Aufbau einer Integrationsarchitektur für ein Auslastungsanalysetool in Kibana

Burak Kayaalp

28. September 2020

Version: 1.0

LEHRSTUHL FÜR WIRTSCHAFTSINFORMATIK
(PROZESSE UND SYSTEME)
AN DER UNIVERSITÄT POTSDAM

BACHELORARBEIT

Aufbau einer Integrationsarchitektur für ein Auslastungsanalysetool in Kibana

Entwurf, Entwicklung und Integration eines ML-Vorhersage- und Visualisierungssystems für Produktinformationsaktualisierungen in einer verteilten Umgebung bei der BSH Hausgeräte GmbH

Autor:
Burak KAYAALP (776813),
Wirtschaftsinformatik B.Sc.

Gutachter:
DR. ELDAR SULTANOW
DR. ANDRÉ ULLRICH

Abschlussarbeit zur Erlangung des akademischen Grades
Bachelor of Science (B.Sc.) in Wirtschaftsinformatik

in Kooperation mit

CAPGEMINI DEUTSCHLAND GMBH
BSH HAUSGERÄTE GMBH

28. September 2020

Zusammenfassung

In dieser Arbeit wird ein Prototyp eines Systems zur Visualisierung verteilter Produktinformationswarteschlangen und die Integration in existierende Datenvisualisierungstools bei einem der größten globalen Haushaltsgerätehersteller, der BSH Hausgeräte GmbH (BSH), vorgestellt. Es wird die Action Design Research Methodik angewandt, um ein brauchbares Artefakt als technologiebasierte Lösung für ein wichtiges und relevantes Geschäftsproblem zu entwickeln. Entwickelt wird ein System für die Visualisierung für Vorhersagedaten und Warteschlangen-Auslastungs-Metriken im Kontext von Produktinformationsaktualisierugs- (PIA) Warteschlangen unter Verwendung neuartiger Lösungen, wie z.B. Machinelles Lernen und Visualisierung. Die Ergebnisse liefern wertvolle Erkenntnisse über die praktische Entwicklung und Integration dieses Systems für ähnliche Umgebungen hinsichtlich der Komplexität verteilter Umgebungen und der Unternehmensgröße.

Schlüsselwörter: Produktinformationsmanagement (PIM), Machinelles Lernen (ML), Künstliche Intelligenz (KI), Monitoring, Visualisierung, Kibana, Plugin, Prediction, Warteschlangen, React

Inhaltsverzeichnis

Abkürzungsverzeichnis

WMS Warteschlangen Monitoring System

ADR Action Design Research

BIE Build Intervention Evaluation

MTTR Mean time to recovery

BSH BSH Hausgeräte GmbH

PIA Produktinformationsaktualisierung

ML Maschinelles Lernen

UI User Interface

IVA Informationsverarbeitungsankunftsstrom

IV Informationsverarbeitung

WZE Warteschlangenzustandserfassung

PIM Product Information Management

PI Produktinformation

ERP Enterprise-Resource-Planning

B2B Business-to-Business

B2C Business-to-Consumer

D2C Direct-to-Consumer

I/O Input/Output

TTM Time-to-Market

REST Representational State Transfer

API Application Programming Interface

SLA Service-Level Agreements

IT Informationstechnologie

Q Queue

ESSQL Elasticsearch SQL

TCP Transmission Control Protocol

SPA Single-Page-Applikationen

GUI Graphical User Isnterface

CSS Cascading Style Sheets

DSL Domain Specific Language

POC Proof of Concept

JSON JavaScript Object Notation

Abbildungsverzeichnis

Tabellenverzeichnis

Listings

1 Einleitung

1.1 Motivation

Das Warten in Warteschlangen ist ein fester Bestandteil des Alltags des Menschen in der Gegenwart. Ob beim Kauf einer Eintrittskarte im Kino, um eine Kontoabhebung durchzuführen oder, wenn wir im Supermarkt zur Kasse gehen, warten wir in Warteschlangen. Oft sind die Warteschlangen aufgrund von Überlastung oder Staus überfüllt. Jedes Mal, wenn eine größere Nachfrage nach einer Dienstleistung besteht, als angeboten werden kann, bildet sich eine Warteschlange. In vielen Geschäftssituationen, insbesondere in kundenorientierten, ist es möglich, die Dauer der Warteschlange vorherzusagen. Man kann beobachten, wie viele Personen in der Schlange stehen und abschätzen, wie schnell die Warteschlange äbgearbeitet"wird. Bei Warteschlangen in Systemen der Informationstechnologie (IT) stellt sich die Situation jedoch anders dar. Um eine Überlastung eines Systems zu vermeiden, ist es notwendig, bestimmte Verarbeitungsaktivitäten einzuschränken, damit das System innerhalb des Service-Level Agreements[1] (SLA) bleibt. Zum rechtzeitigen Handeln bei Vorkommen von Anomalien in den Verabeitungsprozessen können diese einem stetigen Monitoring vollzogen werden. Das Ziel dieses Monitorings besteht in der Echtzeitüberwachung der beobachteten Prozesse, mit dem Ziel, Abweichungen oder Parameter außerhalb der gewünschten Bereiche unverzüglich zu identifizieren, lokalisieren und diese zu beseitigen bevor größere Probleme entstehen [Jp12].

Ein großer Nachteil des bestehenden Monitoringprozess bei der BSH ist, dass externe Beobachter keinen Einblick in den Fortschritt der allgemeinen Verarbeitungsaktivität bzw. die der Elemente, die im speziellen von Interesse sind. Viele Systeme sind in der Lage den Fortschritt der Verarbeitung in Form von Fortschrittsbalken zu visualisieren, jedoch ist dies nicht hilfreich, wenn die genaue Position eines Verarbeitungspostens von Bedeutung ist. Wartezeit ist in diesem Kontext keine werftschöpfende Zeit, da in dieser Zeit selten produktive Aktivi-

[1] Dienstgütevereinbarung [Siep20]

täten durchgeführt werden können. Verstärkt wird dieses Problem in Geschäftsumgebungen, wenn bestimmte Prozesse erst dann durchgeführt werden können, wenn bestimmte wichtige Elemente die Verarbeitung in einer Warteschlange durchlaufen haben. Chircu et al. stellte durch qualitative Interviews einen klaren Bedarf nach einem neuen System zur Vorhersage und Visualisierung verteilter Produktinformationswarteschlangen bei der BSH fest [Chi19].

1.2 Unternehmensprofil

Die BSH Hausgeräte GmbH (BSH) ist eines der weltweit führenden Unternehmen der Branche und der größte Haushaltsgerätehersteller in Europa. Der Umsatz des Unternehmens beläuft sich auf über 13 Milliarden Euro und 58.200 (Stand: 2019) Mitarbeiter beschäftigt das Unternehmen. Die BSH hat 40 Produktionsstandorte weltweit. Das Produktportfolio der 11 Marken umfasst das gesamte Spektrum moderner Hausgeräte. Es reicht von Herden, Backöfen und Dunstabzugshauben über Geschirrspüler, Waschmaschinen, Trockner, Kühl- und Gefrierschränke bis hin zu kleinen Hausgeräten wie Staubsaugern, Kaffeevollautomaten, Wasserkochern und Bügeleisen [Mar20]. Für den Vertrieb ihrer Produkte nutzt die BSH sowohl Business-to-Business- (B2B) als auch Business-to-Consumer-Kanäle (B2C) und stützt sich bei der Verteilung von Produktinformationen an die jeweiligen Vertriebskanäle auf ein Product Information Management-System (PIM) [Bsh20, Chi19].

1.3 Problemstellung und Zielsetzung

Zur Verdeutlichung der Problematik werden im folgenden zwei Szenarien aus dem operativen Umfeld der BSH vorgestellt. Wird ein Fehler in Produktinformationen identifiziert, so wird durch Produktsupport-Spezialisten eine Fehlerbehebung vorgenommen. Ein Beispiel für dieses Szenario ist das Fehlen von Mediendaten bei einem Produkt. Für die Planung von Aufwänden im Fehlerbehebungsprozess und zur frühzeitigen Benachrichtigung von Kunden, ist die Information, wann ein Problem behoben wurde vom großem Interesse. Dadurch können weitere Maßnahmen, wie temporäre Ausblendungen von Produkten frühzeitig angestoßen werden. Für die Prognose von Verarbeitungsendzeitpunkten müssen aktuell chrono-

logisch abgearbeitete Warteschlangen inspiziert werden. Eine Vorhersage-Funktionalität, die den Zeitpunktes wann ein Problem behoben wurde, also eine PIA im Zielsystem als abgeschlossen gilt, ist im gegenwärtigen System nicht installiert.

Das zweite Szenario sind Ausfälle durch massenhaft ungültigen Daten. Zum Bespiel, wenn Produkte mit einem zu früh gesetzten Verkaufsendedatum versehen wurden. Infolgedessen werden alle betroffenen Produkte aus den Online-Shops verschwinden, wodurch es zu wirtschaftlichen Schäden, wie Umsatzeinbußen kommt. Auch besteht die Gefahr, dass Ausfälle dem Ansehen eines Unternehmens schaden und es zu Verlusten bei Marktanteilen kommen kann [Hin06]. In diesem Fall besteht bei diesen Unternehmen das Interesse den Wiederhestellungszeitpunkt, den sog. Mean Time to Recovery (MTTR) herauszufinden [Chi19]. Mean Time to Recovery oder Mean Time to Restore ist die durchschnittliche Zeit, die benötigt wird, um sich von einem Produkt- oder Systemfehler zu erholen. Dies schließt die gesamte Zeit des Ausfalls ein, von dem Zeitpunkt, an dem das System oder Produkt ausfällt, bis zu dem Zeitpunkt, an dem es wieder voll funktionsfähig wird [Kyn10 S.2].

Diese Arbeit stellt eine neuartige Methode vor, mit der Unternehmen eine Visualisierung von Vorhersagen eines Warteschlangenflusses von Produkten durch maschinelles Lernen (ML) zur Informationsbedarfsdeckung einsetzen können. Das Ziel besteht darin den Verarbeitungsstatus der Warteschlangen mit Auslastungsinformationen in einer benutzerfreundlicheren und vorhersehbareren Weise darzustellen und zu visualisieren, um bessere Entscheidungen zu ermöglichen. Mit Hilfe der Action Design Research Methodik bauen wir bei einem der größten globalen Gerätehersteller, der BSH Hausgeräte GmbH (BSH), ein Prototypsystem zur Vorhersage und Visualisierung verteilter Produktinformationswarteschlangen auf und evaluieren seine Nützlichkeit in der Praxis. Motiviert ist diese Arbeit durch Verzögerungen in PIM-Prozessen in zeitkritischen Situationen. Das zu entwickelnde System soll im Allgemeinen bei der Optimierung von PIM-Prozessen beitragen. Anstatt eine separate Lösung zu entwickeln, wird das Warteschlangen-Monitoring-System (WMS), als Plugin zur Integration

in vorhandene Datenanalyse- und Visualisierungsplattformen (Kibana[2]) entwickelt, mit dem Daten aus unterschiedlichen Quellen Daten integriert, in Echtzeit durchsucht, analysiert und visualisiert werden können. Als ganzheitliches Monitoring-System soll die Nutzung von Insellösungen vermiedenen werden, denn die Verwendung von Insellösungen in Unternehmen ist mit erheblichen Risiken verbunden [Bux12].

1.4 Methodisches Vorgehen

Diese Arbeit wendet die Methode der Aktionsforschung (Action Design Research) an. Sie ist für den Aufbau und die Bewertung von IT-Artefakten geeignet [Sei11]. ADR basiert neben der Aktionsforschung auch auf der designwissenschaftlichen Forschung [Sei11]. Die Aktionsforschung stützt sich auf einen iterativen Prozess, der Theorie und Praxis miteinander verknüpft. Die Verbindung beider Bereiche unterstützt die Zusammenarbeit zwischen Praktikern und Forschern [Bas97, Sei11]. Hierbei werden mehrere Zyklen aus Analyse-, Aktions-, und Evaluationsschritten durchlaufen [Wil06]. Diese Methode ist konstruktionsorientiert. Die Problemlösung und das dadurch zu erreichende Verständnis des Problems sind gleichermaßen von Bedeutung [Wil06]. Die ADR Methodik untersucht und löst Probleme, mit denen Organisationen konfrontiert sind. In dem Lösungsprozess werden Änderungen implementiert, diese evaluiert und aus diesen Erkenntnisse gewonnen [Bas97].

