar ist, kann die Latenzzeit z.B. durch hinzugefügte Rechner gesenkt werden.
Die Erweiterbarkeit wird höher priorisiert als die Allgemeingültigkeit, da dem System durch eine gute Erweiterbarkeit neue Funktionalität hinzugefügt werden kann. Dies wird in einem Big Data System benötigt, um zusätzliche Berechnungen für neue Analysen hinzuzufügen. Durch eine gute Erweiterbarkeit kann die vorhandene Allgemeingültigkeit eines Systems gesteigert werden.
Durch einen minimalen Wartungsaufwand wird die Fehlerbehebung erleichtert.

5.3 Bewertung der Lambda Architektur

Im Folgenden wird die Lambda Architektur anhand der ausgewählten Methoden bewertet. Zunächst erfolgt die Bewertung anhand der SAAM. Danach wird die Bewertung anhand der Anforderungen an das Big Data System vorgenommen.

5.3.1 Bewertung anhand der Software Architecture Analysis Method

5.3.1.1 Klassifizierung der Szenarien

Szenario 1: Hinzufügen von Daten aus den Datenquellen in das Big Data System.

Da die Architektur kein Modul zur Verfügung stellt, das sich um das Hinzufügen der Daten kümmert, müssen sie jeweils in den Batch und Speed Layer eingefügt werden. Möglich wäre auch ein außenstehendes Modul, das Daten an beide Layer sendet. Eine Änderung der Architektur ist jedoch sinnvoller.

Aufgrund des fehlenden Moduls, das eingehende Daten entgegennimmt und an Batch und Speed Layer weiterleitet, wird dieses Szenario als indirekt kategorisiert.

Szenario 2: Korrigieren von fehlerhaften Daten und die entsprechende Korrektur der bisher erstellten Analysen.

In der Lambda Architektur müssen fehlerhafte Daten nur im Batch Layer korrigiert werden, da die Daten im Speed Layer nicht langfristig gespeichert und die Ergebnisse der Berechnung verworfen werden, wenn der Batch Layer seine Berechnungen fertig gestellt hat.

Somit bedarf dieses Szenario keiner Änderung an der Architektur und kann folglich als direktes Szenario kategorisiert werden.

Szenario 3: Löschen von Daten innerhalb des Big Data Systems, z.B. aus Datenschutzgründen.

Ähnlich wie in Szenario 2 müssen die Daten nur im Batch Layer gelöscht werden, da sie im Speed Layer nicht langfristig gespeichert werden.
Ob der Batch Layer ansprechbar für die Löschung von Daten ist, ist nicht geklärt. Die Vermutung liegt nahe, dass dies nicht der Fall ist, da der Batch Layer lediglich Daten aus den Datenquellen entgegennimmt und Ergebnisse der Berechnungen an den Serving Layer liefert. Demnach ist eine Änderung der Architektur notwendig und das Szenario

als indirektes Szenario zu betrachten.

Szenario 4: Abfragen von Ergebnissen aus den Auswertungen.

Die Abfrage der Ergebnisse aus den Auswertungen geschieht direkt im Serving Layer. Die Architektur muss nicht geändert werden. Deshalb handelt es sich bei diesem Szenario um ein direktes Szenario.

Szenario 5: Abfragen von einzelnen Datensätzen, um beispielsweise Ausreißer zu identifizieren.

Einzelne Datensätze können im Batch Layer angefragt werden. Da dieser lediglich Daten in Empfang nehmen, speichern und Berechnungen darauf ausführen soll und die Ergebnisse der Berechnungen an den Serving Layer leitet, ist der Batch Layer zumindest nach der vorliegenden Architektur nicht für den Anwender ansprechbar. Eine Abfrage von einzelnen Datensätzen ist nicht möglich und es bedarf einer Architekturänderung. Es liegt ein indirektes Szenario vor.

Szenario 6: Hinzufügen einer neuen Methode zur Auswertung der Daten.

Neue Methoden zur Auswertung müssen sowohl im Batch als auch im Speed Layer eingefügt und für jede Schicht einzeln implementiert werden. Dies stellt einen erhöhten Aufwand dar, der nur durch eine Änderung der Architektur umgangen werden kann. Demnach wird das Szenario als indirektes Szenario gewertet.

Szenario 7: Ergänzen der Datensätze um neue Attribute.

Vorhandene Datensätze lassen sich in der Lambda Architektur erweitern. Hierbei müssen die Änderungen für den Batch und den Speed Layer implementiert werden. Um einen Mehraufwand zu umgehen, sollte eine Änderung der Architektur erfolgen. Deshalb liegt ein indirektes Szenario vor.

Für eine bessere Übersichtlichkeit werden nachfolgend alle Szenarien nach Klassifizie-

rung sortiert.

Direkte Szenarien

- Szenario 2: Korrigieren von fehlerhaften Daten und die entsprechende Korrektur der bisher erstellten Analysen.
- Szenario 4: Abfragen von Ergebnissen aus den Auswertungen.

Indirekte Szenarien

- Szenario 1: Hinzufügen von Daten aus den Datenquellen in das Big Data System.
- Szenario 3: Löschen von Daten innerhalb des Big Data Systems, z.B. aus Datenschutzgründen.
- Szenario 5: Abfragen von einzelnen Datensätzen, um beispielsweise Ausreißer zu identifizieren.
- Szenario 6: Hinzufügen einer neuen Methode zur Auswertung der Daten.
- Szenario 7: Ergänzen der Datensätze um neue Attribute.

5.3.1.2 Bewertung der einzelnen Szenarien

Direkte Szenarien

Szenario 2: Korrigieren von fehlerhaften Daten und die entsprechende Korrektur der bisher erstellten Analysen.

Die Korrektur wird von der Datenquelle oder einem Benutzer an den Batch Layer gesendet und dort verarbeitet. Sobald die Korrektur vorgenommen wurde, wird sie in der neuen Berechnung der Views übernommen. Die durch die falschen Daten beeinflussten Views werden dadurch überschrieben.

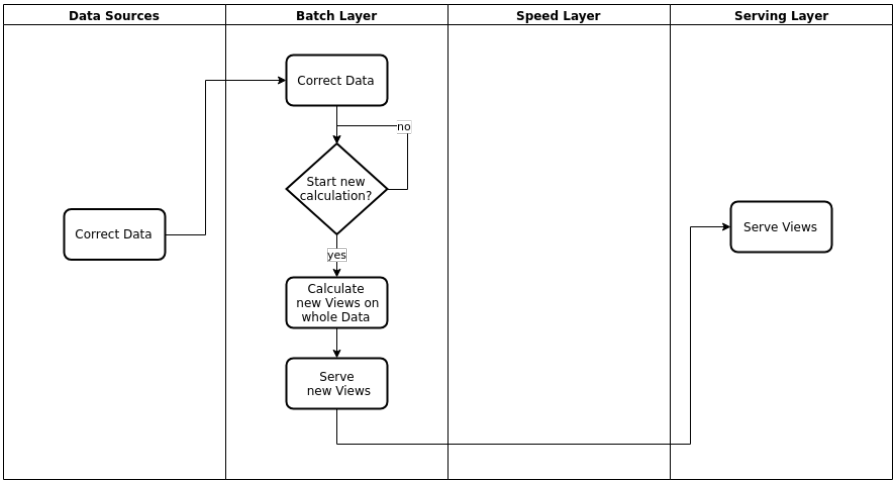

Abbildung 5.2: Ablaufdiagramm der Lambda Architektur für das Szenario 2

Szenario 4: Abfragen von Ergebnissen aus den Auswertungen.

Der Benutzer kann die Ergebnisse der Berechnungen direkt beim Serving Layer abfragen. Die vom Batch und Speed Layer berechneten Views werden gespeichert. Sind aktuellere Ergebnisse vorhanden, werden die alten Views überschrieben.

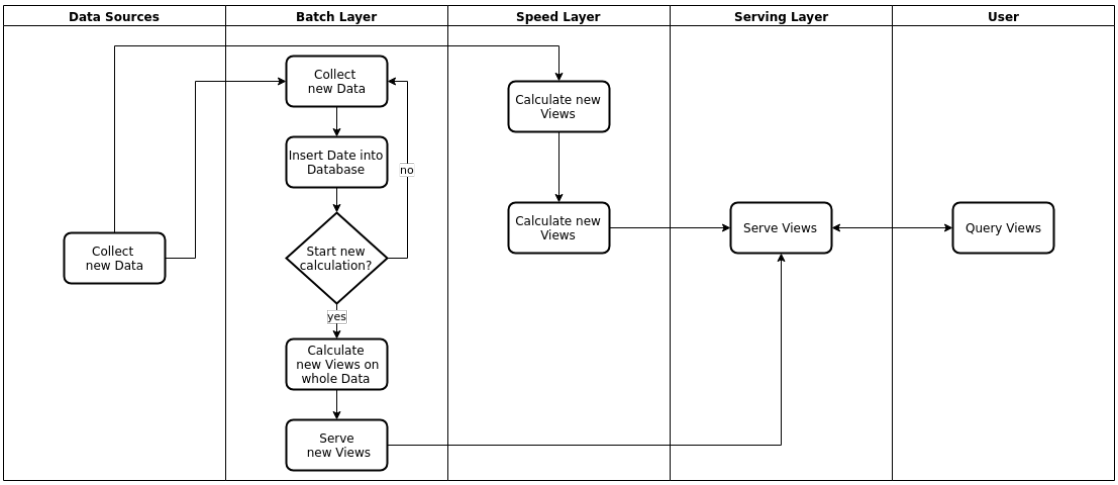

Abbildung 5.3: Ablaufdiagramm der Lambda Architektur für das Szenario 4

Indirekte Szenarien

Szenario 1: Hinzufügen von Daten aus den Datenquellen in das Big Data System.

Die Daten aus den Datenquellen werden in den Batch und in den Speed Layer eingefügt. Im Batch Layer werden sie gesammelt, bis eine Berechnung über alle Daten angestoßen wird. Die Berechnungen werden stets auf allen Daten ausgeführt. Die Ergebnisse der Berechnungen werden im Serving Layer für die Nutzer zur Verfügung gestellt.
Im Speed Layer werden die Daten nach dem Einfügen ausgewertet und die aktuellen Auswertungen werden an den Serving Layer zur Abfrage durch die Nutzer hinterlegt. Sind die Berechnungen des Batch Layers abgeschlossen, werden die Ergebnisse des Speed Layers im Serving Layer von diesen überschrieben, da diese Ergebnisse präziser sind.

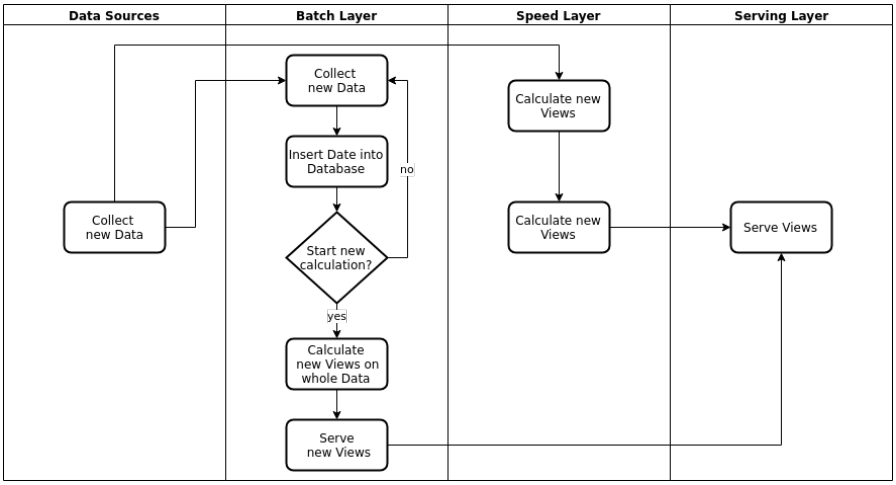

Abbildung 5.4: Ablaufdiagramm der Lambda Architektur für das Szenario 1

Als Änderung wird ein weiterer Layer empfohlen, der die Daten aus den Datenquellen entgegennimmt und an den Batch Layer und den Speed Layer weiterleitet. Hierdurch müssen die Datenquellen neue Daten nur noch an eine Stelle der Architektur senden und diese übernimmt die Verteilung an die folgenden Layer.
Der Aufwand zur Änderung ist gering, da lediglich ein Layer hinzugefügt werden muss, der die Daten entgegennimmt und weiter verteilt.

Szenario 3: Löschen von Daten innerhalb des Big Data Systems, z.B. aus Datenschutzgründen.

Das Löschen der Daten wird durch Benutzer veranlasst und an den Batch Layer weitergegeben, in welchem die Daten direkt gelöscht werden.

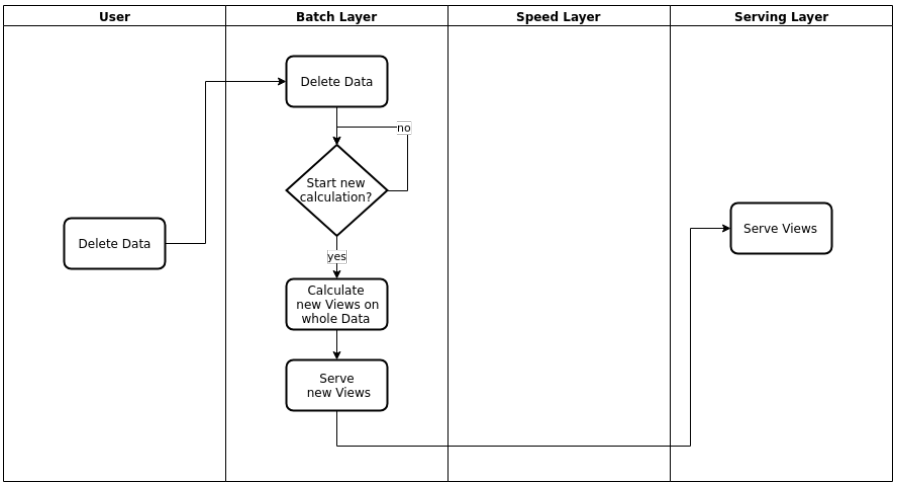

Abbildung 5.5: Ablaufdiagramm der Lambda Architektur für das Szenario 3

Der Aufwand zur Änderung ist als gering zu betrachten, da nur unklar ist, ob der Batch Layer direkt ansprechbar ist. Sofern dies nicht der Fall ist, müssen die Schnittstellen bereitgestellt werden.

Szenario 5: Abfragen von einzelnen Datensätzen, um beispielsweise Ausreißer zu identifizieren.

Da alle Daten im Batch Layer gehalten werden, muss dieser angesprochen werden, um einzelne Datensätze abzufragen.

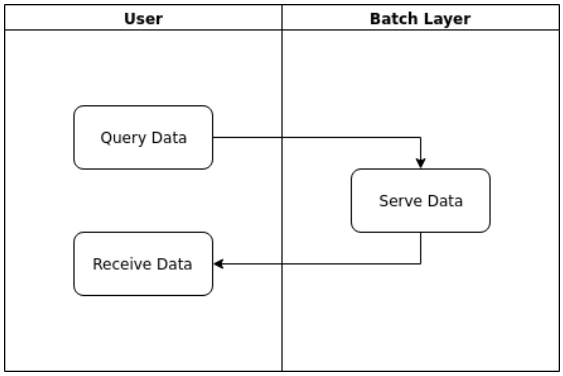

Abbildung 5.6: Ablaufdiagramm der Lambda Architektur für das Szenario 5

Es ist unklar, ob der Batch Layer die Schnittstellen zur Abfrage der Daten anbietet. Eine Änderung ist mit geringem Aufwand umsetzbar.

Szenario 6: Hinzufügen einer neuen Methode zur Auswertung der Daten.

Neue Auswertungen müssen für Batch und Speed Layer implementiert und in diese eingefügt werden. Das stellt einen erhöhten Mehraufwand dar, da ähnliche Funktionalität doppelt implementiert werden muss.

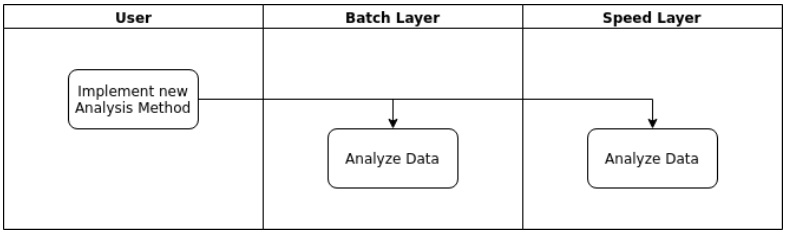

Abbildung 5.7: Ablaufdiagramm der Lambda Architektur für das Szenario 6

Möchte man diesen Mehraufwand vermeiden, muss ein Modul geschaffen werden, das die Methoden zur Analyse für beide Layer erstellt und in diese einfügt. Das ist nur durch einen hohen Aufwand umsetzbar.

Szenario 7: Ergänzen der Datensätze um neue Attribute.

Wenn die behandelten Datensätze um neue Attribute erweitert werden sollen, muss zum einen der Batch Layer angepasst werden, da hier die Daten gehalten werden. Zum anderen müssen alle betroffenen Methoden zur Analyse im Batch und im Speed Layer angepasst werden. Wie in Szenario 6 stellt dies einen Mehraufwand dar, da man jede Methode zur Analyse in beiden Layern anpassen muss.

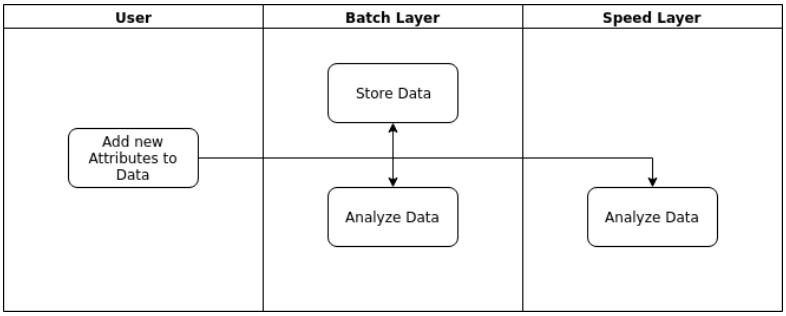

Abbildung 5.8: Ablaufdiagramm der Lambda Architektur für das Szenario 7

Um diesen Mehraufwand zu vermeiden, könnte das gleiche Modul verwendet werden, das schon für die Umsetzung von Szenario 6 implementiert wurde. Deshalb ist der Aufwand zur Änderung als hoch zu betrachten.

5.3.1.3 Untersuchung der Szenariointeraktionen

Bei allen Szenarien besteht nur eine Szenariointeraktion. Diese findet zwischen Szenario 6 und Szenario 7 statt und trifft nur zu, wenn man die Architektur ändern möchte, um einen Mehraufwand zu vermeiden.

5.3.1.4 Erstellen der Gesamtbewertung

Bei der Punktevergabe für die Lambda Architektur ist zu beachten, dass die Szenarien 6 und 7 zwar mit hohem Änderungsaufwand gewertet wurden, jedoch nur die Punkte für einen mittleren Aufwand zugewiesen bekommen. Das liegt daran, dass die Änderungen nicht zwingend erfolgen müssen, sondern lediglich dazu dienen, einen Mehraufwand bei einer Ergänzung gering zu halten.

Die Lambda Architektur hat in der Bewertung anhand der SAAM demnach 9 von insgesamt 35 Punkten, wobei zu beachten ist, dass eine höhere Punktzahl eine schlechtere Bewertung bedeutet.

Tabelle 5.4: Gesamtbewertung der Lambda Architektur

Szenario	**Punkte**
Szenario 1	1
Szenario 2	0
Szenario 3	1
Szenario 4	0
Szenario 5	1
Szenario 6	3
Szenario 7	3
Summe	**9**

5.3.2 Bewertung anhand nichtfunktionaler Anforderungen

5.3.2.1 Belastbarkeit und Fehlertoleranz

Die Belastbarkeit der Lambda Architektur ist hoch, da sie für den Betrieb auf mehreren Rechnern geplant wurde. Dadurch ist sie hochverfügbar und dementsprechend belastbar. Überdies ist die Architektur fehlertolerant, da die Daten im Batch Layer als Fakten gespeichert werden und die Auswertungen lediglich Funktionen auf der Menge aller Fakten sind. Stammen falsche Berechnungen aus dem Speed Layer, werden diese durch die korrekten Ergebnisse des Batch Layers ersetzt.

5.3.2.2 Skalierbarkeit

Die Skalierbarkeit der Lambda Architektur hängt maßgeblich von den benutzten Technologien ab. Es ist zwar vorgesehen, dass sehr gut skalierbare Systeme verwendet werden, allerdings wird dies nicht maßgeblich durch die Architektur beeinflusst. Deshalb wird die Skalierbarkeit der Lambda Architektur auf mittel klassifiziert.

5.3.2.3 Geringe Latenzzeit

Die Ergebnisse der Auswertungen werden auf zwei unterschiedliche Arten berechnet. Der Batch Layer berechnet die Auswertungen aus der Menge aller Daten. Das ist sehr genau, hat jedoch eine hohe Latenzzeit zur Folge. Der Speed Layer berechnet die Auswertungen nicht auf der Menge aller Daten sondern inkrementell. Das ist weniger genau, aber die Latenzzeit ist deutlich geringer. Somit kann die NFA der geringen Latenzzeit als hoch klassifiziert werden.

5.3.2.4 Erweiterbarkeit

Die Erweiterbarkeit der Lambda Architektur wird als mittel klassifiziert, da Auswertungen sowohl für den Batch Layer als auch für den Speed Layer implementiert werden müssen. Dies ist ein Mehraufwand, da eine ähnliche Funktionalität doppelt implementiert werden muss.

5.3.2.5 Allgemeingültigkeit

Da die Lambda Architektur an keinen Zweck gebunden ist, kann die Allgemeingültigkeit als hoch klassifiziert werden. Die Lambda Architektur lässt sich auch ohne Probleme, allerdings mit dem vorher erwähnten erhöhten Aufwand, ändern bzw. erweitern und hat folglich eine sehr gute Allgemeingültigkeit.

5.3.2.6 Ad-Hoc-Abfragen

Ad-Hoc-Abfragen in der Lambda Architektur werden als mittel klassifiziert. Die vorhandenen Auswertungen können sofort abgefragt werden. Die Ergebnisse neu hinzugefügter Auswertungen können dagegen erst nach einer Berechnung zur Verfügung gestellt werden. Da eine Berechnung des Batch Layers mehrere Stunden oder länger dauern kann und der Speed Layer eine hohe Menge an Daten benötigt, kann die Architektur die Auswertungen erst nach einiger Zeit bereitstellen.

5.3.2.7 Wartungsaufwand

Da der Batch Layer und der Speed Layer ähnliche Funktionalität besitzen und beide zeitgleich gewartet werden müssen, erfolgt auch hier doppelte Arbeit. Dementsprechend wird diese NFA ebenfalls als mittel klassifiziert.

5.3.2.8 Fehlerbehebung

Wenn Fehler in den Auswertungen auftauchen, müssen sie im Batch und im Speed Layer korrigiert werden. Da eine doppelte Arbeit erfolgen muss, wird diese NFA als mittel klassifiziert.

5.3.2.9 Gesamtbewertung

In der Bewertung der NFA hat die Lambda Architektur 90 von 108 Punkten erzielt. Es liegt bei höherer Punktzahl eine bessere Bewertung vor.

Tabelle 5.5: Gesamtbewertung der Lambda Architektur für die NFA

Priorität	NFA	Ergebnis
1	Belastbarkeit und Fehlertoleranz	24
2	Skalierbarkeit	14
3	Geringe Latenzzeiten	18
4	Erweiterbarkeit	10
5	Allgemeingültigkeit	12
6	Ad-Hoc-Abfragen	6
7	Minimaler Wartungsaufwand	4
8	Fehlerbehebung	2
	Score	90 / 108

5.4 Bewertung der Learning Analytics Architecture

Im Folgenden werden die ausgewählten Methoden zur Bewertung der Learning Analytics Architecture angewendet.

5.4.1 Bewertung anhand der Software Architecture Analysis Method

5.4.1.1 Klassifizierung der Szenarien

Szenario 1: Hinzufügen von Daten aus den Datenquellen in das Big Data System.

Das Datenerhebungsmodul nimmt die Daten aus den unterschiedlichen Datenquellen entgegen und leitet sie zur Speicherung weiter. Da ein Modul zum Empfangen von Daten vorhanden ist, muss die Architektur nicht geändert werden. Es handelt sich somit um ein direktes Szenario.

Szenario 2: Korrigieren von fehlerhaften Daten und die entsprechende Korrektur der bisher erstellten Analysen.

Die Korrektur von fehlerhaften Daten kann im Datenspeicher- und Managementsystem erfolgen. Die Korrektur der erstellten Analysen erfolgt im Datenanalysesystem und wird durch eine erneute Berechnung vorgenommen. Dadurch ist keine Änderung an der Architektur notwendig. Es liegt ein direktes Szenario vor.

Szenario 3: Löschen von Daten innerhalb des Big Data Systems, z.B. aus Datenschutzgründen.

Die Daten können über das Datenspeicher- und Managementsystem gelöscht werden. Somit ist keine Änderung notwendig. Es handelt sich um ein direktes Szenario.

Szenario 4: Abfragen von Ergebnissen aus den Auswertungen.

Die Ergebnisse der Auswertungen können durch das Datenvisualisierungssystem visuell dargestellt werden. Eine Abfrage der Ergebnisse aus dem Datenanalysesystem sollte auch möglich sein, jedoch ist hier keine Information vorhanden, ob es direkt angesprochen werden kann. Daher wird dieses Szenario als indirektes Szenario gewertet.