Der Forschungsprozess besteht aus mehreren Phasen (siehe Abbildung 2): (1) In der ersten Phase wird auf Grundlage von Theorie und Praxis eine Problemformulierung vorgenommen. (2) In der zweiten Phase folgt der Aufbau, Intervention und Evaluierung des IT-Artefakts in einem spezifischen organisatorischen Kontext (3) In der dritten Phase folgt die Reflexion und Erkenntnisgewinnung, auf Grundlage der in früheren Phasen ergriffenen Maßnahmen. (4) Am Ende des Forschungsprozesses werden die Erkenntnisse durch allgemeine Ergebnisse formalisiert. Zur Verbesserung des IT-Artefaktes können die ersten beiden Phasen nach Bedarf wiederholt werden. Reflexion und Erkenntnisgewinnung können auch

[2]https://github.com/elastic/kibana

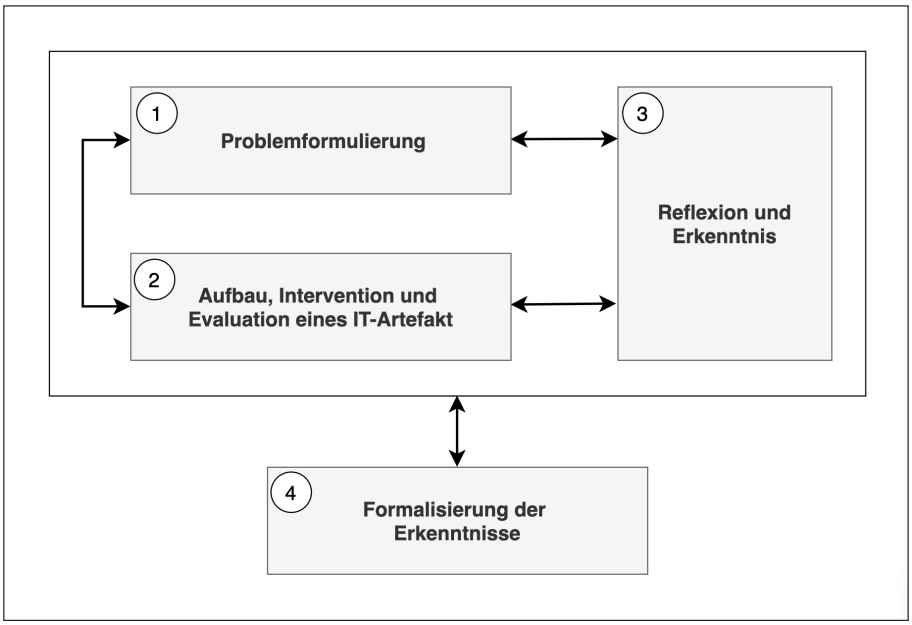

Abbildung 2 Aufbau der Action Research Design Methodik. Eigene Grafik, angl. an [Wil06]

nach jeder der vorhergehenden Phasen erfolgen [Sei11]. Dieser Methodik folgend ist unser Ziel ein neuartiges Visualisierungssystem zum Monitoring von Warteschlangen und deren Integration in ein Datenvisualisierungstools als technologiebasierte Lösung für ein wichtiges und relevantes Geschäftsproblem unter Verwendung neuartiger Lösungen, wie ML und Datenvisualisierung, zu entwickeln.

2 Theoretische Grundlagen

In diesem Kapitel stellen wir die theoretischen Grundlagen der Arbeit vor. Es wird das Grundmodell der Warteschlangentheorie vorstellt und darauf basierend ein Modell der Warteschlangenbildung für die Modellierung des PIA-Prozess in einem System verteilter Produktinformationswarteschlangen vorgeschlagen. Der Prozess der PIA wird durch das Grundmodell der Warteschlangentheorie und Parametern der Kendall-Notation[3] beschrieben, das als Summe von Verarbeitungen mit gemischter Prioritäten ausgeführt wird, und es werden die auf Annahmen der Warteschlangentheorie zugrundeliegenden Leistungsgrößen vorgestellt. Diese Leistungsgrößen von Warteschlangen dienen als Grundlage für die Analyse, Konzeption und Implementierung der Monitoring-Lösung in dieser Arbeit. Zum besseren Verständnis der Ausgangssituation und der daraus resultierenden Anforderungen an das zu entwickelnde System wird außerdem in diesem Kapitel die Systemlandschaft des Unternehmens BSH Hausgeräte GmbH (BSH) und die PIM Prozesse vorgestellt.

2.1 Product Information Management

Product Information Management (PIM) beschreibt sowohl die Prozesse als auch die Systeme, die beim Austausch von Produktinformationen über die gesamte erweiterte Wertschöpfungskette eines Unternehmens, vom Lieferanten bis zum Kunden, beteiligt sind. Unter Produktinformationen werden Produktbeschreibungen und -spezifikationen, aber auch Produkt-Nutzungsinformationen, verstanden [Gri18 S.4]. Mitarbeiter, die mit Produktinformationsaufgaben betraut sind, interagieren mit dem PIM-System, um die Produktinformationen zu integrieren, aufzufinden, zu klassifizieren und sie auf verschiedene Regionen, Vertriebskanäle, Sprachen oder Kundenbedürfnisse zuzuschneiden. Die Bedeutung von PIM-Prozessen und -Systemen nimmt zu, da Unternehmen versuchen, Wettbewerbsvorteile durch Produktdifferenzierungen zu gewinnen [For06 S.3]. Durch diese Produktvielfalt sich vervielfacht die Komplexität, wenn die Anzahl der Produkttypen, Lieferquellen und Vertriebskanäle wächst.

[3] Die Kendall-Notation wurde von David George Kendall zur einheitlichen von Warteschlangensysteme entwickelt [Ken53]

Zum Beispiel vertreibt die BSH über 350 Tausend Produkte in 60 Länder. Hinzu kommen über 300 Tausend Benutzerhandbücher und über 350 Tausend Spezifikationslisten, welche für 102 Sprachen verfügbar sind. Außerdem werden Medien-Daten, wie Bilder (>5 Million) und Energie-Label (>350 Tausend) verwaltet. Außerdem stellen die zunehmende Produktvielfalt mit kürzeren Produktlebenszyklen und komplexeren Produktionsprozessen Unternehmen vor einer großen Herausforderung [For06]. Die Verwaltung von Produktinformationen ist in einem derart komplexen Umfeld von großer Bedeutung.

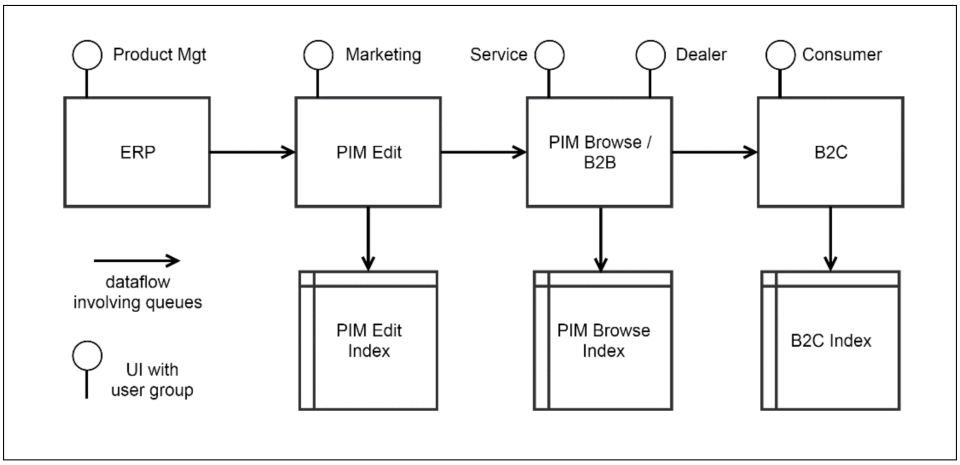

Abbildung 3 Softwarelandschaft PIM-System der BSH [Chi19]

Die BSH besitzt Business-to-Business (B2B) und Business-to-Consumer (B2C) Vertriebskanäle. Produktmanager aktualisieren und veröffentlichen Marketingtexte und Medien über das PIM-System. Diese digitalen Aktualisierungen müssen rechtzeitig online gestellt werden. Die Performance der Aktualisierungen wird bei der BSH durch die Auslastung der zugrunde liegenden Verarbeitungswarteschlangen bestimmt. Betroffen sind hierbei sowohl Alt- als auch moderne Systeme, die über asynchrone Schnittstellen lose gekoppelt sind. Abbildung 3 veranschaulicht die Softwarelandschaft des BSH, wobei der Schwerpunkt auf den relevanten Anwendungen und den Beziehungen zwischen den Datenwarteschlangen liegt. Es ist zu beachten, dass die unterstützenden Datenbanken nicht explizit modelliert sind. Hintergrund ist hierfür ist, dass Verarbeitungsverzögerungen aufgrund von Datenbanküberlastungen nur indirekt durch wachsende Eingabewarteschlangen wahrnehmbar sind. Die Suchin-

dizes wurden hingegen expliziert modelliert, weil sie in der Regel denormalisierte Daten enthalten und jede Änderung von Produktinformationen zu mehrfachen Änderungen der Produkte im Suchindex führt, die von Aktualisierungswarteschlangen verarbeitet werden. Es ist außerdem zu beachten, dass das Software-Landschaftsmodell vereinfacht wurde, um einzelne Datenfluss-Warteschlangen darzustellen, auch wenn jeder dieser Datenflüsse mehreren Systemwarteschlangen entsprechen könnte. Damit Aktualisierungen, die im Enterprise-Resource-Planning (ERP) System angestoßen wurden, im B2C Portal sichtbar werden, müssen alle nachfolgenden Systeme im Warteschlangennetzwerk durchlaufen. Das B2C-Portal benötigt für die länder- und lokalspezifische Sichten zur Vervollständigung, Aggregation und Transformation die Originaldaten vom Vorgängersystem. Daher ist eine Umgehung der Systeme in der Prozesskette nicht möglich. Die meisten Änderungen erfolgen auf Ebene des ERP- und PIM-Edit Systems. Bei jeder Aktualisierung ist das Unternehmen neben dem Aspekt des MTTR, bei Produkteinführungen, auch um den Aspekt des Time-to-Market (TTM), besorgt. Die Zeit, die vergeht, bis eine Produkt oder Dienstleistung marktreif ist und damit eine Markeinführung stattfinden kann, wird als Time-to-Market bezeichnet [Kre20]. Bei einer Produkteinführung werden die Produktdaten im Voraus durch das System geschickt, zur Prüfung und Verifizierung der Daten. Am Startdatum werden sie anhand des beigefügten Verfügbarkeitszeitpunkts sichtbar. Bei der Aktualisierung von Massendaten stellt sich die Situation anders dar. In diesen Fällen kann die Größe der Warteschlangen bis zu mehr als 10.000 Einträge in jeder Warteschlange wachsen. Pro Artikel schwankt die Verarbeitungszeit und kann jeweils bis zu zehn Sekunden dauern. Diese Kumulation der Verzögerungen kann eine Aktualisierungsdauer von nur einer Verarbeitungsinstanz von mehr als zwei Tagen zur Folge haben. Unter diesen Umständen ist eine Vorhersagbarkeit der Gesamtverarbeitungszeit sehr kritisch [Chi19].

2.2 Warteschlangentheorie

Das Problem der Entstehung von Wartezeiten in Geschäftsprozessen lässt sich mit der Warteschlangentheorie darlegen. Grundsätzlich kann man sich eine Warteschlange wie eine Black

Box vorstellen. Aufträge kommen in der Warteschlange an, warten möglicherweise einige Zeit, benötigen Zeit für die Verarbeitung und verlassen dann die Warteschlange wieder (siehe Abbildung 4).

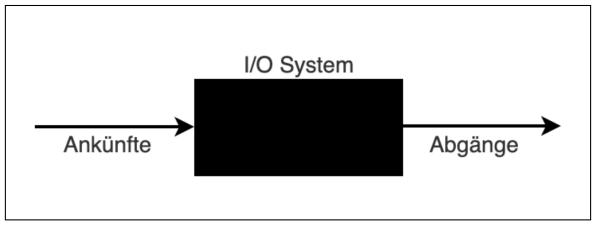

Abbildung 4 Blackbox Modell: Aufträge kommen in der Warteschlange an und verlassen sie wieder [Eigene Darstellung, angelehnt an Plus99].

Die klassische Warteschlangentheorie ist gut entwickelt und wird als grundlegendes Instrument der Leistungsbewertung in vielen Bereichen wie Computer und Telekommunikation, Fertigung, Dienstleistung sowie in Transportsystemen eingesetzt [Tia06]. In der Theorie werden Warteschlangen, sogenannte Bedienungssysteme in einem einfachen Grundmodell beschrieben. Dieses Modell hat einen stochastischen Ankunftsstrom, über das Elemente in das System eintreten können [Ken53]. Dargestellt wird er durch ein Pfeil im Modell. Außerdem beinhaltet das Modell einen den sogenannten Bediener (engl. Server). Der Bediener kann über eine oder mehrere Verarbeitungsinstanzen verfügen. Diese Instanzen arbeiten parallel. Ein weiterer Parameter dieses Modells ist der Warteraum. Der Warteraum symbolisiert den Bereich in dem Elemente verweilen, bis sie vom Bediener bearbeitet werden. Der Warteraum wird in dem Modell als eine Folge von Rechtecken dargestellt (siehe Abbildung 5). In dem Beispiel der Bedienung von Kunden, können diese einzeln an beliebigen Zeitpunkten an einer Bedienungsinstanz eintreffen. Wenn eine Instanz nach dem Abschluss der Bedienung eines Kunden frei wird, können weitere Kunden je Bedienungsinstanz individuell bedient werden. Ist keine Instanz frei, müssen nachfolgend ankommende Kunden sich in die Warteschlange einreihen [Hans12].