Szenario 5: Abfragen von einzelnen Datensätzen, um beispielsweise Ausreißer zu identifizieren.

Die Abfrage von einzelnen Datensätzen ist durch das Datenspeicher- und Managementsystem möglich. Es ist keine Änderung der Architektur notwendig. Auch dieses Szenario ist ein direktes Szenario.

Szenario 6: Hinzufügen einer neuen Methode zur Auswertung der Daten.

In der Architektur wird lediglich beschrieben, dass das Datenanalysesystem Algorithmen zur Analyse der Daten enthält. Ob sich diese anpassen oder ergänzen lassen, ist unbekannt. Infolgedessen ist für dieses Szenario eine eventuelle Änderung der Architektur notwendig. Demnach ist dieses Szenario ein indirektes Szenario.

Szenario 7: Ergänzen der Datensätze um neue Attribute.

Die Datensätze können im Datenspeicher- und Managementsystem geändert werden. Für dieses Szenario ist also keine Änderung an der Architektur notwendig. Es handelt sich um ein direktes Szenario.

Direkte Szenarien

- Szenario 1: Hinzufügen von Daten aus den Datenquellen in das Big Data System.
- Szenario 2: Korrigieren von fehlerhaften Daten und die entsprechende Korrektur der bisher erstellten Analysen.
- Szenario 3: Löschen von Daten innerhalb des Big Data Systems, z.B. aus Datenschutzgründen.
- Szenario 5: Abfragen von einzelnen Datensätzen, um beispielsweise Ausreißer zu identifizieren
- Szenario 7: Ergänzen der Datensätze um neue Attribute.

Indirekte Szenarien

- Szenario 4: Abfragen von Ergebnissen aus den Auswertungen.
- Szenario 6: Hinzufügen einer neuen Methode zur Auswertung der Daten.

5.4.1.2 Bewertung der einzelnen Szenarien

Direkte Szenarien

Szenario 1: Hinzufügen von Daten aus den Datenquellen in das Big Data System.

Die Daten werden vom Datenerhebungsdienst, der eine Sammlung aller Datenquellen darstellt, in das Datenspeicher- und Managementsystem weitergeleitet. Von hier aus werden die eingehenden Daten bereinigt und für die Analyse vorverarbeitet. Im Datenanalysesystem werden die Daten mittels intelligenter Verarbeitungsalgorithmen ausgewertet und die Ergebnisse werden zur visuellen Darstellung im Datenvisualisierungssystem weitergegeben. Die Nutzer des Systems können Erkenntnisse aus den Daten sammeln.

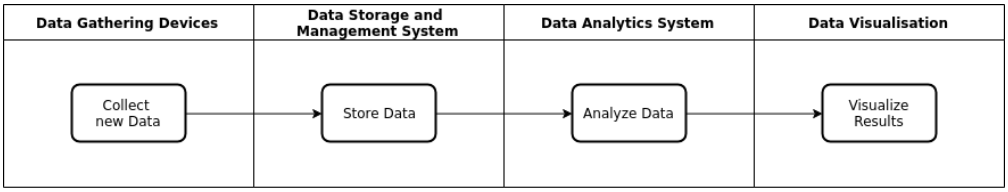

Abbildung 5.9: Ablaufdiagramm der Learning Analytics Architecture für das Szenario 1

Der Ablauf ist nur geschätzt, da die Architektur nicht genau beschrieben wurde. Dadurch, dass eine detaillierte Beschreibung fehlt, wird dieses Szenario wie ein indirektes Szenario mit geringem Änderungsaufwand gewertet.

Szenario 2: Korrigieren von fehlerhaften Daten und die entsprechende Korrektur der bisher erstellten Analysen.

Die Daten werden von den Benutzern über die Datenerhebungsdienste oder direkt über das Datenspeicher- und Managementsystem korrigiert. Die korrigierten Daten werden gespeichert und falls nötig neue Analysen durchgeführt. Die Ergebnisse der Analysen werden im Datenvisualisierungssystem dargestellt.

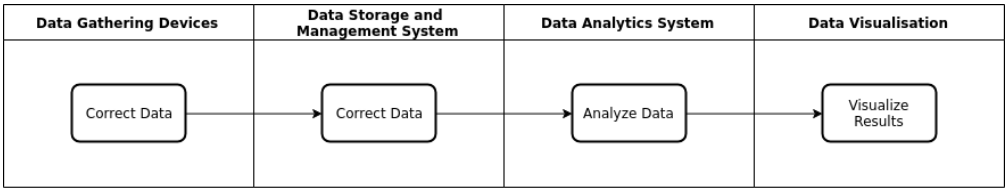

Abbildung 5.10: Ablaufdiagramm der Learning Analytics Architecture für das Szenario 2

Der Ablauf ist nur geschätzt, da die Architektur nicht genau beschrieben wurde. Dadurch, dass eine detaillierte Beschreibung fehlt, wird dieses Szenario wie ein indirektes Szenario mit geringem Änderungsaufwand gewertet.

Szenario 3: Löschen von Daten innerhalb des Big Data Systems, z.B. aus Datenschutzgründen.

Das Löschen der Daten wird durch den Benutzer über die Datenerhebungsdienste oder direkt über das Datenspeicher- und Managementsystem beantragt. Die Daten werden gelöscht und falls nötig neue Analysen durchgeführt. Die Ergebnisse der Analysen werden im Datenvisualisierungssystem dargestellt.

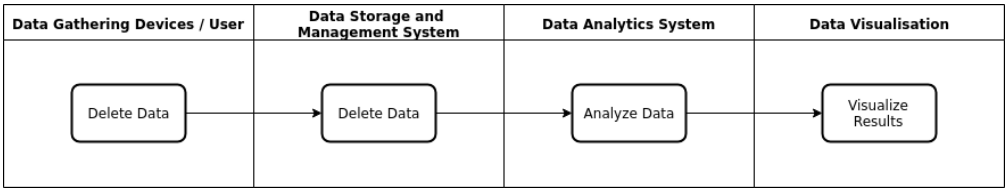

Abbildung 5.11: Ablaufdiagramm der Learning Analytics Architecture für das Szenario 3

Der Ablauf ist nur geschätzt, da die Architektur nicht genau beschrieben wurde. Dadurch, dass eine detaillierte Beschreibung fehlt, wird dieses Szenario wie ein indirektes Szenario mit geringem Änderungsaufwand gewertet.

Szenario 5: Abfragen von einzelnen Datensätzen, um beispielsweise Ausreißer zu identifizieren.

Der Benutzer kann die Daten direkt über das Datenspeicher- und Managementsystem

abfragen und empfangen.

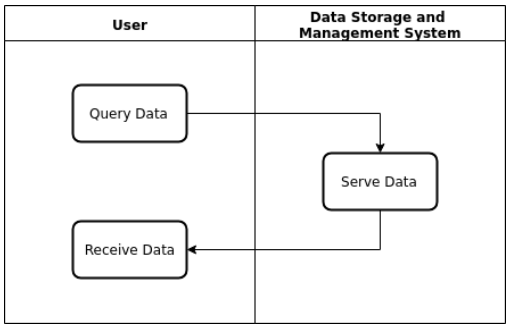

Abbildung 5.12: Ablaufdiagramm der Learning Analytics Architecture für das Szenario 5

Szenario 7: Ergänzen der Datensätze um neue Attribute.

Die neuen Attribute der Daten werden durch den Benutzer in das Datenspeicher- und Managementsystem eingefügt. Neu ankommende Daten werden künftig mit den neuen Attributen gespeichert.

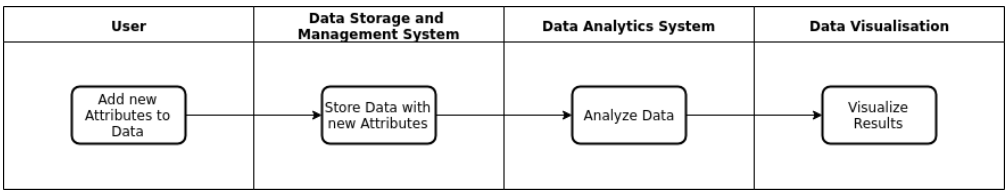

Abbildung 5.13: Ablaufdiagramm der Learning Analytics Architecture für das Szenario 7

Der Ablauf ist nur geschätzt, da die Architektur nicht genau beschrieben wurde. Dadurch, dass eine detaillierte Beschreibung fehlt, wird dieses Szenario wie ein indirektes Szenario mit geringem Änderungsaufwand gewertet.

Indirekte Szenarien

Szenario 4: Abfragen von Ergebnissen aus den Auswertungen.

Die Ergebnisse der Auswertungen werden durch das Datenanalysesystem bereitgestellt und dem Benutzer durch das Datenvisualisierungssystem präsentiert.

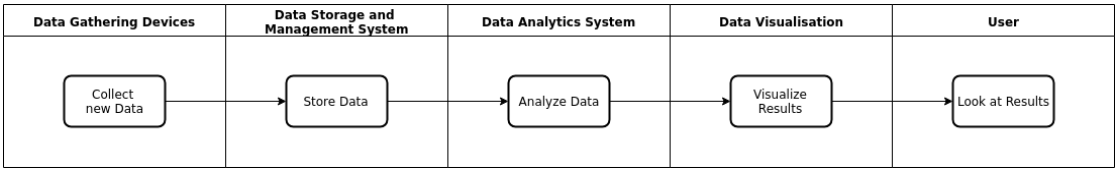

Abbildung 5.14: Ablaufdiagramm der Learning Analytics Architecture für das Szenario 4

Sollen die Ergebnisse aber direkt abgefragt und nicht visualisiert werden, müsste das Datenanalysesystem direkt angefragt werden. Da unklar ist, ob diese Funktion zur Verfügung steht, wurde dieses Szenario als indirektes Szenario klassifiziert. Der Aufwand zur Änderung ist aber als gering anzusehen.

Szenario 6: Hinzufügen einer neuen Methode zur Auswertung der Daten.

Neue Methoden zur Auswertung müssen in das Datenanalysesystem eingefügt werden.

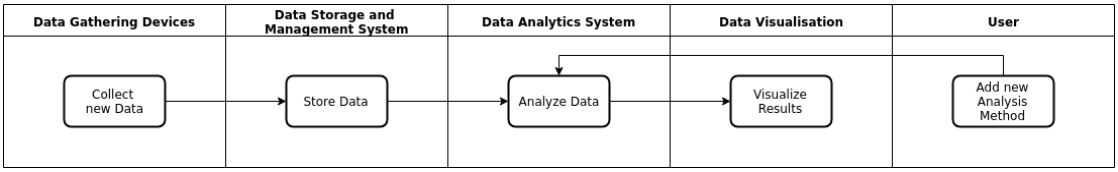

Abbildung 5.15: Ablaufdiagramm der Learning Analytics Architecture für das Szenario 6

Da unklar ist, ob eine Schnittstelle zur Verfügung steht, mit der man diese Methoden nachträglich hinzufügen kann, ist dieses Szenario ein indirektes Szenario. Der Aufwand für eine eventuelle Änderung der Architektur ist als gering zu klassifizieren, da hier nur eine Schnittstelle hinzugefügt werden muss.

5.4.1.3 Untersuchung der Szenariointeraktionen

Es besteht keine Szenariointeraktion.

5.4.1.4 Erstellen der Gesamtbewertung

Bei der Punktevergabe für die Learning Analytics Architecture ist zu beachten, dass die Szenarien 1, 2, 3 und 7 zwar als direkte Szenarien klassifiziert wurden, jedoch trotzdem wie indirekte Szenarien mit geringem Änderungsaufwand gewertet werden, da die Architekturbeschreibung nicht detailliert genug ist.

Die Learning Analytics Architecture konnte hier 6 von 35 Punkten erlangen und liegt daher in dieser Kategorie vor der Lambda Architektur, die 9 Punkte erreicht hat. Zu bemerken ist, dass eine höhere Punktzahl eine schlechtere Bewertung bedeutet.

Tabelle 5.6: Gesamtbewertung der Learning Analytics Architecture

Szenario	**Punkte**
Szenario 1	1
Szenario 2	1
Szenario 3	1
Szenario 4	1
Szenario 5	0
Szenario 6	1
Szenario 7	1
Summe	**6**

5.4.2 Bewertung anhand nichtfunktionaler Anforderungen

5.4.2.1 Belastbarkeit und Fehlertoleranz

Im Vergleich zu den anderen Architekturen findet die Belastbarkeit und auch die Fehlertoleranz wenig Beachtung. Daher wird diese NFA als mittel klassifiziert.

5.4.2.2 Skalierbarkeit

Da die Skalierbarkeit von den genutzten Technologien abhängt, wird diese NFA als mittel kategorisiert.