Überträgt man das Grundmodell der Warteschlangentheorie auf die Verarbeitung von Produktinformationswarteschlangen verteilter Umgebungen, wird durch die verteile Umgebung

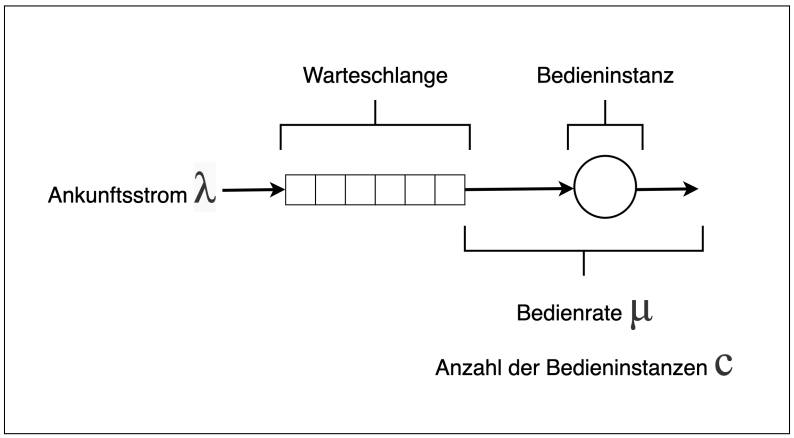

Abbildung 5 Grundmodell der Warteschlangentheorie [Eigene Darstellung, angelehnt an Hans12]

der Systemlandschaft die Komplexität des Grundmodells um den Faktor drei erhöht. In Abbildung 6 wurde das Warteschlangensystem-Modell des Untersuchungsgegenstands mit Parameter der Kendall-Notation erweitert. Bei diesem Anwendungsfall handelt es sich um ein System bestehend aus drei verknüpften Systemen. Man spricht hierbei von einem komplexen Warteschlangennetzwerk [Mod08]. Ein Warteschlangennetzwerk funktioniert im Grunde wie eine Warteschlange, nur dass in diesem Fall mehrere Warteschlangensysteme miteinander verknüpft sind [Weis09].

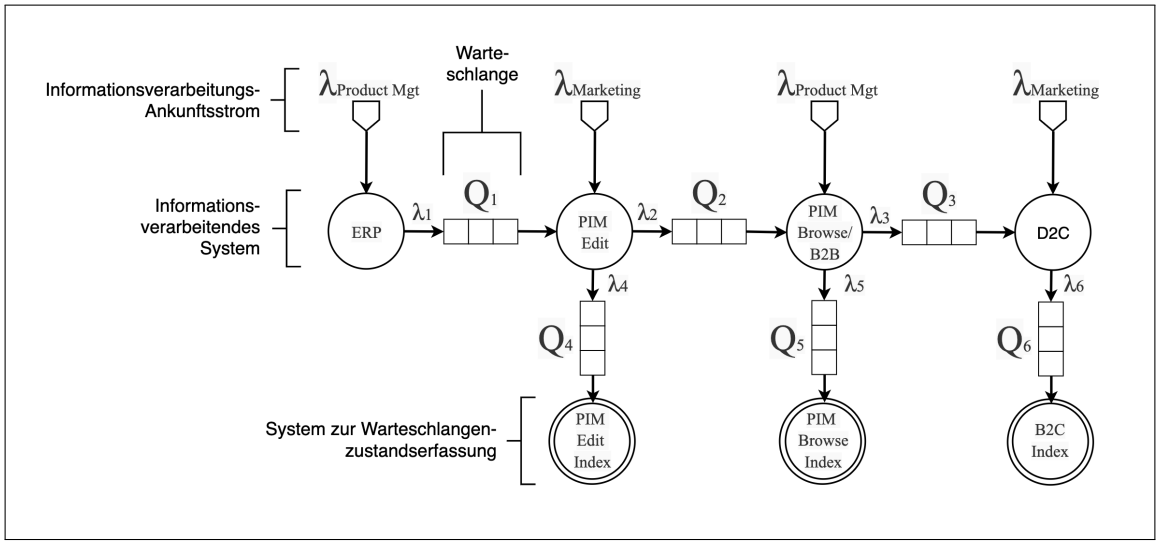

Abbildung 6 Modell des Warteschlangensystems der BSH Systemlandschaft [Eigene Darstellung]

Dabei entsprechen die Bedieninstanzen (Bedienstelle bzw. Server), dem Modell der Warteschlangentheorie den Anwendungssytemen der Systemlandschaft der betrachteten Geschäftsprozesse der BSH Hausgeräte GmbH. In dem Kontext existieren 6 Warteschlagen (Q). Einen besonderen Fokus legen wir auf die Warteschlangen Q1 bis Q3. Sie stellen die Warteschlangen nach dem Informationsverarbeitungsankunftsstrom (IVA) aus den jeweiligen informationsverarbeitenden Systemen (IV-System) in nachfolgende IV-Systeme dar. Jede der IV-Systeme erhalten durch eine Nutzergruppe (außer beim System PIM Browse/ B2B, bei diesem sind es zwei Nutzergruppen (Mehrkanaliges-Bediensystem[4])) angestoßenen Ankunftsstrom an Informationen (In dieser Arbeit modellhaft als Elemente bezeichnet). Diese in einem System eintreffenden Elemente werden von den IV-Systemen über eine Warteschlage verarbeitet. Nach der Verarbeitung eines Elements wird der Stand der Verarbeitung in einem System zur Warteschlangenzustandserfassung (WZE) persistiert, in dem der Stand als Liste der Elemente deren Verarbeitung noch aussteht festgehalten wird. Neben den zur Verarbeitung ausstehenden Elementen, werden noch die der Warteschlange, das System aus dem der Informations-Verarbeitungsstrom stammt und die Art der Elemente in der Warteschlange, erfasst. Nach der Verarbeitung eines Elements von einem IV-System, bewegt es sich über eine weitere Warteschlange zur Verarbeitung eines nachfolgenden IV-Systems.

Die Warteschlangen (Q1 bis Q3) können mit Hilfe mathematischer Verfahren nach folgenden Aspekten analysiert werden. Bei den genannten Warteschlangen ist speziell der Aspekt der Analyse interessant. Mittels der zuvor genannten Annahmen liefert die Warteschlangentheorie Aussagen über folgende Leistungsgrößen:

Allgemeine Ankunftsrate: Die Ankunftsrate ist der Kehrwert der mittleren Zwischenankunftszeit. Sie gibt an wieviele Aufträge im Durchschnitt pro Zeiteinheit in ein System eintreten

[4]Bediensystem mit beliebig vielen Bedienstellen [Hel07].

[Hel20 S.25 ff.]. Es ist der Erwartungswert der Anzahl der Ankünfte pro Zeiteinheit.

$$\lambda = \frac{\text{Anzahl der Ankünfte}}{Zeiteinheit} \tag{i}$$

Bedienrate: Die Bedienrate gibt an, wieviele Elemente im Durchschnitt pro Zeiteinheit von dem Bedienungssystem abgefertigt werden können. Sind mehrere parallele und gleichartige Bedienungsinstanzen vorhanden, erhöht sich die Bedienrate entsprechend der Anzahl der Instanzen [Hans12].

Auslastung: Die Auslastung gibt an wie sehr das System ausgelastet ist [Hans12]. Wenn die Server-Auslastung größer als 1 ist, wächst die Warteschlange unendlich lang [Plus99].

$$\rho = \frac{\lambda}{\mu} \tag{ii}$$

Bedienzeit: Die Bedienzeit gibt an wie lange eine Bedieninstanz für die Verarbeitung eines Auftragselements benötigt [Zim14]. Ist die mittlere Bedienzeit für ein Element größer als die mittlere Ankunftszeit führt es dazu, dass die Warteschlange immer länger wird [Reu16].

Durchsatz: Auch als Abgangsrate bezeichnet gibt der Durchsatz die mittlere Anzahl von Anfragen, die pro Zeiteinheit bedient werden, an [Zim14]. Wenn ein System stabil ist, ist die Abgangsrate gleich der Ankunftsrate = λ [Dom94]. Der effektive Durchsatz, also der tatsächlich abgearbeiteten Aufträge kann durch Abzählung ermittelt werden.

$$\chi = \sum_{n+1}^{\infty} \mu_n \rho_n \tag{iii}$$

Warteschlangentiefe: Diese Eigenschaft gibt an wieviele Aufträge sich zur Zeit t im Bediensystem aufhalten [Hans12].

$$N_t > 0 \tag{iv}$$

Die *durchschnittliche Warteschlangentiefe* gibt an wieviele Elemente sich im Durchschnitt

im System befindet [Mod08]. Sie kann mithilfe Littles Gesetz berechnet werden [Mod08, Chh08 S.81 ff.]:

$$\overline{N} = \frac{\lambda}{\mu - \lambda} - \frac{\lambda}{\mu} = N - \rho \tag{v}$$

2.3 Plugins

Software-Systeme waren in Zeiten der imperativen Programmierung abgeschlossen. Erweiterungen an den Systemen waren mit großen Aufwand verbundene, fehlerträchtige Wartungsarbeiten. Außerdem waren Programmerweiterungen nur denen vorenthalten, die Zugang zum Quellcode hatten. Plugin-Architekturen bieten jedoch Entwicklern von Drittanbieter-Software die Möglichkeit, ein Software-System an dafür vorgesehenen Punkten um zusätzliche Funktionalität zu erweitern, ohne es zu modifizieren [Sun19 S.104, Lud10].

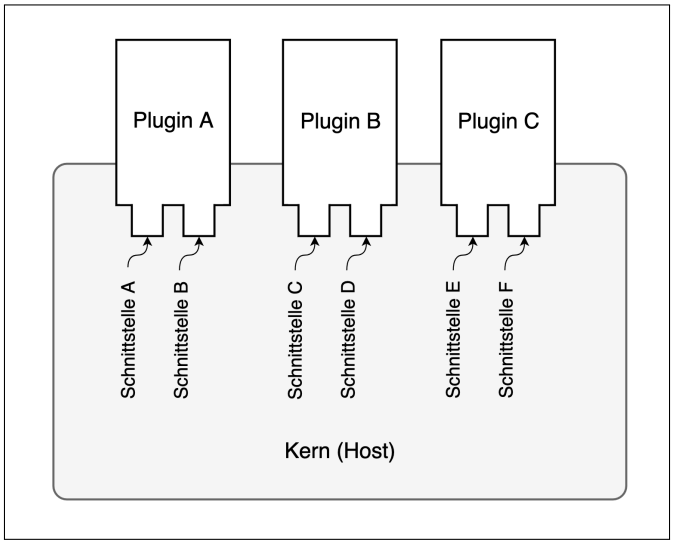

Abbildung 7 Schema einer Plugin-Architektur [Eigene Darstellung, angelehnt an Lud10]

Es handelt sich hierbei um ein zusätzliches Programm, das als Zusatz-Modul in der vorhandenen Software-Architektur besteht. Ein System mit der Plugin-Architektur setzt das Offen-geschlossene-Prinzip auf der Ebene von Anwendungen um. Programme, deren Aufbau diesem Architekturmuster folgt, bestehen grundsätzlich aus den folgenden zwei Teilen. Zum einen aus dem Kern (Host), der durch sogenannte Plugins um neue Funktionen erwei-

tert werden kann. Auf den sog. Erweiterungspunkten, welche als spezielle Schnittstellen vom Host-System definiert wurden, kann ein Plugin Bezug nehmen (siehe Abbildung 7). Ein Plugin muss damit es im Kontext des Host ausgeführt werden kann, gemäß den vom Host vorgegebenen technischen Konventionen realisiert werden. Neben der entsprechenden Integration des Plugin-Codes in die Host-System-Architektur, muss das Plugin beim Host registriert sein. Die Identifizierung des Plugins findet beim Start des Hosts statt, wobei es sofort oder nach Bedarf geladen wird. Auch ein Plugin kann selbst als Host fungieren, d.h. sie definieren ebenfalls Erweiterungspunkte für weitere Plugins [Lud10].

3 Anforderungsanalyse

Dieser Abschnitt beschäftigt sich mit den Anforderungen an das WMS, dem System für die Visualisierung für Vorhersagedaten und Warteschlangen-Auslastungs-Metriken im Kontext von PIA-Warteschlangen. Wir identifizieren und skizzieren die ersten Rahmenbedingungen für die Entwicklung des WMS. Zunächst beschreiben wir die Fähigkeiten des Software-Systems, die ein Anwender erwartet, um mit Hilfe dieser Software ein fachliches Problem zu lösen, die sogenannten funktionalen Anforderungen. Anschließend folgen die nicht-funktionalen Anforderungen, also den Eigenschaften, die sich aus den funktionalen Anforderungen ergeben und im Architekturentwurf berücksichtig werden müssen (Bal09 S.489). In Tabelle 1 sind alle funktionalen und nich-funktionalen Anforderungen des WMS kategorisiert aufgelistet.

3.1 Funktionale Anforderungen

Balzert definiert Funktionale Anforderungen als „die Fähigkeiten eines Systems, die ein Anwender erwartet, um mit Hilfe des Systems ein fachliches Problem zu lösen“ [Bal09 S. 489]. Aus den Geschäftsprozessen lassen sich die Anforderungen ableiten [Bal09 S. 489].

3.1.1 Use-Cases

In Form von Anwendungsfällen (Use Cases) können die funktionalen Anforderungen definiert werden. Ein konkreter, fachlich in sich geschlossener Teilvorgang wird als Anwendungsfall bezeichnet. Dabei werden von bestimmte Grundfunktionalitäten des zu entwickelnden Software-Systems visuell dargestellt [Bal09 S.489]. In Abbildung 8 werden die funktionalen Anforderungen in Form eines Use Case Diagramms visuell zusammengefasst.

Eine Benutzerschnittstelle, das sogenannte User Interface (UI) muss dem Anwender zur Verfügung stehen, über das dem Nutzer Informationen zu den Produkten im PIA-Prozess geliefert wird. Im Fokus stehen in der Visualisierung die Darstellung von Eintritts- und vorhergesagte Austrittszeitpunkte aus den PIA-Warteschlangen. Aufgrund der großen Anzahl der in den Warteschlangen befindlichen Produkte, muss dem Anwender ein Suchfeld zur Verfü-

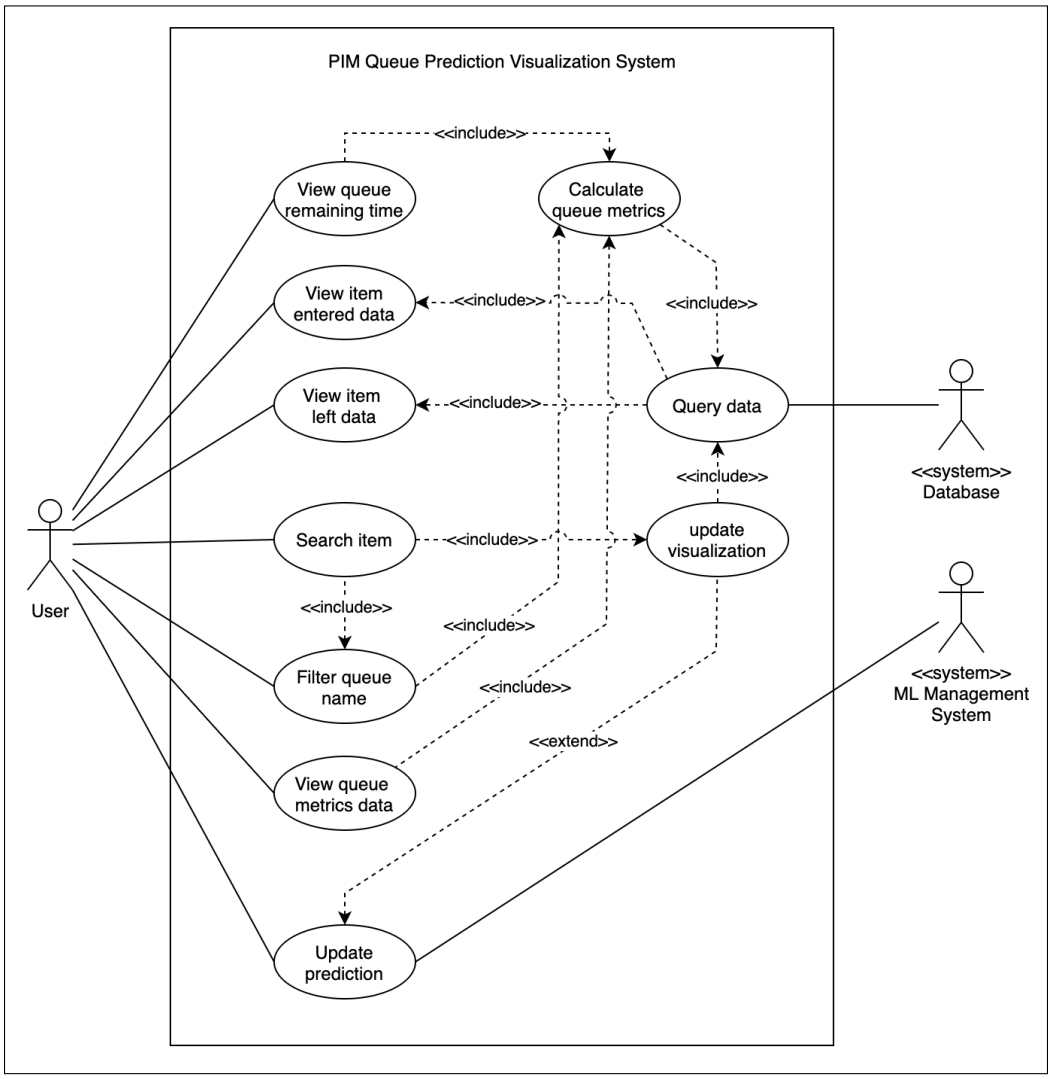

Abbildung 8 UML Use-Case Diagramm [Eigene Darstellung]

gung gestellt werden, über das er über ihre Bezeichnungen, Produkte filtern und gesondert in der Visualisierung darstellen kann. Produkte können je Prozessstufe in mehreren Warteschlangen verweilen, daher ist eine Filterfunktion für die diversen Warteschlangen-Typen zur Verfügung zu stellen.

Als Akteure werden Rollen bezeichnet, die mit Geschäftsprozessen interagieren. Bei den Rollen können Menschen oder Systeme, insbesondere Computer-Systeme, sein, die als externe Beteiligte kommunizieren und Daten austauschen. Nach der UML Notation wer-

den Akteure durch Strichmännchen dargestellt [Bal09 S.227]. Im WMS sind neben dem menschlichen Akteur, dem Anwender, auch zwei System-Akteure in der Interaktion mit dem Monitoring-System beteiligt (siehe Abbildung 8). Zum einen das Datenbank-System, in dem Daten persistiert werden, und diese als Datenquelle für das WMS dient.

3.2 Nicht-funktionale Anforderungen

Im folgenden werden wir die wesentlichen Aspekte nicht-funktionaler Anforderungen vorstellen. Balzert definiert nicht-funktionale Anforderungen als „grundlegende Eigenschaften eines Systems, die im Architekturentwurf berücksichtigt werden müssen“ [Bal09 S.489]:

Benutzbarkeit: Die Fähigkeit einer Software von einem Benutzer verstanden und unter den festgelegten Bedingungen bedient zu werden (Bal09 S.469).

Effizienz: Die Fähigkeit der Bereitstellung einer angemessenen Leistung unter den festgelegten Bedingungen bereitzustellen (Bal09 S. 469).

Zuverlässigkeit: Die Fähigkeit einer Software eine bestimmte Leistung unter festgelegten Bedingungen zu gewährleisten (S.468).

Portabilität: Die Fähigkeit einer Software eine Übertragung von einer Umgebung in eine andere zu ermöglichen. Bei der Umgebung kann es sich um eine Hardware- oder Software-Umgebung handeln (Balz09 S. 470).

Wartbarkeit: Die Fähigkeit einer Software geändert werden zu können. Änderungen können Korrekturen, Optimierungen oder Anpassungen der Anwendung an veränderte Umgebungen, Anforderungs- oder Spezifikationsänderungen einschließen (Bal09 S.470).

Funktionalität: Die Fähigkeit von einer Software durch die Bereitstellung von Funktionen

unter spezifizierten Bedingungen festgelegte Bedürfnisse zu befriedigen (Bal09 S.468).

Identifikator	Kurzbeschreibung
	Produktsuche und -filterung
R1.1	Suche nach Produkten
R1.2	Filterung von Produkten nach Warteschlangen
R1.3	Filterung von Produkten nach Systemstufe
	Vergangenheits- und Vorhersagezeitpunkte
R2.1	Anzeige von Eintrittszeitpunkte in Warteschlagen von einem Produkt
R2.2	Anzeige von Austrittszeitpunkte aus Warteschlangen von einem Produkt
R2.3	Anzeige von verbleibender Zeit bis zum Austrittszeitpunkt eines Produktes
	Warteschlangenmetriken
R3.1	Anzeige Warteschlangen-Tiefe
R3.2	Anzeige Warteschlangen-Durchsatz
R3.3	Anzeige Warteschlangen-Auslastung
	Vorhersage-Aktualisierung
R4.1	Aktualisierung von Vorhersage-Daten
R4.2	Anliege des Zeitpunkts der letzten Vorhersage-Aktualisierung
NF1	Zuverlässigkeit
NF2	Benutzbarkeit
NF3	Effizienz
NF4	Wartbarkeit
NF5	Portabilität

Tabelle 1 Zusammenfassung der System-Anforderungen

4 Entwurf und Implementierung

In diesem Kapitel wird im Detail darauf eingegangen, wie die Anwendung in dieser Arbeit konzipiert und prototypisch implementiert wurde.

4.1 Entwurf

Beim Entwurf berücksichtigen wir die zukünftige Infrastruktur der BSH, weil in dieser Umgebung das System später betrieben wird. In Folge dessen gibt sie Richtlinien für Technologieentscheidungen für die zu entwickelnde Anwendung [Bal09 S.488]. Die Applikation besteht aus einem Frontend, das die grafische Benutzerschnittstelle bildet; und der Datenerhaltung. Die Datenerhaltung wurde mit Elasticsearch[5] umgesetzt. In diesem System spielt Elasticsearch eine zentrale Rolle, weil sie bei der BSH als Datenbank für die PIA-Daten dient und dem zu entwickelnden WMS als Datenquelle dient. Es ist eine Such- und Analyse-Engine. Elasticsearch basiert auf Apache Lucene[6] Technologie. Diese Technologie unterscheidet sich grundlegend von traditionellen relationalen Datenbanken. Sie stellt Speicher-, Such- und Analysefunktionen bereit. Die Volltextsuche von Elasticsearch ermöglicht das Durchsuchen aller Begriffe in der Datenbank verfügbaren Dokumente. Darüber hinaus verfügt Elasticsearch über eine sehr umfangreiche REST-API (Representational State Transfer), die mit dem Hypertext Transfer Protocol (HTTP) arbeitet. Die REST-API ist sehr gut dokumentiert und recht umfassend und macht alle Funktionen über HTTP verfügbar. Somit lässt sich Elasticsearch sehr einfach in jede Anwendung mit REST Schnittstelle, integrieren. Elasticsearch ist horizontal skalierbar und bietet Datenabfragen in nahezu Echtzeit an. Elasticsearch ermöglicht Tausende bis Hunderttausende von Dokumenten pro Sekunde zu indexieren und macht sie fast sofort für die Suche verfügbar.

Das Frontend verarbeitet das, was in der grafischen Benutzeroberfläche (GUI) in einem Web-Browser angezeigt wird. Der Implementierungsarbeit wurde die Gestaltung der GUI

[5]https://github.com/elastic/elasticsearch

[6]https://lucene.apache.org

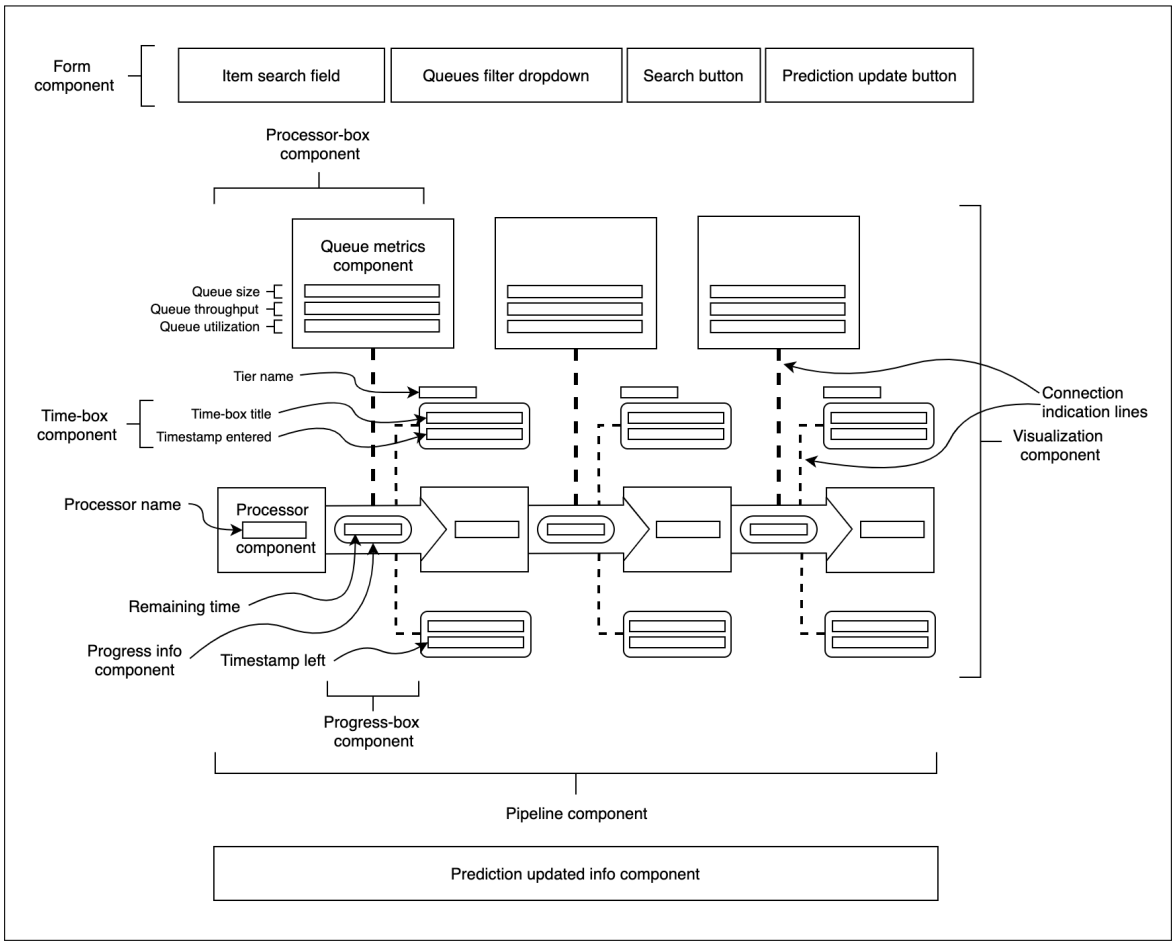

Abbildung 9 React App Komponenten Mockup [Eigene Darstellung]

vorausgesetzt. Abbildung 9 zeigt das finale Mockup der grafischen Strukturierung des WMS als Resultat des grafischen Entwurfsprozess. Die Visualisierung wird in einer Prozesslandkarten-Abstraktion entworfen. Eine Prozesslandkarte zeigt visuell die Schritte einer Arbeitsaktivität an, die an der Durchführung der einzelnen Schritte beteiligt sind. Damit soll ein genauer und schneller Überblick über den PIA-Status eines Produktes und Informationen zu Metriken der jeweiligen Warteschlangen zwischen den Anwendungssystemen in der Prozesskette ermöglicht werden.