5.4.2.3 Geringe Latenzzeit

In der Architekturbeschreibung steht lediglich, dass im Datenanalysesystem intelligente Verarbeitungsalgorithmen verwendet werden. Inwieweit diese eine geringe Latenzzeit versprechen, ist also unbekannt. Folglich wird die NFA mit gering klassifiziert.

5.4.2.4 Erweiterbarkeit

Hier müssen zwar nicht die Algorithmen zur Verarbeitung doppelt implementiert werden, jedoch betreffen einige Änderungen alle Module. Somit wird diese Anforderung als mittel eingestuft.

5.4.2.5 Allgemeingültigkeit

Da die Ergebnisse lediglich visuell darstellbar sind, ist diese Architektur nicht für alle Anwendungsfälle einsetzbar. Aus diesem Grund wird die Allgemeingültigkeit ebenfalls als mittel eingeordnet.

5.4.2.6 Ad-Hoc-Abfragen

Da die Ergebnisse nur visuell dargestellt werden, können keine speziellen Daten abgefragt werden. Dies scheint für den geplanten Anwendungsfall nicht notwendig zu sein, jedoch sind keine Ad-Hoc-Abfragen möglich. Aufgrund dessen erfolgt eine Klassifizierung als gering.

5.4.2.7 Geringer Wartungsaufwand

Der geringe Wartungsaufwand wird als mittel klassifiziert, da eine Anbindung zu jeder Datenquelle Teil des Systems ist. Diese erzeugen einen hohen Wartungsaufwand.

5.4.2.8 Fehlerbehebung

Die Fehlerbehebung wird als mittel klassifiziert, da auftretende Fehler, vor allem in den Analysen, sich nur schwer nachvollziehen und korrigieren lassen.

5.4.2.9 Gesamtbewertung

In der Bewertung der NFA hat die Learning Analytics Architecture 63 von insgesamt 108 Punkten erreicht. Damit hat die Architektur weniger Punkte erreicht als die Lambda Architektur.

Tabelle 5.7: Gesamtbewertung der Learning Analytics Architecture für die NFA

Priorität	NFA	Ergebnis
1	Belastbarkeit und Fehlertoleranz	16
2	Skalierbarkeit	14
3	Geringe Latenzzeiten	6
4	Erweiterbarkeit	10
5	Allgemeingültigkeit	8
6	Ad-Hoc-Abfragen	3
7	Minimaler Wartungsaufwand	4
8	Fehlerbehebung	2
	Score	63 / 108

5.5 Bewertung der Reference Architecture for Big Data Systems in the National Security Domain

5.5.1 Bewertung anhand der Software Architecture Analysis Method

5.5.1.1 Klassifizierung der Szenarien

Szenario 1: Hinzufügen von Daten aus den Datenquellen in das Big Data System.

Das Collection Modul ist für das Entgegennehmen der Daten zuständig und leitet es an die jeweiligen Folgemodule weiter. Deswegen ist dieses Szenario ein direktes Szenario.

Szenario 2: Korrigieren von fehlerhaften Daten und die entsprechende Korrektur der bisher erstellten Analysen.

Das Korrigieren der Daten kann entweder über das Collection Modul durch die Datenquellen oder über das Access Modul durch Benutzer erfolgen. Da jedoch nicht beschrieben wurde, ob eine derartige Funktionalität in den jeweiligen Modulen besteht, wird dieses Szenario als ein indirektes Szenario klassifiziert.

Szenario 3: Löschen von Daten innerhalb des Big Data Systems, z.B. aus Datenschutzgründen.

Das Löschen kann, genau wie die Korrektur der Daten, entweder über das Collection Modul oder das Access Modul erfolgen. Da unklar ist, ob die Funktionalität existiert, ist ein indirektes Szenario gegeben.

Szenario 4: Abfragen von Ergebnissen aus den Auswertungen.

Die Ergebnisse der Auswertungen können entweder über das Visualisation Modul für die Benutzer visualisiert oder über das Access Modul mittels API abgefragt werden. Folglich ist diese Szenario ein direktes Szenario.

Szenario 5: Abfragen von einzelnen Datensätzen, um beispielsweise Ausreißer zu identifizieren.

Um einzelne Datensätze abzufragen, kann das Access Modul verwendet werden. Die Abfrage geschieht mittels API. Jedoch ist nicht beschrieben, ob diese Funktionalität vorhanden ist. Somit ist ein geringer Änderungsaufwand notwendig und es handelt sich um ein indirektes Szenario.

Szenario 6: Hinzufügen einer neuen Methode zur Auswertung der Daten.

Durch das Application Orchestration Modul können neue Anwendungen durch den Benutzer integriert werden. Eine Anwendung ist die komplette Datenverarbeitung durch das System. So können bequem neue Methoden hinzugefügt werden. Aus diesem Grund besteht kein Änderungsaufwand und dieses Szenario ist ein direktes Szenario.

Szenario 7: Ergänzen der Datensätze um neue Attribute.

Die Daten und die Methoden zur Auswertung können durch das Application Orchestration Modul geändert oder neu erstellt werden. Infolgedessen ist keine Änderung nötig, sodass es sich um ein direktes Szenario handelt.

Zur besseren Übersicht hier noch einmal alle Szenarien nach Klassifizierung sortiert.

Direkte Szenarien

- Szenario 1: Hinzufügen von Daten aus den Datenquellen in das Big Data System.
- Szenario 5: Abfragen von einzelnen Datensätzen, um beispielsweise Ausreißer zu identifizieren.
- Szenario 6: Hinzufügen einer neuen Methode zur Auswertung der Daten.
- Szenario 7: Ergänzen der Datensätze um neue Attribute.

Indirekte Szenarien

- Szenario 2: Korrigieren von fehlerhaften Daten und die entsprechende Korrektur der bisher erstellten Analysen.

- Szenario 3: Löschen von Daten innerhalb des Big Data Systems, z.B. aus Datenschutzgründen.
- Szenario 4: Abfragen von Ergebnissen aus den Auswertungen.

5.5.1.2 Bewertung der einzelnen Szenarien

Direkte Szenarien

Szenario 1: Hinzufügen von Daten aus den Datenquellen in das Big Data System.

Die Daten werden mit Hilfe des Collection Moduls aus den Datenquellen in das System eingefügt. Von dort aus werden sie mit dem Preparation Modul für die Analyse und zum Speichern vorverarbeitet. Danach werden sie im Data Storage Modul gespeichert. Das Analytics Modul verarbeitet die Daten mit Hilfe des Processing Moduls, welches für eine verteilte Analyse der Daten verantwortlich ist. Die Ergebnisse können dann entweder über das Visualisation Modul angezeigt oder durch das Access Modul abgefragt werden.

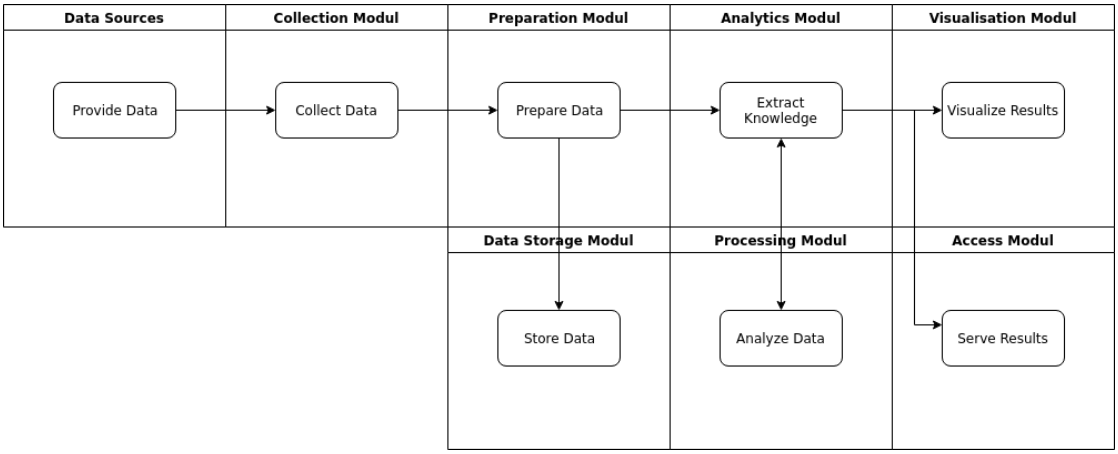

Abbildung 5.16: Ablaufdiagramm der Learning Analytics Architecture für das Szenario 1

Szenario 4: Abfragen von Ergebnissen aus den Auswertungen.

Nach Einfügen und Analysieren der Daten wie unter Szenario 1 beschrieben, können

die Ergebnisse der Auswertungen mit Hilfe des Visualization Moduls visuell dargestellt oder durch das Access Modul abgefragt werden.

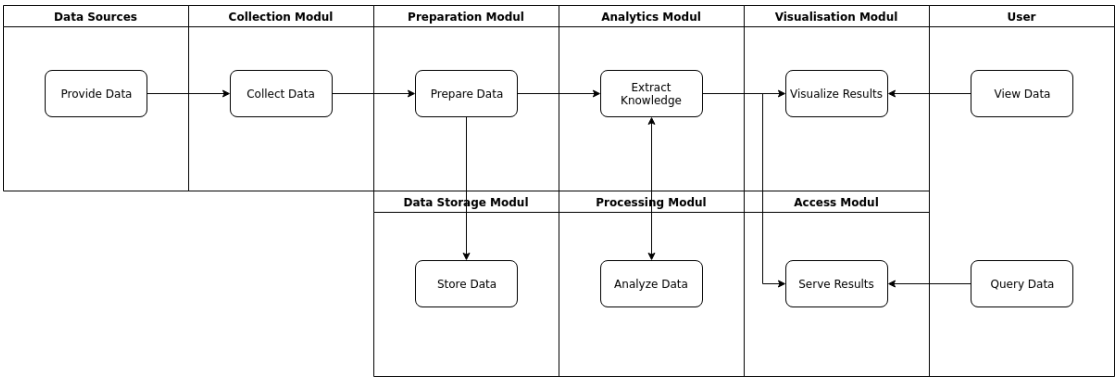

Abbildung 5.17: Ablaufdiagramm der Reference Architecture for Big Data Systems in the National Security Domain für das Szenario 4

Szenario 6: Hinzufügen einer neuen Methode zur Auswertung der Daten.

Mittels des Application Orchestration Moduls kann ein Benutzer neue Anwendungen anlegen. Eine Anwendung ist ein kompletter Prozess von der Datenbeschaffung bis hin zur Auswertung. Diese Anwendungen werden anhand einer GUI oder einer DSL erstellt.

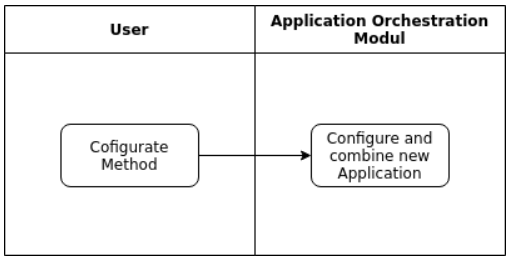

Abbildung 5.18: Ablaufdiagramm der Reference Architecture for Big Data Systems in the National Security Domain für das Szenario 6

Szenario 7: Ergänzen der Datensätze um neue Attribute.

Mit Hilfe des Application Orchestration Moduls können die Attribute der Daten durch den Benutzer mittels einer GUI oder einer DSL verändert werden.

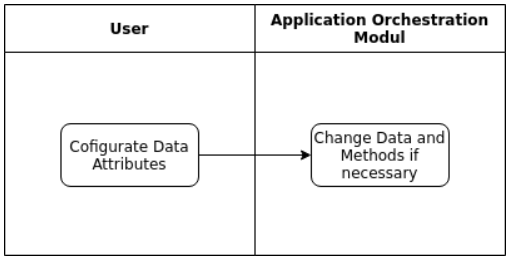

Abbildung 5.19: Ablaufdiagramm der Reference Architecture for Big Data Systems in the National Security Domain für das Szenario 7

Indirekte Szenarien

Szenario 2: Korrigieren von fehlerhaften Daten und die entsprechende Korrektur der bisher erstellten Analysen.

Die korrigierten Daten können durch den Benutzer in das Access Modul oder durch die Datenquellen im Collection Modul eingefügt werden. Von hier aus werden die Daten zur Vorverarbeitung an das Preparation Modul geleitet. Sie werden wie neue Daten vorverarbeitet, in das Data Storage Modul eingefügt und zur Analyse an das Analytics Modul weitergegeben. Durch das Processing Modul werden sie analysiert und die Ergebnisse werden über das Visualisation Modul und das Access Modul bereitgestellt.