4.2 Prototypische Umsetzung des Entwurfs

Nach dem Entwurf wurde anschließend ein interaktiver Prototyp des WMS entwickelt. Abbildung 10 zeigt das erste prototypisch implementierte WMS System.

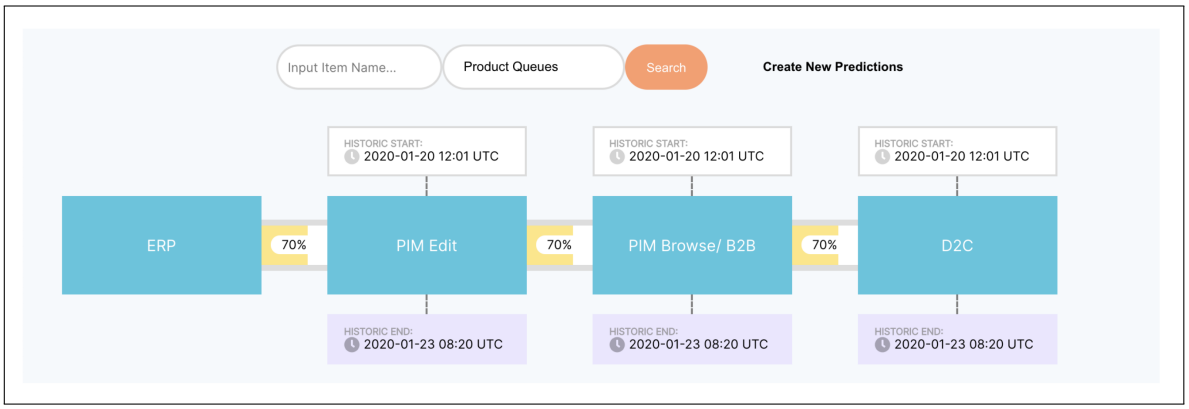

Abbildung 10 React App - Erster Rapid-Prototype [Eigene Darstellung]

Das Frontend besteht aus einer Reihe von React[7]-Komponenten. React ist eine JavaScript-Bibliothek zur Erstellung von Benutzeroberflächen. Sie ermöglicht eine einfache Entwicklung von zustandsgesteuerten Anwendungen. React läuft clientseitig. Eine React-Anwendung kann mehrere Komponenten enthalten, die mehrfach verwendet werden können. Mit React können Single-Page-Applikationen (SPA) entwickelt werden. Im Vergleich zu herkömmlichen Web-Applikationen sind SPA's performanter.

Alle React-Komponenten wurden von Grund auf für dieses Projekt entwickelt. Die für diese Anwendung implementierten React-Komponenten können aus der Abbildung 11 entnommen werden. Die Übersicht zeigt die Struktur der wesentlichen Komponenten innerhalb der Anwendung. Die Entitäten in dieser Abbildung stellen die verschiedenen GUI Komponenten in der Monitoring-Lösung dar. Die Pfeile zeigen an, welche Komponenten durch andere definiert sind, z.B. wird die Formular-Komponente und die Visualisierungskomponente durch die App-Komponente definiert. Bestandteile der Formularkomponente sind ein Suchfeld, Dropdown-Element, Such-Button und der Vorhersagen-Aktualisierungsbutton. Die Vi-

[7]https://reactjs.org

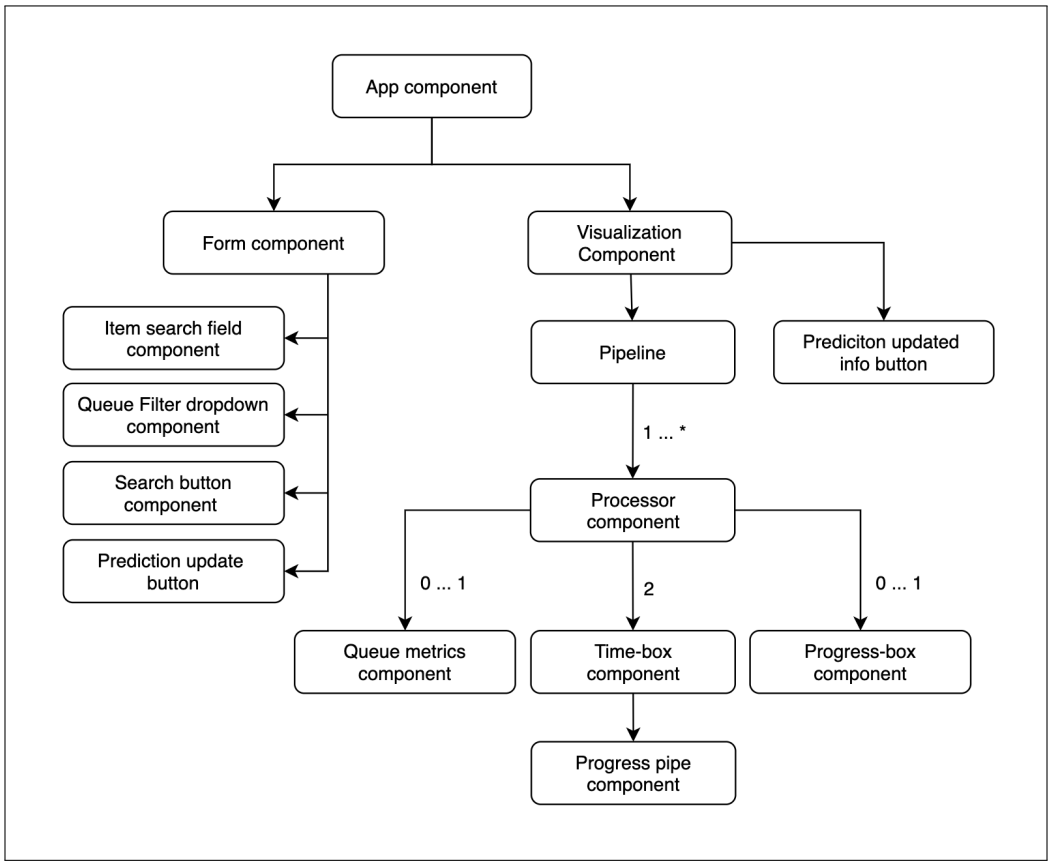

Abbildung 11 Struktur der React-Komponenten [Eigene Darstellung]

sualisierung definiert die Prozessoren-Pipeline und die Informationskomponente zur Anzeige der letzten Vorhersagen-Aktualisierung. Die Pipeline-Komponente definiert eine beliebige Anzahl an Prozessoren, die jeweils eine oder gar keine Metrik-Komponente, mindestens zwei Zeitanzeige-Komponenten und eine oder keine Fortschrittskomponente. Die Fortschrittskomponente definiert jeweils eine Fortschrittsbalken-Komponente. Außerdem kam bei der Entwicklung des Prototyp-Systems Typescript[8] zum Einsatz. Typescript ist eine Art Wrapper um JavaScript oder eine Erweiterung davon. Jeder gültige JavaScript-Code auch gültiger TypeScript-Code. TypeScript fügt jedoch Dinge über JavaScript hinaus hinzu, wobei das Wichtigste die Datentypen sind, wie der Name andeutet. Bei JavaScript und seiner locker getippten Natur sind Fehler leicht zu machen, z.B. die Übergabe einer Zeichenfolge, bei der ein numerischer Wert erwartet wird. Mit TypeScript werden solche Fehler durch verschiede-

[8]https://www.typescriptlang.org

ne Werkzeuge schnell und einfach erkannt [Typ20, Zam20 (S.87-88)]. Mit CSS (Cascading Style Sheets) wurden die HTML-Elemente in den implementierten React-Komponenten gestalterisch beschrieben [css20].

Zur Abbildung der PIA-Warteschlangendaten wurde zum Testen der Anwendung Testdaten, die lokal über ein Test-Elasticsearch-index erreichbar waren, verwendet. Es wurde die Elasticsearch-API (Application Programming Interface) verwendet um Datenabfragen (Query) an Elasticsearch zu realisieren. Quelltext 1 zeigt ein Beispiel zur Nutzung der Elasticsearch API mit einer einfachen Datenabfrage. Dabei wird wird ein JSON[9]-Objekt mit der Elasticsearch-spezifische Abfragesprache DSL (Domain Specific Language) -Query an den Elasticsearch Client gesendet, welcher mit den gewünschten Daten aus dem Elasticsearch Index antwortet, ebenfalls im JSON-Format. Query-DSL ist eine auf JSON-basierende Sprache zur Definition von Abfragen [dsl20].

[9]https://www.json.org/json-de.html

5 Kibana Integration

Dieser Abschnitt ist dem Prozess der Integration in die Datenanalyse- und -visualisierungsplattform Kibana gewidmet. Zunächst stellen wir Kibana vor. Anschließend wird die Architektur der Plattform analysiert und darauf basierend Integrationsentscheidungen getroffen. Die jeweiligen Integrationsoptionen werden vorgestellt und beginnend von der geringsten Integrationskomplexität eine Integration des zuvor entwickelten Visualisierungssystem vorgenommen. Abschließend werden die erprobten Integrationen hinsichtlich ihrer Grenzen und Flexibilität evaluiert.

5.1 Kibana Einführung

Kibana ist eine Open-Source-Analyse- und Visualisierungsplattform. (Funktioniert mit Elasticsearch als Datenerhaltung-Grundlage). Es kann zum Suchen, Anzeigen und Interagieren mit allen in Elasticsearch gespeicherten strukturierten oder unstrukturierten Daten verwendet werden. Es ermöglicht eine nahtlose erweiterte Datenanalyse und ermöglicht die Visualisierung von Daten in einer Vielzahl von Histogrammen, Diagrammen, Grafiken, Tabellen und Karten. Elasticsearch fungiert als Datenbank, für die Datenerhaltung von Kibana. Abbildung 12 zeigt die Benutzeroberfläche der Kibana-Plattform. Einige der wichtigsten Funktionen von Kibana sind die folgenden [KibanaFeatures20]:

Visualisierung: Kibana bietet Nutzern Daten auf vielseitige Art und weise zu Visualisieren. Einige der Angebotenen Visualisierungen sind Balkendiagramme, Kreisdiagramme, Liniengraphen, Heatmaps etc.

Dashboard: Nach der Erstellung von Visualisierungen, können diese in benutzerdefinierte Dashboards integriert werden. Das Ziel eines Dashboards ist übersichtlich komplexe Daten, in der Regel auf genau einer Seite, aufzubereiten [Cha20].

Dev Tools: Über das Entwicklerwerkzeug „Dev Tools“ können Nutzer in Kibana Datenabfra-

gen und Modifizierungen an Elasticsearch Indices vornehmen. Mit Hilfe dieses Werkzeugs können Daten hinzugefügt, aktualisiert oder gelöscht werden.

Reports: Alle Daten aus Dashboards und Visualisierungen können als Berichte (Reports) exportiert werden.

Filter und *Suchanfragen*: Filter und Suchabfragen können zur Extraktion von Details aus Dashboards oder Visualisierungen verwendet werden.

Plugins: Nutzer können Plugins von Drittanbietern in Kibana hinzufügen, um beispielsweise Kibana um neue Visualisierungen und andere UI-Module zu erweitern.

Timelion: Bei Timelion handelt es sich um ein spezielles Visualisierungstool für zeitbasierte Datenanalyse.

Canvas: Kibana Canvas ist ein mächtiges Datenvisualisierungs- und Präsentationswerkzeug in Kibana. Mit Canvas können Daten von Elasticsearch abgerufen und grafisch visualisiert werden [Can20].

Es gibt zahlreiche Vorteile, die Kibana bietet, wie die einfache Bedienung und standardmäßig integrierten Werkzeuge zur Erstellung von Visualisierungen von Daten, und Analyse von Daten. Jedoch gibt es auch Nachteile von Kibana. Das Hinzufügen von Plugins zu Kibana kann sehr mühsam sein, wenn Versionsunterschiede existieren. Auch kann es zu Problemen mit Plugins von Drittanbieter kommen, wenn auf eine neue Version aktualisiert wird.