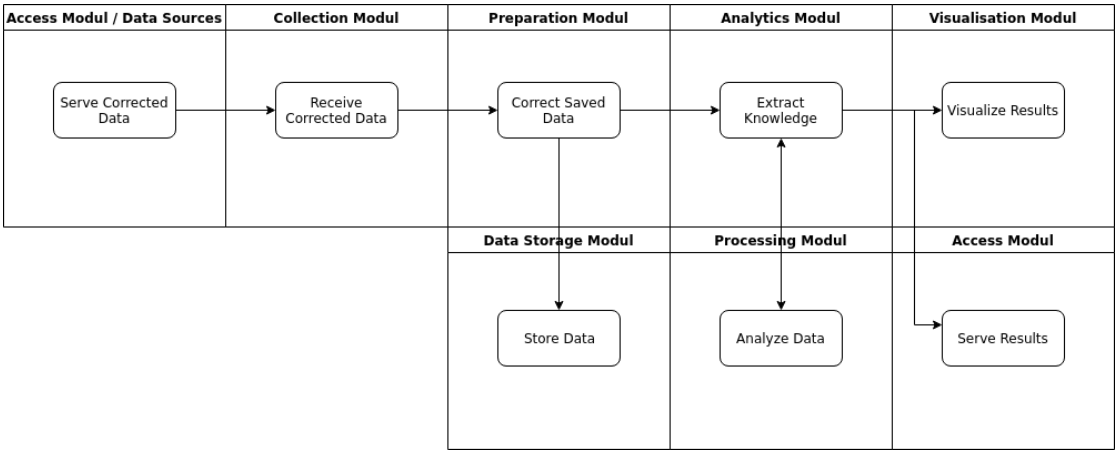

Abbildung 5.20: Ablaufdiagramm der Reference Architecture for Big Data Systems in the National Security Domain für das Szenario 2

An der Architektur sind keine Änderungen nötig, aber die Funktionalität ist nicht genau

beschrieben. Folglich wird ein geringer Änderungsaufwand veranschlagt.

Szenario 3: Löschen von Daten innerhalb des Big Data Systems, z.B. aus Datenschutzgründen.

Ein Benutzer kann anhand der Access Moduls die Löschung von Daten veranlassen. Die Anfrage wird an das Data Storage Modul weitergeleitet und die Daten werden gelöscht.

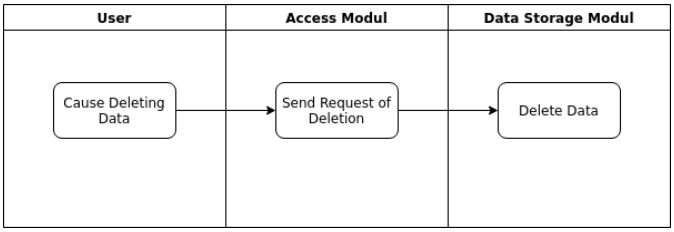

Abbildung 5.21: Ablaufdiagramm der Reference Architecture for Big Data Systems in the National Security Domain für das Szenario 3

Es sind keine Änderungen notwendig, allerdings ist keine genauere Beschreibung der Funktionalität vorhanden. Deshalb wird ein geringer Änderungsaufwand geschätzt.

Szenario 5: Abfragen von einzelnen Datensätzen, um beispielsweise Ausreißer zu identifizieren.

Der User kann mittels des Access Moduls die Daten anfragen. Dieses leitet die Anfrage an das Data Storage Modul weiter und liefert die Daten an den Benutzer zurück.

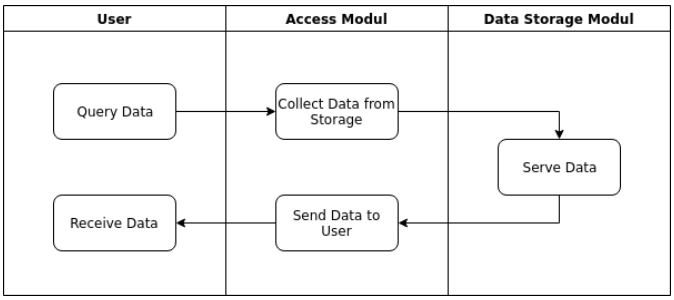

Abbildung 5.22: Ablaufdiagramm der Reference Architecture for Big Data Systems in the National Security Domain für das Szenario 5

Es ist nicht genau beschrieben, ob die Funktionalität vorhanden ist. Infolgedessen wird auch hier ein geringer Aufwand zur Änderung geschätzt.

5.5.1.3 Untersuchung der Szenariointeraktionen

Szenariointeraktionen bestehen nicht.

5.5.1.4 Erstellen der Gesamtbewertung

Die Reference Architecture for Big Data Systems in the National Security Domain konnte in der Bewertung anhand der SAAM 4 von 35 möglichen Punkten erreichen.

Tabelle 5.8: Gesamtbewertung der Reference Architecture for Big Data Systems in the National Security Domain

Szenario	**Punkte**
Szenario 1	0
Szenario 2	1
Szenario 3	1
Szenario 4	0
Szenario 5	1
Szenario 6	0
Szenario 7	1
Summe	**4**

5.5.2 Bewertung anhand nichtfunktionaler Anforderungen

5.5.2.1 Belastbarkeit und Fehlertoleranz

Da sowohl die Belastbarkeit als auch die Fehlertoleranz durch das komplette Cross-Cutting Modul sichergestellt werden, wird die Anforderung als hoch klassifiziert.

5.5.2.2 Skalierbarkeit

Eine gute Skalierbarkeit wird durch das Common Concerns Modul und im Processing Modul sichergestellt. Zudem wird im Data Storage Modul Wert auf eine gute Skalierbarkeit gelegt. Weiterhin ist ein Messaging Modul vorhanden, mit dem entfernte Prozesse

vereinfacht kommunizieren können. Aus diesem Grund wird die Anforderung Skalierbarkeit als hoch eingestuft.

5.5.2.3 Geringe Latenzzeit

Die Berechnungen werden zwar effizient, skalierbar und zuverlässig durchgeführt, jedoch ist dies bei hohen Datenmengen kein Indikator dafür, dass die Berechnungen auch in angemessener Zeit vorliegen. Folglich wird diese Anforderung als mittel kategorisiert.

5.5.2.4 Erweiterbarkeit

Durch das Application Orchestration Modul wird sichergestellt, dass neue Anwendungen innerhalb des Systems mittels GUI oder einer DSL erstellt werden können. Eine Erweiterbarkeit ist somit gegeben und diese Anforderung ist als hoch einzustufen.

5.5.2.5 Allgemeingültigkeit

Durch das Application Orchestration Modul wird eine freie Konfiguration der Anwendungen zugelassen. Das System kann für jeden Anwendungszweck genutzt werden und diese Anforderung wird als hoch klassifiziert.

5.5.2.6 Ad-Hoc-Abfragen

Da nur vorhandene Ergebnisse direkt abgefragt werden können und die Ergebnisse neuer Analysen noch nicht erstellt wurden, wird diese Anforderung als mittel eingeordnet.

5.5.2.7 Wartungsaufwand

Das System ist sinnvoll in einzelne Module aufgeteilt, die eine sehr gute Wartbarkeit aufweisen. Jedoch ist das ganze System sehr komplex aufgebaut und das Zusammenspiel der einzelnen Module nicht ganz offensichtlich. Aufgrund dessen wird diese Anforderung als mittel eingestuft.

5.5.2.8 Fehlerbehebung

Die Architektur setzt auf starke Verteilung und ist sehr komplex. Das Auffinden und Beheben von Fehlern ist nicht trivial und daher als mittel zu bewerten.

5.5.2.9 Gesamtbewertung

In der Bewertung anhand der NFA konnte die Reference Architecture for Big Data Systems in the National Security Domain mit 96 von insgesamt 108 Punkten die bisher beste Punktzahl erreichen.

Tabelle 5.9: Gesamtbewertung der Reference Architecture for Big Data Systems in the National Security Domain für die NFA

Priorität	NFA	Ergebnis
1	Belastbarkeit und Fehlertoleranz	24
2	Skalierbarkeit	21
3	Geringe Latenzzeiten	12
4	Erweiterbarkeit	15
5	Allgemeingültigkeit	12
6	Ad-Hoc-Abfragen	6
7	Minimaler Wartungsaufwand	4
8	Fehlerbehebung	2
	Score	96 / 108

5.6 Bewertung der Big Data Architecture for Automotive Applications

5.6.1 Bewertung anhand der Software Architecture Analysis Method

5.6.1.1 Klassifizierung der Szenarien

Szenario 1: Hinzufügen von Daten aus den Datenquellen in das Big Data System.

Der Frontend Layer nimmt die Daten von den TSU entgegen. Hier werden sie zur weiteren Verarbeitung im System weitergeleitet. Demnach ist für dieses Szenario keine Änderung notwendig. Es liegt ein direktes Szenario vor.

Szenario 2: Korrigieren von fehlerhaften Daten und die entsprechende Korrektur der bisher erstellten Analysen.

Fehlerhafte Daten können durch die TSU über das Frontend Layer oder durch einen Benutzer über den Device and Service Layer korrigiert werden. Ob die dafür nötige Funktionalität vorhanden ist, ist nicht beschrieben. Dadurch wird dieses Szenario als indirekt eingestuft.

Szenario 3: Löschen von Daten innerhalb des Big Data Systems, z.B. aus Datenschutzgründen.

Das Löschen von Daten kann durch einen Benutzer über den Device and Service Layer veranlasst werden. Es ist nicht beschrieben, ob diese Funktionalität vorhanden ist. Folglich handelt es sich um ein indirektes Szenario.

Szenario 4: Abfragen von Ergebnissen aus den Auswertungen.

Das Abfragen der Ergebnisse kann über den Serving Layer erfolgen. Hier ist keine Änderung notwendig und das Szenario ein direktes.

Szenario 5: Abfragen von einzelnen Datensätzen, um beispielsweise Ausreißer zu identifizieren.

Die Daten können unter Rücksichtnahme von Berechtigungen, ebenfalls über den Serving

Layer abgefragt werden. Demnach ist hier keine Änderung nötig und dieses Szenario wird als direkt klassifiziert.

Szenario 6: Hinzufügen einer neuen Methode zur Auswertung der Daten.

Die Methoden zur Auswertung müssen für jeden Layer erzeugt werden. Da dies einen erhöhten Aufwand darstellt, wird dieses Szenario als ein indirektes Szenario eingeordnet.

Szenario 7: Ergänzen der Datensätze um neue Attribute.

Die Datensätze können mit Hilfe des Device and Service Management Layers angepasst werden. Da die Daten lediglich im Speed Layer vorverarbeitet werden, muss eine eventuelle Anpassung des Verfahrens nur hier erfolgen. Jedoch müssen die Analysemethoden wenn nötig im Batch und im Speed Layer angepasst werden. Also wird dieses Szenario als indirekt bewertet.

Zur besseren Übersicht werden alle Szenarien nach Klassifizierung sortiert.

Direkte Szenarien

- Szenario 1: Hinzufügen von Daten aus den Datenquellen in das Big Data System.
- Szenario 4: Abfragen von Ergebnissen aus den Auswertungen.
- Szenario 5: Abfragen von einzelnen Datensätzen, um beispielsweise Ausreißer zu identifizieren.

Indirekte Szenarien

- Szenario 2: Korrigieren von fehlerhaften Daten und die entsprechende Korrektur der bisher erstellten Analysen.
- Szenario 3: Löschen von Daten innerhalb des Big Data Systems, z.B. aus Datenschutzgründen.
- Szenario 6: Hinzufügen einer neuen Methode zur Auswertung der Daten.
- Szenario 7: Ergänzen der Datensätze um neue Attribute.

5.6.1.2 Bewertung der einzelnen Szenarien

Direkte Szenarien

Szenario 1: Hinzufügen von Daten aus den Datenquellen in das Big Data System.

Die TSU leiten die Daten zum Frontend Layer weiter. Sie werden so transformiert, dass sie mittels Message Queue versendet werden können. Diese versendet die Daten an den Speed Layer, der eine Vorverarbeitung der Daten übernimmt. Nach der Vorverarbeitung werden die Daten vom Speed Layer analysiert und an den Batch Layer weitergeleitet, der ebenfalls Analysen berechnet. Die Ergebnisse von beiden Layern werden vom Serving Layer bereitgestellt.

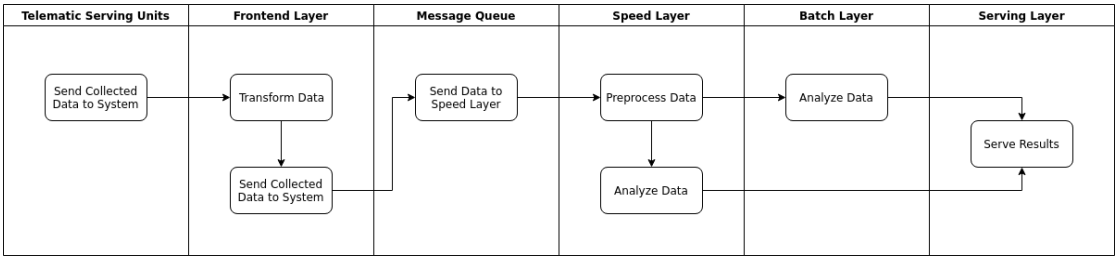

Abbildung 5.23: Ablaufdiagramm der Reference Architecture for Big Data Architecture for Automotive Applications für das Szenario 1

Szenario 4: Abfragen von Ergebnissen aus den Auswertungen.

Die Benutzer können die Ergebnisse nach der Berechnung vom Serving Layer anfragen.

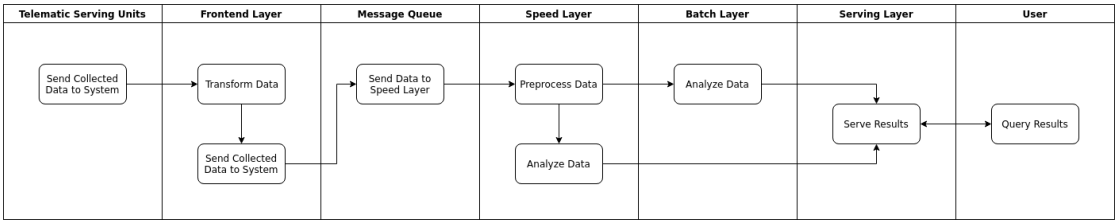

Abbildung 5.24: Ablaufdiagramm der Reference Architecture for Big Data Architecture for Automotive Applications für das Szenario 4

Szenario 5: Abfragen von einzelnen Datensätzen, um beispielsweise Ausreißer zu identifizieren.

Einzelne Datensätze können durch den Benutzer über den Device and Service Management Layer abgefragt werden.

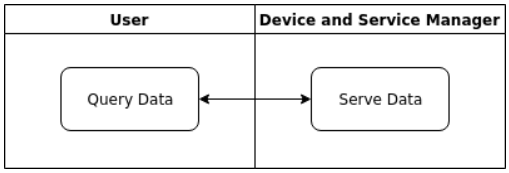

Abbildung 5.25: Ablaufdiagramm der Reference Architecture for Big Data Architecture for Automotive Applications für das Szenario 5

Indirekte Szenarien

Szenario 2: Korrigieren von fehlerhaften Daten und die entsprechende Korrektur der bisher erstellten Analysen.

Die Daten können durch den Benutzer über den Device and Service Management Layer korrigiert werden. Von hier aus werden sie in die Datenbank eingefügt und innerhalb des Batch Layers werden die Analysen erneut berechnet.

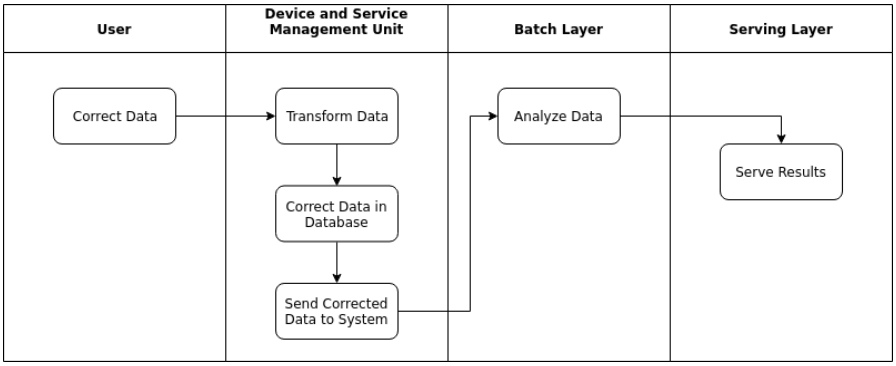

Abbildung 5.26: Ablaufdiagramm der Reference Architecture for Big Data Architecture for Automotive Applications für das Szenario 2

Da die Funktionalität in der Architektur nicht beschrieben wurde, wird dieses Szenario mit einem geringen Änderungsaufwand bewertet.

Szenario 3: Löschen von Daten innerhalb des Big Data Systems, z.B. aus Datenschutzgründen.

Der Benutzer kann das Löschen der Daten über den Device and Service Management Layer veranlassen. Die Daten werden aus der Datenbank gelöscht.

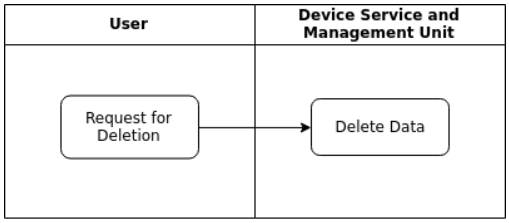

Abbildung 5.27: Ablaufdiagramm der Reference Architecture for Big Data Architecture for Automotive Applications für das Szenario 3

Da die Funktionalität in der Architektur nicht beschrieben wurde, wird dieses Szenario mit einem geringen Änderungsaufwand bewertet.

Szenario 6: Hinzufügen einer neuen Methode zur Auswertung der Daten.

Die neuen Methoden zur Auswertung der Daten müssen für den Batch und den Speed Layer erstellt und in das System eingefügt werden.

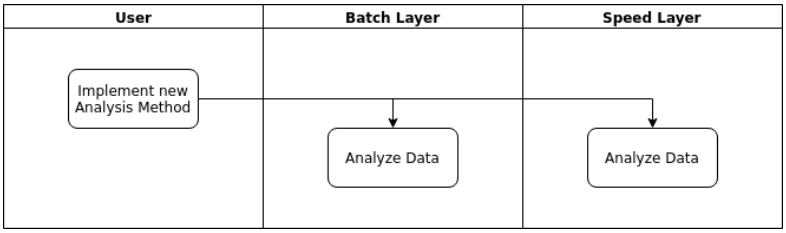

Abbildung 5.28: Ablaufdiagramm der Reference Architecture for Big Data Architecture for Automotive Applications für das Szenario 6

Da für beide Layer Methoden erstellt werden müssen, die eine ähnliche Funktionalität besitzen, wird dieses Szenario mit einem geringen Änderungsaufwand bewertet.

Szenario 7: Ergänzen der Datensätze um neue Attribute.

Der Benutzer kann über die Schnittstelle im Device and Service Management Layer den Daten neue Attribute hinzufügen. Dazu müssen der Prozess der Vorverarbeitung und die Methoden zur Analyse im Speed und Batch Layer angepasst werden.

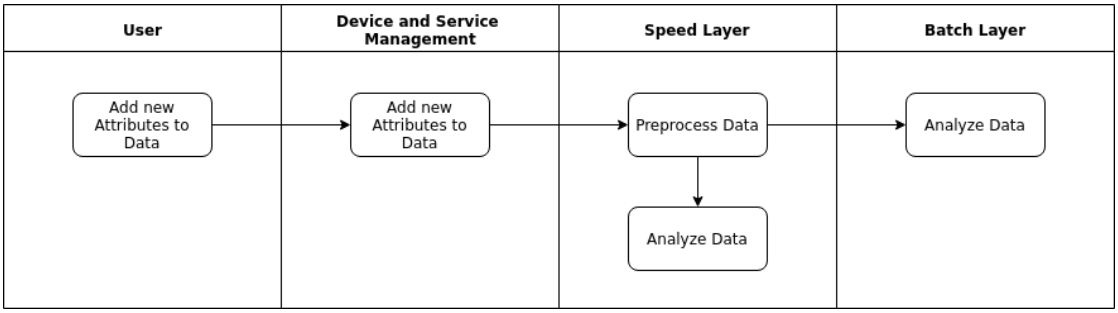

Abbildung 5.29: Ablaufdiagramm der Reference Architecture for Big Data Architecture for Automotive Applications für das Szenario 7

Da nicht bekannt ist, ob diese Funktionalität vorhanden ist und die Methoden zur Auswertung in beiden Layern angepasst werden müssen, wird das Szenario ebenfalls mit einem mittleren Änderungsaufwand bewertet.

5.6.1.3 Untersuchung der Szenariointeraktionen

Es bestehen keine Szenariointeraktionen.

5.6.1.4 Erstellen der Gesamtbewertung

Die Big Data Architecture for Automotive Applications hat in dieser Kategorie 6 von 35 möglichen Punkten erreicht und ist damit gleichauf mit der Learning Analytics Architecture, jedoch vor der Lambda Architektur und hinter der Reference Architecture for Big Data Systems in the National Security Domain.

Tabelle 5.10: Gesamtbewertung der Big Data Architecture for Automotive Applications

Szenario	**Punkte**
Szenario 1	0
Szenario 2	1
Szenario 3	1
Szenario 4	0
Szenario 5	0
Szenario 6	1
Szenario 7	3
Summe	**6**

5.6.2 Bewertung anhand nichtfunktionaler Anforderungen

5.6.2.1 Belastbarkeit und Fehlertoleranz

Durch den Message Queue Layer wird eine erhöhte Belastbarkeit geschaffen, da eine mögliche Nichtverfügbarkeit überbrückt werden kann. Da die Daten im Batch Layer zunächst vorverarbeitet und dann als unveränderlich abgespeichert werden, ist eine Fehlertoleranz vorhanden. Bei fehlerhaft implementierten Algorithmen können die Berechnungen wiederholt werden. Infolgedessen wird diese Anforderung als mittel eingestuft.

5.6.2.2 Skalierbarkeit

Wie auch bei der Lambda Architektur ist die Skalierbarkeit von den verwendeten Technologien abhängig. Folglich wird diese NFA als mittel klassifiziert.

5.6.2.3 Geringe Latenzzeit

Die Berechnung der Analysen erfolgt im Speed und im Batch Layer. Dies ermöglicht eine schnelle Bereitstellung der Berechnungen durch den Speed Layer, die von den genaueren Ergebnissen aus dem Batch Layer überschrieben werden. Deshalb ist die geringe Latenzzeit als hoch zu bewerten.

5.6.2.4 Erweiterbarkeit

Speed und Batch Layer sind zur Auswertung vorhanden und müssen bei der Erweiterung verändert werden. Das fällt jedoch nicht ganz so stark ins Gewicht, wie bei der Lambda Architektur, da der Speed Layer die Vorverarbeitung der Daten übernimmt und diese an den Batch Layer sendet. Darum wird die Erweiterbarkeit als hoch eingegliedert.

5.6.2.5 Allgemeingültigkeit

Die Allgemeingültigkeit der Anwendung ist als mittel zu kategorisieren, da die Daten über die Sensoren der Fahrzeuge gewonnen werden. Das erfordert sehr spezielle Schnittstellen, die nicht von jedem Big Data System unterstützt werden.

5.6.2.6 Ad-Hoc-Abfragen

Diese Anforderung wird als mittel eingestuft, da die Ergebnisse neuer Auswertungen nicht direkt, sondern erstmalig nach einer hohen Latenzzeit bereitstehen.

5.6.2.7 Wartungsaufwand

Da Batch und Speed Layer gewartet werden müssen und ein doppelter Aufwand entsteht, wird diese Anforderung ebenfalls als mittel bewertet.

5.6.2.8 Fehlerbehebung

Fehler müssen im Batch und im Speed Layer korrigiert werden. Aufgrund des höheren Aufwands wird die Anforderung als mittel kategorisiert.

5.6.2.9 Gesamtbewertung

In der Bewertung anhand der NFA hat die Big Data Architecture for Automotive Applications 91 von 108 Punkten erreicht. Damit ist diese Architektur in der Bewertung vor der Learning Analytics Architecture und auch vor der Lambda Architektur. Allerdings ordnet sie sich hinter der Reference Architecture for Big Data Systems in the National Security Domain ein.

Tabelle 5.11: Gesamtbewertung der Big Data Architecture for Automotive Applications für die NFA

Priorität	NFA	Ergebnis
1	Belastbarkeit und Fehlertoleranz	24
2	Skalierbarkeit	14
3	Geringe Latenzzeiten	18
4	Erweiterbarkeit	15
5	Allgemeingültigkeit	8
6	Ad-Hoc-Abfragen	6
7	Minimaler Wartungsaufwand	4
8	Fehlerbehebung	2
	Score	91 / 108

6 Fazit

Ziel dieser Arbeit war es, unterschiedliche Big Data Architekturen vorzustellen und diese zu bewerten. In Kapitel 2 wurden zunächst Grundlagen zu Big Data vermittelt. Die Vorstellung der Architekturen erfolgte in Kapitel 3. In Kapitel 4 wurde beschrieben, wie die vorgestellten Architekturen implementiert werden können. Kapitel 5 beschäftigte sich mit der Vorstellung von Methoden zur Bewertung von Softwarearchitekturen. Anschließend wurde die Bewertung vorgenommen. Dazu wurden zwei Methoden ausgewählt, die zum einen die gewünschten Anforderungen von Big Data Systemen und zum anderen Szenarien betrachten, die im laufenden Betrieb auftreten können. Im Zuge dessen wurde bewertet, in welchem Maße die Architekturen die Anforderungen erfüllen und wie die Szenarien durch die Architekturen abgebildet werden oder wie hoch der Aufwand eine Änderung wäre.