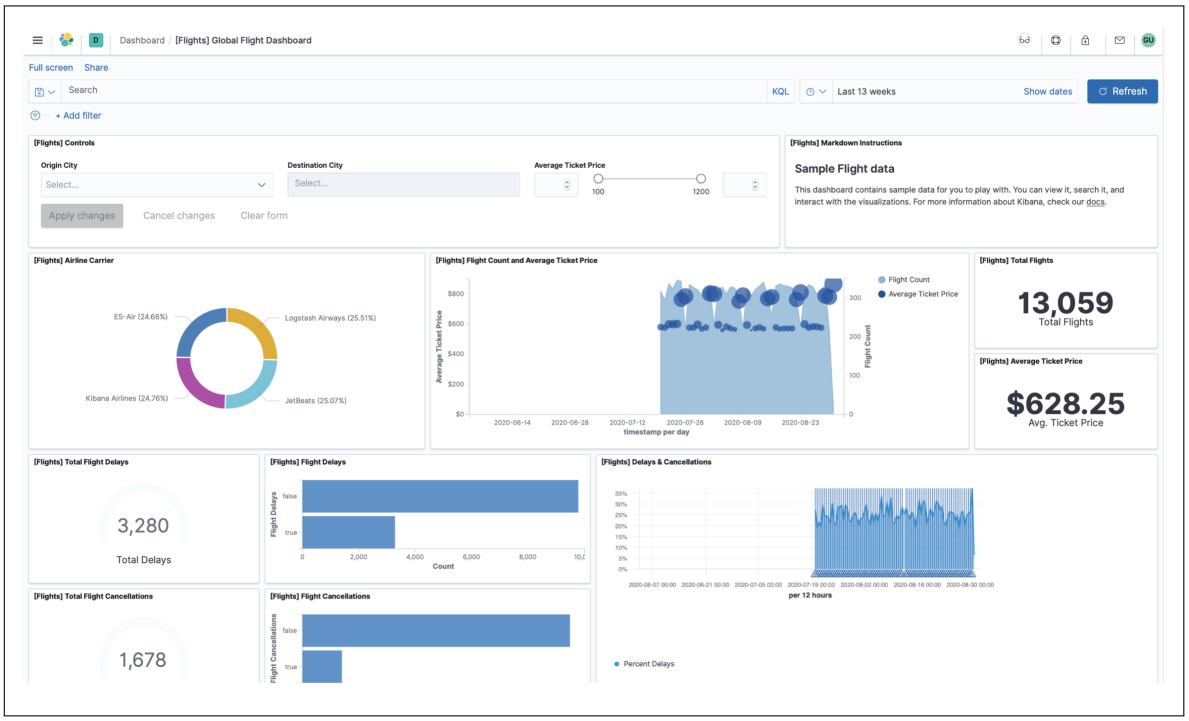

Abbildung 12 Demo Dashboard Kibana [KiDemo]

5.2 Analyse der Kibana Architektur

Dieser Abschnitt ist der System-Architektur von Kibana gewidmet. Die Architektur lässt sich in zwei Kernmodule unterteilen. Erstens, das UI-Modul, das die grafische Benutzeroberfläche definiert, mit der Benutzer interagieren. Zweitens, das Server-Modul, das die Daten aus den Elasticsearch-Clustern dem ersten Modul über eine interne API zur Verfügung stellt.

Die Funktionalität in Kibana wird durch Plugins implementiert. Diese Plugins enthalten die Geschäftslogik und kommunizieren mit dem UI-Modul. Dieser modulare Ansatz schafft eine lockerer gekoppelte Codebasis. Es erleichtert Plugin-Entwickler von Drittanbietern, Funktionalität in Kibana zu implementieren. Eine vereinfachte Übersicht über die Architektur auf hoher Ebene ist in Abbildung 13 dargestellt. Das Server-Modul ist mit Elasticsearch verbunden und bietet eine interne API, die vom UI-Modul genutzt wird. Wenn der Benutzer über die graphische Benutzeroberfläche auf Kibana zugreift, lädt das UI-Modul alle Kern-Plugins, die die Kern-Funktionalitäten von Kibana umfassen, und muss daher immer

enthalten sein. Alle Kibana-Module hängen von einer oder mehreren externen Abhängigkeiten ab, die zusammen das letzte Modul im Diagramm bilden. Diese Pakete von Drittanbietern werden über den Paketmanager Yarn[10] verwaltet.

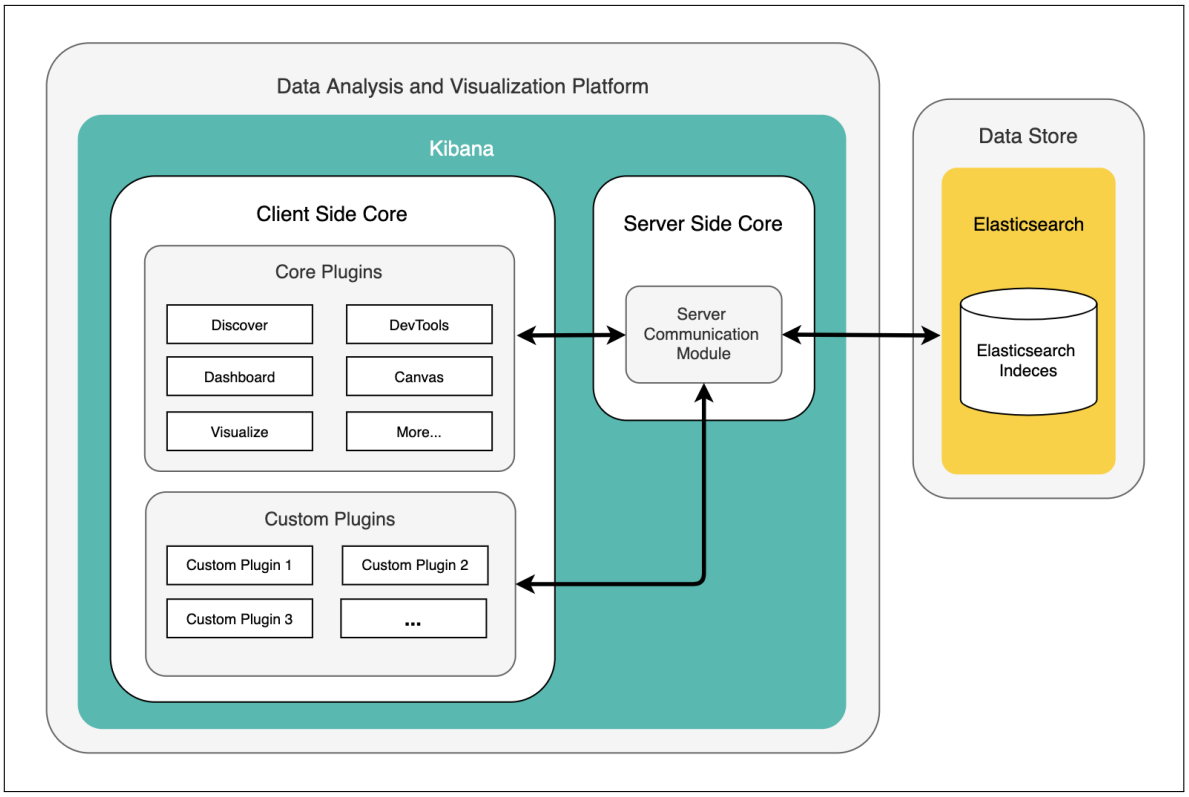

Abbildung 13 High-level Modul-Abhängigkeitsdiagramm [Eigene Darstellung]

Der Kibana-Kern, der sog. Core, bildet die Grundlage von Kibana. Sie ist das Root-System, das Kibana bootet, die Konfigurationen validiert, Plugins lädt und stellt die primitiven API's, die zur Entwicklung von Plugins in Kibana benötigt werden, bereit. Der Kern besteht aus einer Reihe von Diensten, von denen jeder API's für Plugins an verschiedenen Punkten des Lebenszyklus des Systems, bereitstellt. Die Kern-Dienste sind immer gegenwärtig und können nicht deaktiviert werden. Alle essentiellen Elemente zur Entwicklung eines Plugin werden von den Kern-Diensten bereitgestellt. Die zugrunde liegenden Implementierungsdetails der Kerndienste sind gekapselt und vor Plugins verborgen. Dies bringt eine Reihe

[10]https://yarnpkg.com

von Vorteilen mit sich. Der größte Vorteil liegt in der Möglichkeit einer hohen Testabdeckung, indem Plugins nur der Zugriff auf speziell für den Zweck entwickelten Eigenschaften gestattet wird. Somit kann sichergestellt werden, dass die API's des Kerns, aufgrund ihrer kleinen API-Oberfläche, sehr robust sind.

Nahezu jede Funktion, die in Kibana verwendet werden kann, ist in einem Plugin eingebettet. Im Allgemeinen ist ein Plugin eine Gruppe von Funktionen, die ein- oder ausgeschaltet werden können, um Funktionen oder Anwendungen in Kibana zu integrieren. Plugins haben Zugriff auf alle API's, die von Kerndiensten zur Verfügung gestellt werden. Plugin können Abhängigkeiten zu anderen Plugins haben. Eine beliebige Erweiterung von Funktionen ist in Kibana möglich, ohne die API-Oberfläche des Kibana Kerns zu modifizieren. Plugins in Kibana können Daten in Elasticsearch abfragen, erstellen und generische Dienste für andere Plugins bereitstellen.

Es werden bei Kibana ein serverseitiger und clientseitiger Kern unterschieden. Die serverseitigen Dienste bieten typische Backend-Funktionalitäten, wie HTTP-Routen, während die clientseitigen Dienste Frontend-Funktionalitäten bieten. Beide Kerne folgen einem ähnlichen Entwurf. Der Vorteil dieses Designs ist, dass Entwickler ein einziges Muster für die Erstellung von Plugins lernen müssen, das zwischen beiden Umgebungen portabel ist [Dov16].

5.3 Übersicht der Integrationsoptionen

Benutzerdefinierte Anpassungen in Kibana lassen sich in unterschiedlicher Form umsetzen. Nach intensiver Recherche über Erweiterungsmöglichkeiten in Kibana können folgende Möglichkeiten in Erwägung gezogen werden (siehe Abbildung 14).

Die Integration der jeweiligen benutzerspezifischen Modifikationen unterscheiden sich in der Komplexität der Umsetzung und der funktionalen Flexibilität. Am einfachsten lassen sich individuelle Anpassungen mithilfe des (1) Kibana Canvas Plugins umsetzen. Dieses Plugin ist

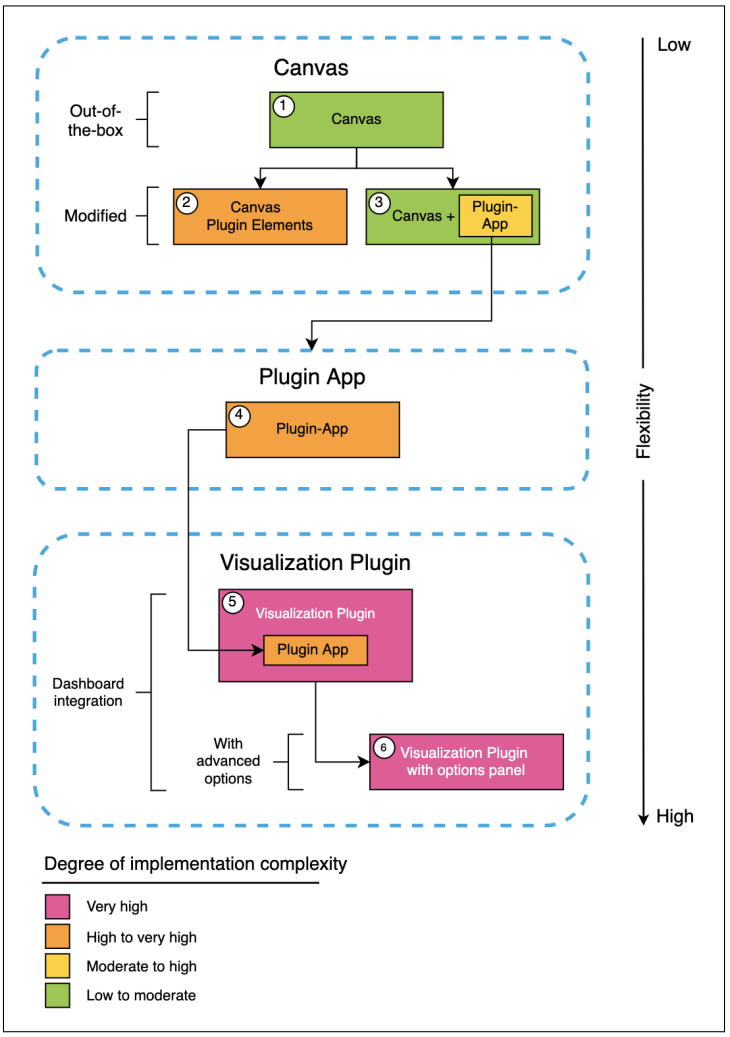

Abbildung 14 Benutzerdefinierte Kibana-Integrationen: Flexibilität und Komplexität [Eigene Darstellung]

bereits im Auslieferungszustand von Kibana vorinstalliert. Der Nachteil dies Integrationsansatzes von erweiterter Funktionalität ist die geringe Flexibilität. Hiermit lassen sich nur eine begrenzte Anzahl an Funktionselementen in das Canvas Workpad integrieren. Diese Canvas-Elemente haben den Nachteil, dass sie nicht interaktiv sind und Datenabfragen sind nur in einer begrenzen Komplexität möglich. Durch komplexere Erweiterungen kann die Flexibilität des Kibana Canvas erweitert werden. Einerseits kann durch die Entwicklung integrierbarer (2) Canvas Elementen als Plugins, die funktionale Flexibilität erhört werden. Andererseits kann unter Verwendung eines (3) parallel zum Canvas verwendeten Kibana Plugins der Grad

der funktionalen Flexibilität erhöht werden. Dabei werden Funktionalitäten die mit Standard-Werkzeugen des Kibana Canvas nicht umsetzbar sind in ein separates Plugin-Modul verlagert. Zusätzliche Funktionalitäten können z.B. durch interaktive Elemente wie z.B. Buttons oder Input-Felder integriert werden. Abgesehen von Funktionserweiterungen von kleineren Umfang, können Programmerweiterungen auch ausschließlich mit Hilfe eines (4) Kibana Plugins durchgeführt werden. Möchte man das Kibana Plugin in einem Dashboard integriert nutzen, so kann dies über ein sogenanntes (5) Kibana Visualization Plugin erfolgen, bei dem zunächst nur eine einfache Integration durchgeführt wird. Durch fortgeschrittene Optionen kann das in das Dashboard integrierte Visualisierung-Plugin angepasst werden. Dafür muss die Integration zusätzlich programmatisch angepasst werden.