6.1 Vergleich der Implementierungen

Vergleicht man den Aufwand der Implementierung jeder einzelnen Architektur miteinander, fällt auf, dass die Reference Architecture for Big Data Systems in the National Security Domain sehr einfach umzusetzen war, obwohl dies die umfangreichste der vorgestellten Architekturen war. Die Lambda Architektur und die Big Data Architecture for Automotive Applications waren sowohl in der Architektur, als auch in der Implementierung sehr ähnlich und daher nahezu gleichzusetzen. Die Implementierung der Learning Analytics Architecture war ebenfalls einfach umzusetzen, was die Einfachheit der Architektur bereits vermuten lies.

6.2 Gesamtbetrachtung der Architekturbewertung

Die Bewertung hat gezeigt, dass die Architekturen, die sowohl schnelle als auch genaue Ergebnisse liefern, ein Defizit in der Änderbarkeit und Erweiterbarkeit haben. Dies liegt daran, dass die Analysemethoden doppelt implementiert und erweitert werden müssen. Des Weiteren fällt auf, dass die Architekturen das Löschen und Korrigieren der Daten oft nur unzureichend genau behandeln.

Tabelle 6.1: Gesamtbewertungen aller Architekturen anhand der SAAM

	Punkte			
Szenario	**A. 1**	**A. 2**	**A. 3**	**A. 4**
Szenario 1	1	1	0	0
Szenario 2	0	1	1	1
Szenario 3	1	1	1	1
Szenario 4	0	1	0	0
Szenario 5	1	0	1	0
Szenario 6	3	1	0	1
Szenario 7	3	1	1	3
Summe	**9**	**6**	**4**	**6**

Tabelle 6.2: Gesamtbewertung aller Architekturen für die NFA

Priorität	NFA	Ergebnis			
		A. 1	A. 2	A. 3	A. 4
1	Belastbarkeit und Fehlertoleranz	24	16	24	24
2	Skalierbarkeit	14	14	21	14
3	Geringe Latenzzeiten	18	6	12	18
4	Erweiterbarkeit	10	10	15	15
5	Allgemeingültigkeit	12	8	12	8
6	Ad-Hoc-Abfragen	6	3	6	6
7	Minimaler Wartungsaufwand	4	4	4	4
8	Fehlerbehebung	2	2	2	2
	Score	90 / 108	63 / 108	96 / 108	91 / 108

Legende:

- **A. 1:** Lambda Architektur
- **A. 2:** Learning Analytics
- **A. 3:** Reference Architecture for Big Data Systems in the National Security Domain
- **A. 4:** Big Data Architecture for Automotive Applications

Ein und dieselbe Architektur erzeugte je nach Bewertungsmethode häufig unterschiedliche Ergebnisse. Lediglich eine Architektur konnte in der Bewertung konstant gute Ergebnisse erzielen. Dies liegt daran, dass in der Lambda Architektur und in der Big Data Architecture for Automotive Applications die Änderbarkeit und Erweiterbarkeit jeweils schlechter bewertet wurden. Grund dafür ist, dass die Auswertungen mittels zwei unterschiedlicher Methoden berechnet werden. Durch diese Berechnung können die Ergebnisse schneller bereitgestellt werden. Allerdings bedeutet das auch einen doppelten Aufwand für die Wartbarkeit und Änderbarkeit der Architekturen. Eine Abwägung der gewünschten Anforderungen und die damit einhergehende Priorisierung ist zwingend erforderlich und muss für jeden Anwendungsfall mit Bedacht vorgenommen werden.

Nimmt man beide Bewertungen zusammen, lässt sich folgende Rangfolge der Architekturen erstellen:

1. Reference Architecture for Big Data Systems in the National Security Domain
2. Big Data Architecture for Automotive Applications
3. Lambda Architecture
4. Learning Analytics Architecture

Die Learning Analytics Architecture belegt den letzten Platz, da der Abstand zu den anderen Architekturen in der Bewertung anhand der NFA zu groß war.

6.3 Ausblick

Sowohl bei der Implementierung als auch bei der Bewertung konnte die Reference Architecture for Big Data Systems in the National Security Domain überzeugen. Das mag an der Microservice ähnlichen Architektur liegen. Durch die Art der Architektur lässt sich das System sehr gut verteilt implementieren und ist daher auch sehr gut skalierbar. Durch den verteilten Betrieb lässt sich ebenfalls die Belastbarkeit des Systems steigern. Zusätzlich ist eine gute Wartbarkeit gegeben, da die einzelnen Module übersichtlicher sind und nur kleine Aufgaben übernehmen. Weitere Funktionalitäten, die die Sicherheit oder Stabilität des Systems unterstützen, können einfach ergänzt werden. So ist das System sehr gut erweiterbar und auch allgemeingültig einsetzbar. Durch geeignete Schnittstellen können Ad-Hoc-Abfragen unterstützt werden.

In dieser Arbeit konnte eine Methode zur Bewertung von Big Data Architekturen erstellt werden, die sowohl gewünschte Eigenschaften als auch auftretende Szenarien in der Nutzungsphase der Systeme berücksichtigt. Die angewendeten Methoden müssen aber für jede Bewertung angepasst werden. Die Szenarien und die gewünschten Eigenschaften sollten kritisch und unter Einbeziehung aller Nutzerbereiche gewählt und ausgewertet werden. Für eventuelle Rückfragen zur Architektur sollte zwingend ein beteiligter Softwarearchitekt für die Bewertung anwesend sein. Die Methoden sollten früh in den Entwicklungsprozess eingebunden werden, da die Architektur präzisiert und damit leichter verständlich festgehalten werden kann. Das trägt zur effizienteren Implementierung und einer besseren Wartbarkeit bei.

Eine weitere Erkenntnis dieser Arbeit ist, dass sich die Programmiersprache Elixir zur Implementierung von Big Data Systemen sehr gut eignet. Aufgrund der leichtgewichtigen Prozesse und der einfachen Möglichkeit, Nachrichten zwischen den verteilten Prozessen, die logisch und auch physisch voneinander getrennt sein können, auszutauschen, können Systeme implementiert werden, die genau die Anforderungen von Big Data Systemen erfüllen. Jedoch sind nicht für alle Datenbankmanagementsysteme Treiber vorhanden, sodass man entweder stark eingeschränkt ist, oder eigene Module dafür entwickeln muss.

Literaturverzeichnis

[1] Andresen, J.: *ATAM: Welchen Preis zahle ich?* https://www.judithandresen.com/2012/04/01/atam-welchen-preis-zahle-ich/. – Zugegriffen am: 06.05.2019

[2] Anger, R.: *Vom Großen zum Ganzen – Architektur-Reviews mit ATAM & Co.* https://www.sigs-datacom.de/uploads/tx_dmjournals/anger_OS_Architekturen_12_j6tf8.pdf. – Zugegriffen am: 03.05.2019

[3] Apache Software Foundation: *Apache Cassandra Dokumentation.* http://cassandra.apache.org/. – Zugegriffen am: 19.06.2019

[4] Bondi, A.: Characteristics of Scalability and Their Impact on Performance. In: *WOSP2000: Second International Workshop on Software and Performance*, ACM, 2000, S. 195 – 203. – URL https://dl.acm.org/citation.cfm?id=350432. – ISBN 1-58113-195-X

[5] Byrnes, C. ; Kyratzoglou, I.: *Applying Architecture Tradeoff Assessment Method (ATAM) As Part Of Formal Software Architecture Review.* https://www.mitre.org/sites/default/files/pdf/07_0094.pdf. – Zugegriffen am: 03.05.2019

[6] Clark, John O.: System of Systems Engineering and Family of Systems Engineering From a Standards, V-Model, and Dual-V Model Perspective. In: *IEEE SysCon 2009 —3rd Annual IEEE International Systems Conference, 2009*, IEEE, 2009, S. 381 – 387. – URL https://ieeexplore.ieee.org/document/4815831. – ISBN 978-1-4244-3463-3

[7] Clements, P. ; Kazman, R. ; Klein, M.: *Evaluating Software Architectures.* Addison-Wesley, 2001. – 211 – 220 S. – ISBN 0-201-70482-X

[8] Dessables, F. ; Haroun, A. ; Mostefaoui, A.: A Big Data Architecture for Automotive Applications: PSA Group Deployment Experience. In: *CCGrid '17 Proceedings of the 17th IEEE/ACM International Symposium on Cluster, Cloud*

and Grid Computing, IEEE Press Piscataway, 2017, S. 921 – 928. – URL https://dl.acm.org/citation.cfm?id=3101254. – ISBN 978-1-5090-6610-0

[9] Deutscher Wetterdienst: *Historische stündliche Stationsmessungen der Lufttemperatur und Luftfeuchte für Deutschland.* ftp://ftp-cdc.dwd.de/pub/CDC/observations_germany/climate/hourly/air_temperature/historical/BESCHREIBUNG_obsgermany_climate_hourly_tu_historical_de.pdf. – Zugegriffen am: 23.05.2019

[10] Ericsson AB.: *Distributed Erlang.* http://erlang.org/doc/reference_manual/distributed.html. – Zugegriffen am: 09.07.2019

[11] Ericsson AB.: *Mnesia Dokumentation.* http://erlang.org/doc/man/mnesia.html. – Zugegriffen am: 19.06.2019

[12] EWE AG: *Smart Data: Die fabelhafte Welt der Zahlen.* https://www.hallonachbar.de/de/energiewende/smart-data. – Zugegriffen am: 02.05.2019

[13] Fasel, D. ; Meier, A.: *Big Data - Grundlagen, Systeme und Nutzungspotentiale.* Springer Vieweg, 2016. – URL https://link.springer.com/book/10.1007/978-3-658-11589-0. – ISBN 978-3-658-11589-0

[14] Industry Analytics: *Big Data in der Kritik: Eine kritische Betrachtung.* http://www.industry-analytics.de/big-data-eine-kritische-betrachtung/. – Zugegriffen am: 02.05.2019

[15] Klein, J. et a.: A Reference Architecture for Big Data Systems in the National Security Domain. In: *2016 IEEE/ACM 2nd International Workshop on Big Data Software Engineering (BIGDSE)*, IEEE, 2016, S. 51 – 57. – URL https://ieeexplore.ieee.org/document/7811387. – ISBN 978-1-4503-4152-3

[16] König, C. ; Schröder, J.: *Big Data - Chancen, Risiken, Entwicklungstendenzen.* Wiegand, E. - Springer-Verlag GmbH, 2018. – 117 f. S. – URL https://link.springer.com/book/10.1007/978-3-658-20083-1. – ISBN 978-3-658-20083-1

[17] Manyika, J. et a.: *Big Date: The next frontier for innovation, competition, and productivity.* 2011. – URL https://www.mckinsey.com/business-functions/digital-mckinsey/our-insights/big-data-the-next-frontier-for-innovation

[18] Marz, N. ; Warren, J.: *Big Data - Entwicklung und Programmierung von Systemen für große Datenmengen und Einsatz der Lambda-Architektur*. mitp Verlags GmbH & Co. KG, 2016. – ISBN 978-3-95845-175-9

[19] Matsebula, F. ; Mnkandla, E.: A Big Data Architecture for Learning Analytics in Higher Education. In: *2017 IEEE Africon - Science, Technology and Innovation for Africa*, IEEE, 2017, S. 951 – 956. – URL https://ieeexplore.ieee.org/document/8095610. – ISBN 978-1-5386-2774-4

[20] Meier, A. ; Kaufmann, M.: *SQL- & NoSQL-Datenbanken*. Springer-Verlag GmbH, 2016. – 13 S. – URL https://link.springer.com/book/10.1007%2F978-3-662-47664-2. – ISBN 978-3-662-47664-2

[21] Müller, S. IT-Novum: *7 Beispiele für erfolgreiche BI- und Big-Data-Projekte*. https://www.bigdata-insider.de/7-beispiele-fuer-erfolgreiche-bi-und-big-data-projekte-a-752259/. – Zugegriffen am: 02.05.2019

[22] Nelson, P.: *Sechs Nutzungsfälle für Big Data im modernen Unternehmen*. https://www.searchtechnologies.com/de/blog/big-data-nutzungsfaelle-fuer-unternehmen. – Zugegriffen am: 02.05.2019

[23] Netflix Technology Blog: *Benchmarking Cassandra Scalability on AWS — Over a million writes per second*. https://medium.com/netflix-techblog/benchmarking-cassandra-scalability-on-aws-over-a-million-writes-per-secon – Zugegriffen am: 19.06.2019

[24] Plataformatec: *Elixir Documentation*. https://elixir-lang.org/. – Zugegriffen am: 19.06.2019

[25] Plataformatec: *Elixir Documentation - Distributed tasks and configuration*. https://elixir-lang.org/getting-started/mix-otp/distributed-tasks-and-configuration.html. – Zugegriffen am: 19.06.2019

[26] Posch, T. ; Birken, K. ; Gerdom, M.: *Softwarearchitektur - Verstehen, entwerfen, bewerten und dokumentieren*. dpunkt.verlag GmbH, 2004. – 180 – 191 S. – ISBN 3-89864-270-4

[27] Seufert, Andreas ; Sexl, Stefan: Competing on Analytics –Wettbewerbsvorsprung durch Business Intelligence. In: Gleich, Ronald (Hrsg.): *Challenge Controlling 2015*. Freiburg i. Br.; Freiburg i. Br. : Haufe Verlag, 2011 (Der Controlling-Berater 17), S. 201–218. – URL http://hdl.handle.net/10419/124188

[28] Software Engineering Institute: *John Klein*. https://resources.sei.cmu.edu/library/author.cfm?authorID=3271. – Zugegriffen am: 15.07.2019

A Anmerkungen zum Code

Die für die Arbeit genutzen Temperaturdaten können hier gefunden werden:
ftp://opendata.dwd.de/climate_environment/CDC/observations_germany/climate/hourly/air_temperature/historical/

Der für die Arbeit erstellte Code befindet sich in diesem GitHub-Repository:
https://github.com/AndreB87/ba_thesis
Zudem befindet sich im Verzeichnis crawler ein Python Script, welches zum automatischen Download der Daten genutzt werden kann.

A.1 Installation

Um die implementierten Architekturen starten zu können, müssen Elixir und Apache Cassandra installiert werden. Um Elixir zu installieren, müssen die Schritte in der Elixir Documentation ausgeführt werden. Diese Dokumentation ist hier zu finden:
https://elixir-lang.org/install.html

Apache Cassandra kann sowohl als Docker Container, als auch direkt betrieben werden. Hier befindet sich die offizielle Homepage von Cassandra mit dem Download der Installationsdatei:
http://cassandra.apache.org/

A.2 Installation der Abhängigkeiten

Um die benötigten Abhängigkeiten einer implementierten Architektur zu installieren ist es notwendig, mit der Shell oder einer ähnlichen Anwendung in den jeweiligen Ordner mit der Implementierung zu wechseln und folgenden Befehl einzugeben:

```
mix deps.get
```

Daraufhin werden alle benötigten Abhängigkeiten direkt heruntergeladen. Da auch einige Abhängigkeiten unter den selbst implementierten Modulen bestehen, ist es wichtig, die Ordnerstruktur nicht zu verändern.

A.3 Starten des Batch Processing Moduls

Um das Batch Processing Modul zu starten, muss man lediglich mit der Shell im Ordner des Batch Processing Moduls folgenden Befehl eingeben:

```
elixir --name batch@<IP-Adresse> -S mix run -e "BatchProcessing.start()"
 --no-halt
```

Damit wird das Batch Processing Modul in einer eigenen Node gestartet und ist auch für Prozesse auf Computern innerhalb des Netzwerks erreichbar.

A.4 Starten des Stream Processing Moduls

Um das Stream Processing Modul zu starten, muss man lediglich mit der Shell im Ordner des Stream Processing Moduls folgenden Befehl eingeben:

```
elixir --name stream@<IP-Adresse> -S mix run -e "StreamProcessing.start()"
 --no-halt
```

Damit wird das Stream Processing Modul in einer eigenen Node gestartet und ist auch für Prozesse auf Computern innerhalb des Netzwerks erreichbar.

A.5 Starten des Mnesia Moduls

Um das Mnesia Modul zu starten, muss man lediglich mit der Shell im Ordner des Mnesia Moduls folgenden Befehl eingeben:

```
elixir --name mnesia@<IP-Adresse> -S mix run -e "MnesiaServer.start()"
 --no-halt
```

Damit wird das Mnesia Modul in einer eigenen Node gestartet und ist auch für Prozesse auf Computern innerhalb des Netzwerks erreichbar.

A.6 Starten der implementierten Architekturen

Um eine implementierte Architektur zu starten muss zunächst die Konfigurationsdatei angepasst werden. Diese befindet sich im jeweiligen Ordner der Architektur im Unterordner "config und ist eine Datei mit dem Namen "config.exs. Hier muss jeweils der Eintrag zur Node angepasst werden. Zudem muss für den Eintrag zum Batch Processing Modul der Eintrag für den db_host mit der IP-Adresse und dem Port unter dem die Cassandra Datenbank zu erreichen ist, angepasst werden.

Um die Architektur zu starten muss folgender Befehl eingegeben werden.

```
iex -S mix
```

Für jede Architektur ist zudem ein Script erstellt worden, dass alle benötigten Services startet. Hierbei ist jedoch zu beachten, dass diese Services dann nicht von anderen Rechnern innerhalb des Netzwerks angesprochen werden können.

Damit startet eine Elixir Interactive Shell, in der man dann jede Architektur mit den folgenden Befehlen starten kann:

Lambda Architektur

Für die Lambda Architektur müssen zuerst das Batch Processing Modul, das Stream Processing Modul und das Mnesia Modul gestartet werden.

```
iex > Lambda.init()
iex > {:ok, lambda_pid} = Lambda.start()
```

Damit wird die Lambda Architektur gestartet. Um die Dateien einzulesen, muss folgender Befehl eingegeben werden.

```
iex > Lambda.read_data(lambda_pid, "../../Daten/")
```

Danach wird das Einlesen der Daten gestartet und die Ergebnisse können folgendermaßen abgerufen werden.

```
# Gesamtdurchschnitt
iex > Lambda.get_result(lambda_pid)
# Durchschnitt für das jeweilige Jahr
iex > Lambda.get_result(lambda_pid, year)
# Durchschnitt für den jeweiligen Monat im Jahr
iex > Lambda.get_result(lambda_pid, year, month)
# Durchschnitt für den jeweiligen Tag des Monats im Jahr
iex > Lambda.get_result(lambda_pid, year, month, day)
# Gesamtdurchschnitt der jeweiligen Station
iex > Lambda.get_station_result(lambda_pid, station)
# Durchschnitt der jeweiligen Station für das gegebene Jahr
iex > Lambda.get_station_result(lambda_pid, station, year)
# Durchschnitt der jeweiligen Station für den gegebenen Monat im Jahr
iex > Lambda.get_station_result(lambda_pid, station, year, month)
# Durchschnitt der jeweiligen Station für den gegebenen Tag des Monats
# im Jahr
iex > Lambda.get_station_result(lambda_pid, station, year, month, day)
```

Learning Analytics Architecture

Für die Learning Analytics Architecture müssen zuerst das Batch Processing Modul und das Mnesia Modul gestartet werden.

```
iex > LearningArchitecture.init()
iex > {:ok, la_pid} = LearningArchitecture.start()
```

Damit wird die Learning Analytics Architecture gestartet. Um die Dateien einzulesen, muss folgender Befehl eingegeben werden.

```
iex > LearningArchitecture.read_data(la_pid, "../../Daten/")
```

Danach wird das Einlesen der Daten gestartet und die Ergebnisse können folgendermaßen abgerufen werden.

```
# Gesamtdurchschnitt
iex > Lambda.get_result(lambda_pid)
# Durchschnitt für das jeweilige Jahr
iex > Lambda.get_result(lambda_pid, year)
# Durchschnitt für den jeweiligen Monat im Jahr
iex > Lambda.get_result(lambda_pid, year, month)
# Durchschnitt für den jeweiligen Tag des Monats im Jahr
iex > Lambda.get_result(lambda_pid, year, month, day)
# Gesamtdurchschnitt der jeweiligen Station
iex > Lambda.get_station_result(lambda_pid, station)
# Durchschnitt der jeweiligen Station für das gegebene Jahr
iex > Lambda.get_station_result(lambda_pid, station, year)
# Durchschnitt der jeweiligen Station für den gegebenen Monat im Jahr
iex > Lambda.get_station_result(lambda_pid, station, year, month)
# Durchschnitt der jeweiligen Station für den gegebenen Tag des Monats
# im Jahr
iex > Lambda.get_station_result(lambda_pid, station, year, month, day)
```

Reference Architecture for Big Data Systems in the National Security Domain

Für die Reference Architecture for Big Data Systems in the National Security Domain müssen zuerst das Batch Processing Modul und das Mnesia Modul gestartet werden.

```
iex > NationalSecurity.init()
iex > {:ok, pid} = Lambda.start()
```

Damit wird die Reference Architecture for Big Data Systems in the National Security Domain gestartet. Hier wird das Einlesen der Daten direkt gestartet.

```
# Gesamtdurchschnitt
iex > Lambda.get_result(lambda_pid)
# Durchschnitt für das jeweilige Jahr
iex > Lambda.get_result(lambda_pid, year)
# Durchschnitt für den jeweiligen Monat im Jahr
iex > Lambda.get_result(lambda_pid, year, month)
# Durchschnitt für den jeweiligen Tag des Monats im Jahr
iex > Lambda.get_result(lambda_pid, year, month, day)
# Gesamtdurchschnitt der jeweiligen Station
iex > Lambda.get_station_result(lambda_pid, station)
# Durchschnitt der jeweiligen Station für das gegebene Jahr
iex > Lambda.get_station_result(lambda_pid, station, year)
# Durchschnitt der jeweiligen Station für den gegebenen Monat im Jahr
iex > Lambda.get_station_result(lambda_pid, station, year, month)
# Durchschnitt der jeweiligen Station für den gegebenen Tag des Monats
# im Jahr
iex > Lambda.get_station_result(lambda_pid, station, year, month, day)
```

Big Data Architecture for Automotive Applications

Für die Big Data Architecture for Automotive Applications müssen zuerst das Batch Processing Modul, das Stream Processing Modul und das Mnesia Modul gestartet werden.

```
iex > Automotive.init()
iex > {:ok, automotive_pid} = Automotive.start()
```

Damit wird die Lambda Architektur gestartet. Um die Dateien einzulesen, muss folgender Befehl eingegeben werden.

```
iex > Automotive.read_data(automotive_pid, "../../Daten/")
```

Danach wird das Einlesen der Daten gestartet und die Ergebnisse können folgendermaßen abgerufen werden.

```
# Gesamtdurchschnitt
iex > Automotive.get_result(lambda_pid)
# Durchschnitt für das jeweilige Jahr
iex > Automotive.get_result(lambda_pid, year)
# Durchschnitt für den jeweiligen Monat im Jahr
iex > Automotive.get_result(lambda_pid, year, month)
# Durchschnitt für den jeweiligen Tag des Monats im Jahr
iex > Automotive.get_result(lambda_pid, year, month, day)
# Gesamtdurchschnitt der jeweiligen Station
iex > Automotive.get_station_result(lambda_pid, station)
# Durchschnitt der jeweiligen Station für das gegebene Jahr
iex > Automotive.get_station_result(lambda_pid, station, year)
# Durchschnitt der jeweiligen Station für den gegebenen Monat im Jahr
iex > Automotive.get_station_result(lambda_pid, station, year, month)
# Durchschnitt der jeweiligen Station für den gegebenen Tag des Monats
# im Jahr
iex > Automotive.get_station_result(lambda_pid, station, year, month, day)
```

Glossar

Cassandra Eine skalierbare und fehlertolerante NoSQL-Datenbank. [3].

Elixir Eine funktionale Programmiersprache, die in der Erlang Virtual Machine läuft. [24].

Mnesia Ein verteiltes Datenbankmanagementsystem, das in Erlang implementiert wurde. [11].

Node Ein Laufzeitsystem der Programmiersprache Erlang. [10].

System of Systems Ein System, dass mehrere unabhängige Systeme verbindet. [6].

Erklärung zur selbstständigen Bearbeitung einer Abschlussarbeit

Gemäß der Allgemeinen Prüfungs- und Studienordnung ist zusammen mit der Abschlussarbeit eine schriftliche Erklärung abzugeben, in der der Studierende bestätigt, dass die Abschlussarbeit „– bei einer Gruppenarbeit die entsprechend gekennzeichneten Teile der Arbeit [(§ 18 Abs. 1 APSO-TI-BM bzw. § 21 Abs. 1 APSO-INGI)] – ohne fremde Hilfe selbständig verfasst und nur die angegebenen Quellen und Hilfsmittel benutzt wurden. Wörtlich oder dem Sinn nach aus anderen Werken entnommene Stellen sind unter Angabe der Quellen kenntlich zu machen."

Quelle: § 16 Abs. 5 APSO-TI-BM bzw. § 15 Abs. 6 APSO-INGI

Erklärung zur selbstständigen Bearbeitung der Arbeit

Hiermit versichere ich,

Name: _____

Vorname: _____

dass ich die vorliegende Bachelorarbeit – bzw. bei einer Gruppenarbeit die entsprechend gekennzeichneten Teile der Arbeit – mit dem Thema:

Bewertung von Architekturalternativen für Big Data Systeme

ohne fremde Hilfe selbständig verfasst und nur die angegebenen Quellen und Hilfsmittel benutzt habe. Wörtlich oder dem Sinn nach aus anderen Werken entnommene Stellen sind unter Angabe der Quellen kenntlich gemacht.

_____	_____	_____
Ort	Datum	Unterschrift im Original