5.4 Entwurf und Implementierung der Integration

5.4.1 Integration als Kibana Canvas

Kibana Canvas ist ein Datenvisualisierungs- und Präsentationswerkzeug in Kibana. Mit Canvas können Live-Daten direkt von Elasticsearch abgerufen und die Daten mit Farben, Bildern und Texten zur dynamische Anzeige von Grafiken kombiniert werden [Can20]. Elasticsearch SQL ist eine SQL[11]-ähnliche Abfragesprache für Elasticsearch-Abfragen in Echtzeit und im Canvas Workpad verwendet werden kann. Man kann sich Elasticsearch SQL als einen Übersetzer vorstellen, der sowohl SQL als auch Elasticsearch versteht und durch die Nutzung der Elasticsearch-Funktionen das Lesen und Verarbeiten von Daten in Echtzeit erleichtert [Esql20].

Bei dem Ansatz zur Integration des WMS über ein Kibana Canvas, wurde zunächst ein neues Projekt, welches auch als Canvas-Workpad bezeichnet wird, erstellt. Im Canvas-Workpad wurde anschließend die Visualisierung visuell als Prozesslandkarte modelliert, in dem sie um weitere Elemente erweitert wurde. Abbildung 15 zeigt das modellierte Canvas-Workpad. Es wurden Text-, Form-, Tabellen- und Metrikelemente dem Workpad hinzugefügt.

[11] Datenbankabfragesprache [Cham74]

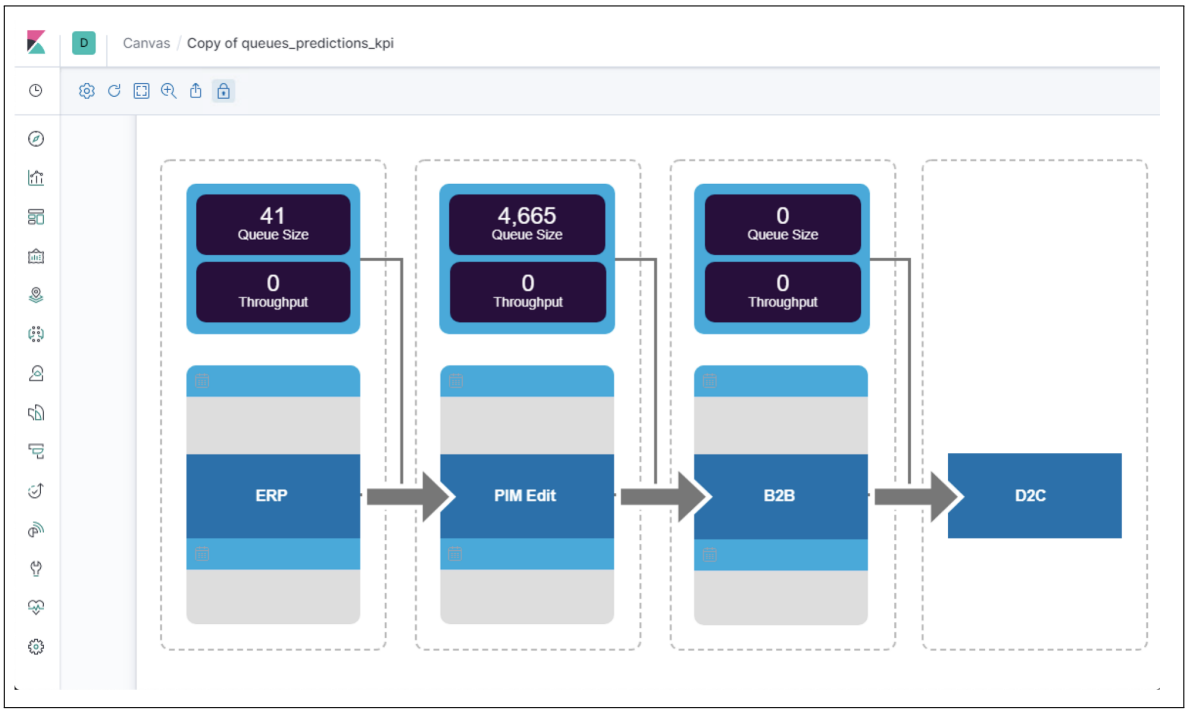

Abbildung 15 Kibana Canvas Rapid-Prototype [Eigene Darstellung]

Anschließend wurden Datenanfragen definiert, wodurch gewünschte Daten aus den Elasticsearch Indices abgebildet werden sollten. Eintritts- und Austrittszeitpunkte eines in einer Warteschlange konnten z.B. durch folgende Datenabfrage (Query) über ESSQL erfolgen:

```
SELECT size FROM index_name WHERE name='queue_name'
AND tier='tier_name' ORDER BY timestamp DESC LIMIT 1
```
[1]

Die Ermittlung des Durchsatzes mit ausschlich ESSQL Funktionen ist wesentlich Komplexer dar. In jedem neuen Elasticsearch-Dokument die Größe des Array-Feldes in jedem nachfolgenden Dokument abnehmen oder zunehmen. Da auch die Größe der Elementliste zunehmen kann, wollen wir eine Abfrage definieren, die bereits verarbeitete Elemente ignoriert. Folgende Frage stellte sich, ob es möglich ist, die Elementenanzahl zu berechnen, indem man nur vorgefertigte Abfragefunktionen wie die Kombination mehrerer Abfragen verwendet, die als eine Abfrage verwendet werden können, um den berechneten Wert in einem Kibana Canvas-Element darzustellen. In diesem Zusammenhang sind folgende Herausforderungen zu lösen:

1. Verwendung von Funktionen, wie `COUNT`[12] in ESSQL
2. Verwendung der Array-Funktion in ESSQL, wie like `INTERSECT`[13] aus SQL
3. Wiederverwendung von Unterabfragen in SSQL
4. Verwendung von mehreren Indices in einer ESSQL Abfrage

Die Anzahl der Elemente des Dokuments des frühsten Zeitpunkts des betrachteten Zeitraums (z.B. Letzte Stunde):

```
SELECT items FROM index_name
WHERE timestamp < NOW()
AND timestamp > NOW() - INTERVAL 1 HOUR
ORDER BY timestamp ASC LIMIT 1
```
[2]

Hier soll mit Hilfe einer Array-Funktion auf das Feld `items`, die Anzahl aller Elemente in dem Array als Ergebnis ermittelt werden. Das selbe soll auch für das späteste Dokument des ausgewählten Zeitraums durchgeführt werden:

```
SELECT items FROM index_name
WHERE timestamp < NOW()
AND timestamp > NOW() - INTERVAL 1 HOUR
ORDER BY timestamp DESC LIMIT 1
```
[3]

Nachfolgend muss die Anzahl der Elemente der Schnittmenge zwischen den Mengen, des frühsten und spätesten Dokuments ermittelt werden:

[12] SQL-Funktion [Cham74 S.253]

[13] SQL-Array-Funktion

```
SELECT items FROM index_name WHERE timestamp < NOW()
AND timestamp > NOW() - INTERVAL 1 HOUR
ORDER BY timestamp ASC LIMIT 1
INTERSECT
SELECT items FROM index_name
WHERE timestamp < NOW()
AND timestamp > NOW() - INTERVAL 1 HOUR
ORDER BY timestamp DESC LIMIT 1
```
[4]

Als nächstes ist die Differenz der Anzahl der Elemente des frühsten und spätesten Dokuments zu ermitteln:

```
(Query 1) - (Query 4)
```
[5]

Damit eine Datenabfrage wie in dem zuvor beschriebenen Szenario über ESSQL innerhalb von Kibana Canvas möglich ist, darf jedes Canvas Element nur eine Datenquelle besitzen. Eine Abfrage muss somit alle vier zuvor genannten Abfragen zusammenfassen, damit eine Ausgabe in dem Zielelement erfolgen kann.

Nach intensiver Recherche über die Realisierung dieses Ansatzes, wurden folgende Erkenntnisse gewonnen. Elasticsearch SQL unterstützt in der aktuellsten Version keine Array-Funktionen wie `Intersect`, das auch in SQL Verwendung findet. Somit sind Schnittmengenbildungen zweier spaltenseparierten Listen aus dem Elasticsearch Index nicht möglich. Außerdem sind keine mehrfach verschachtelten Dokumente nutzbar, so dass eine Abfrage nicht auf mehr als ein verschachteltes Feld in einem Index verweisen kann. Dies gilt für mehrstufig verschachtelte Felder, aber auch für mehrere verschachtelte Felder, die auf der gleichen Ebene definiert sind [EsL20]. Es lässt zusammenfassend sagen, dass keine komplexen Datenbankabfragen mit der Datenabfrage-Sprache von Kibana Canvas nicht möglich sind. Eine Realisierung sowohl als ganzheitliche WMS, als auch nur als Warteschlangen-Auslastungssystem nicht in den zuvor im Kapitel 3 definierten Anforderungen nicht möglich.

Durch das Fehlen von interaktiven Elementen wie z.B. benutzerdefinierte Buttons oder Suchfelder, können Funktionen, wie z.B. eine manuelle Aktualisierung der Vorhersagedaten oder Filterung von Daten, nicht gewährleistet werden. Dieser Ansatz könnte in späteren mächtigeren Version von ESSQL als Lösungsentwicklung in Erwägung gezogen werden, da es noch in einem frühen Stadium der Entwicklung befindet. Denn der große Vorteil bei der Entwicklung eines Visualisierung-Systems mit Canvas ist, dass durch sie eine schnelle und einfache Entwicklung begünstigt wird.

5.4.2 Kibana Canvas Erweiterung

Weil die Umsetzung des WMS in dem geforderten Funktionsumfang über Standwerkzeug in Kibana nicht implementiert werden kann. Untersuchen wir die Erweiterungsoptionen eines Kibana Canvas. Die Auswahl an Funktionselementen im Workpad lassen sich theoretisch erweitern. Das Einbringen eines benutzerdefinierten Elements oder einer Funktion in Canvas beginnt wie bei jedem anderen benutzerdefinierten Code in Kibana, mit einem Plugin. Es gibt einige wenige Abhängigkeiten, die das Plugin benötigt. Die wichtigste ist der Interpreter, den Kibana und Canvas verwenden, um Ausdrücke der sogenannten Kibana-Expression-Language zu registrieren und auszuwerten. Es können beliebige Modifikationen in das Plugin geschrieben werden Es ist zu beachten, dass die Elemente und Funktionen, zur Verfügungstellung in Kibana, registriert werden müssen [Dev20].

Benutzerdefinierte Kibana Canvas Plugin Element Plugins lassen sich zwar bereit in der aktuellsten Version (Stand Version 7.8.1) in Kibana programmatisch implementieren, jedoch existiert für diesen Ansatz keine offizielle Dokumentation und es sind inkompatible Änderungen in zukünftigen Versionen zu erwarten. Elastic hat sich bei der Entwicklung dieses Features noch zu keiner Endgültigen Architektur entschieden und es sind daher inkompatible Änderungen zu erwarten, wodurch bei Aktualisierungen das Risiko der Entstehung von großen Wartungsaufwänden droht und somit gegen die nicht-funktionale Anforderung der Wartbarkeit (NF4 in Kapitel 3) verstößt.

5.4.3 Integration als Kibana Plugin App

In den Abschnitten 6.4.1 bis 6.4.2 konnten die Integrationen den Anforderungen nicht gerecht werden. Benutzerdefinierte Integration einer Visualisierungslösung muss in Anbetracht der zum gegeben Zeitpunkt Nichtrealisierbarkeit einer Lösung mit ausschließlich Standardwerkzeugen, programmatisch erfolgen. Hierfür eignet sich der Ansatz über die Entwicklung eines Kibana Plugins. Es gibt diverse Optionen benutzerdefinierte Plugins für Kibana zu entwicklen. Eine Möglichkeit besteht über die Entwicklung einer Kibana Plugin App. Im folgenden möchten wir eine Architektur zur Integration der Visualisierungslösung über den Ansatz als Kibana Plugin App vorstellen. Zu Beginn soll ein kurzer Überblick über die Funktionsweise des Systems gegeben werden. Es werden zunächst die Plugin-Architektur und der Datenfluss innerhalb des Systems in der Kibana-Plattform und mit externen Beteiligten Systemen, dargestellt.

Zunächst wurde folgendermaßen ein Plugin-Grundgerüst mit der Javascript Runtime Node.js[14] generiert: `node /scripts/generate_plugin.js`. Diese Programm wird von der Kibana- Entwicklungsumgebung zur Verfügung gestellt. Nachfolgend ist das automatisch generierte Plugin im Verzeichnis `/Plugins` im Kibana Projektverzeichnis auffindbar. Der Client-Code ist im Verzeichnis `/public` und das Backend in `/server` (Siehe Abbildung I). Die Generierung des Plugins ist als einfach zu bewerten. Folgende Abhängigkeiten bei der Entwicklung einer Kibana Plugin App sind zu beachten. Die Pluginentwicklung setzt technologische Abhängigkeiten voraus. Neben der Javascript Runtime Node.js, wird noch für die Entwicklung der UI eine Frontend-Technologie vorausgesetzt, weil Kibana Apps als SPA implementiert werden. Für die UI-Entwicklung wird die Javascript-Bibliothek React.js unterstützt.

Das Frontend haben wir bereits zuvor als React-Applikation implentiert (Kapitel 4.2). Daher muss das System nun nur noch in die Kibana-Plattform integriert werden. React Kom-

[14] https://nodejs.org/en/

ponenten konnten übernommen werden, sodass diese in der Kibana App angezeigt werden können. Die Komponenten: Suchfeld, Dropdown und Buttons wurden durch entsprechende Elastic UI[15] (EUI) Komponente ersetzt. Elastic UI ist eine Design-Bibliothek, die von Elastic verwendet wird, um interne Produkte zu implementieren. Sie entsprechen den Vorgaben von Elastic. EUI kann auch bei der Entwicklung von Kibana-Plugins verwendet werden. Zur ästhetischen Angleichung des Plugins an die Kibana-Plattform, wurde EUI beim WMS Plugin verwendet.

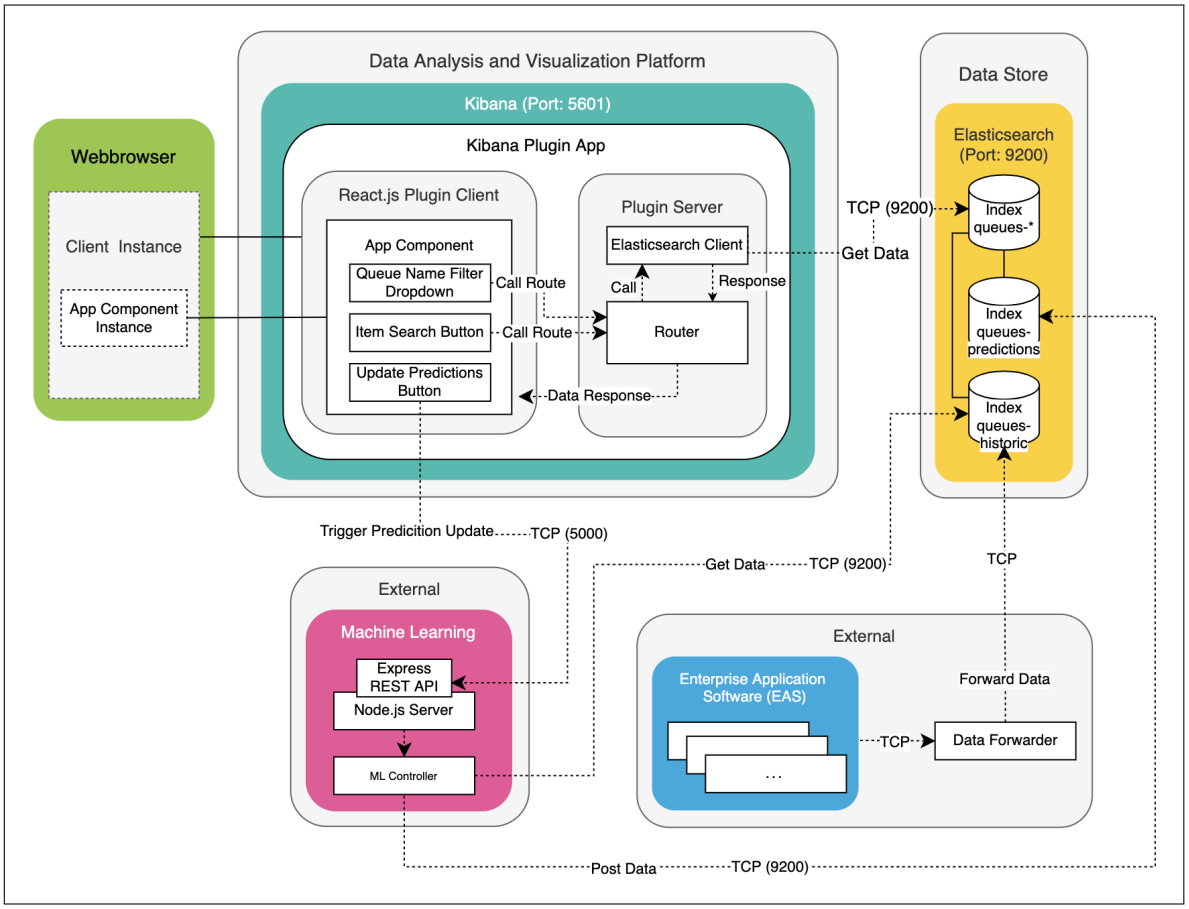

Abbildung 16 High-level Modul-Abhängigkeitsdiagramm [Eigene Darstellung]

Die Queries werden bei einem Kibana Plugin in das `/routes` Verzeichnis ausgelagert und in `index.ts` definiert. Quelltext 2 zeigt ein Beispiel einer Definition einer Route zur Datenabfrage an das gewünschte Elasticsearch Index. Die Route wird vom Frontend

[15] https://elastic.github.io/eui/#/

über einen HTTP-Request aufgerufen. In Quelltext 3 wird ein POST-Request mit Query-Parametern an die an den Router im Plugin-Server gesendet. Der Router sendet eine Anfrage an die interne Elasticsearch API, welche mit Elasticseachsearch kommuniziert.

Eine Realisierung der Aktualisierungsfunktion von Vorhersagedaten wurde in diesem Integrationsansatz über eine Request-Anfrage an einen API-Server implementiert. Dabei wird das Starten einer Python Anwendung angestoßen, wodurch die Vorhersagen durch Maschinellen Lernens aktualisiert und an die Ziel Datenbank gesendet wird. Die Modelle des Maschinellen Lernens und das ML Management sind kein Bestandteil dieser Arbeit. Es wurde lediglich Infrastrukturelle Ansätze im Kontext von ML Prozessen über das vorgestellte Proof of Concept (POC) erprobt.

Das Kibana Plugin hat den höchsten Grad der Flexibilität bei der Implementierung von Funktionen (siehe Abbildung 14). Besonders hervorzuheben ist die Unterstützung der mächtigen Query DSL, die es uns ermöglicht sehr komplexe Datenabfragen zu definieren, die vom Plugin verarbeitet werden können. Außerdem war es uns, im Gegensatz zu vorläufigen Integrationsiteration, gelungen im Plugin interaktive GUI Elemente wie zB Buttons, Suchfelder, Dropdown Filterung, etc. zu verwenden. Abbildung 17 zeigt die das entwicklete und in Kibana integrierte WMS Plugin.

Der Kibana-Plugin-Integrationsansatz ist jedoch auch kritisch zu betrachten. Die Entwicklung von benutzerdefinierter Software nach Kundenbedürfnissen kann aufgrund des hohen Grades der Implementierungskomplexität einen signifikant höheren Mehraufwand in der Entwicklung und Anpassungsarbeit haben. Hierbei muss bei der Integration in Hostinfrastrukturen abgewogen werden wie hoch der Grad der Modifizierung der Standardprogrammcodes, sein darf, sodass eine Update-Fähigkeit noch gewährleistet wird [Ber16 S.13].

Außerdem ist bei dem Einsatz vom Kibana Plugin zu beachten, dass die Kibana-Plugin Schnittstellen in ständiger Weiterentwicklung befinden. Aufgrund der hohen Änderungsrate

kann keine Garantie zur Rückwärtskompatibilität von Kibana angeboten werden. Kibana erzwingt, dass die installierten Plugins mit der von Kibana übereinstimmen. Plugin-Entwickler müssen daher für jedem neuen Kibana-Release eine neue Version ihres Plugins veröffentlichen [Plug20].

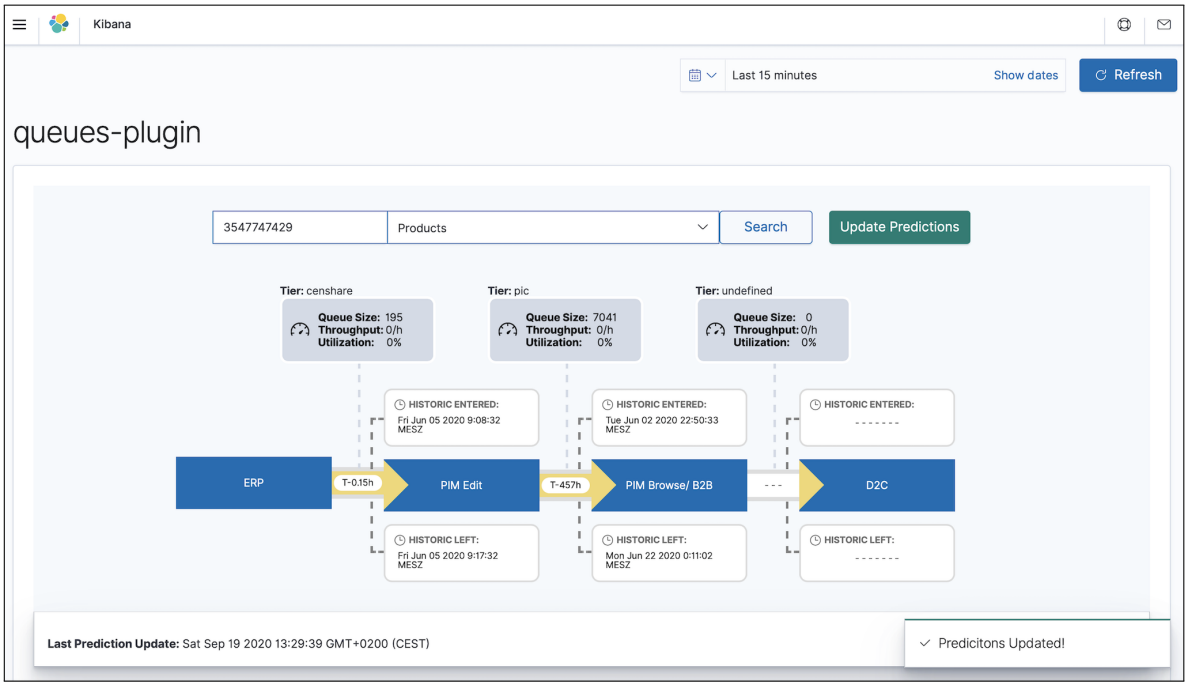

Abbildung 17 Demo Queues Kibana Plugin [Eigene Darstellung]

6 Evaluation

In diesem Kapitel stellen wir die Ergebnisse aus einer qualitativen Befragung eines Endnutzers des entwickelten Systems vor und untersuchen die verschiedenen Integrations-Ansätze aus dem ADR-Prozess hinsichtlich der in Kapitel 3 definierten Qualitätsmerkmale.

Um die Nützlichkeit der vorgestellten WMS zu untersuchen, führten wir eine ADR-basierte Prototyp-Evaluierung mit einem Endnutzer (Support-Mitarbeiter) durch. Wir baten darum, den Software-Prototyp zu benutzen. Es soll sie bei ihrer Arbeit unterstützen. Die genutzten Daten waren die Realdaten aus der BSH Infrastruktur. Der Teilnehmer der Befragung setzt das Plugin derzeit in der Produktionsumgebung ein. Der Proband bevorzugte das neue System und machte folgende Anmerkungen zu den Verbesserungen, die der Prototyp im Vergleich zum bestehenden System bietet:

- Der Prototyp bietet eine Visualisierung mit servicerelevanten Metriken in einer Ansicht.
- Die Prozesslandkarten-Visualisierung stellt eine benutzerfreundliche Möglichkeit bei Informationsbedarfsdeckung dar.
- Die prädiktive Analyse ist hinsichtlich der Verbesserung der Effizient bei der Ermittlung von MTTR sehr hilfreich.

Folgende Verbesserungsvorschläge machte der Teilnehmer während der Auswertung des ADR-basierten Prototyps:

- Weitere Datenquellen hinzufügen: B2C Queue Daten mit aufnehmen und durchsuchbar machen.
- Ausprägung des UI Interfaces gestalten: Farbliche Markierung von langsam laufenden Komponenten.

Qualitätsmerkmal	Prototyp	Kibana Canvas	Canvas Erweiterung	Kibana Plugin
Zuverlässigkeit	✓	✘	✓	✓
Benutzbarkeit	✘	✓	✓	✓
Effizienz	✘	✓	✓	✓
Wartbarkeit	✓	✓	✘	✓
Portabilität	✓	✘	✘	✓

Tabelle 2 Integrationsarchitekturen im Vergleich: Qualitätserfüllung [eigene Darstellung]

In jeder Entwicklungsiteration wurden Artefakte entwickelt. Die verschiedenen Ansätze sollen nun hinsichtlich folgender Qualitätsmerkmale verglichen werden. Die Kriterien dieser Bewertung wurden aus den grundlegenden Eigenschaften eines Systems abgeleitet, die im Architekturentwurf berücksichtigt werden müssen, welche wir bereits im Kapitel 3.2 beschrieben haben. Wir betrachten die Merkmale: Zuverlässigkeit, Benutzbarkeit, Effizienz, Wartbarkeit und Portabilität bei den jeweilen Integrationsansätzen.

In der Tabelle 2 wurden die jeweiligen Bewertungen hinsichtlicher der Qualitätsmerkmale zusammengefasst. Es lässt sich feststellen, dass die Integration über ein Kibana Plugin den Merkmalen betreffend den höchsten Grad der Erfüllung aufweist. Es ist jedoch hinzuweisen, dass dieser Vergleich lediglich einen Momentaufnahme der Ansätze darstellt. In zukünftigen Version kann sich die Qualitätserfüllung ändern. Besonders das Kibana Canvas ist ein Ansatz mit vielversprechenden Aussichten. Aufgrund der rasanten Entwicklungsdynamik des Softwareherstellers Elastic, ist es wahrscheinlich, dass dieser Integrationsansatz zukünftig eine höhere Qualitätserfüllung aufweisen konnte.

7 Fazit

Dieser Abschnitt widmet sich abschließend unter Berücksichtigung der ursprünglichen Zielsetzung der Zusammenfassung der erarbeiteten Ergebnisse. Die Ergebnisse werden anschließend evaluiert. Außerdem werden Ideen für zukünftige Erweiterungen und Optimierungen der Anwendung und dessen Funktionsumfang diskutiert.

7.1 Zusammenfassung

Im Rahmen dieser Arbeit wurde eine Architektur, die die Integration einer React Anwendung als Warteschlangen-Monitoring System, in das Datenanalyse- und -visualisierungsplattform Kibana, ermöglicht, entwickelt. Es wurde die ADR-Methodik verwendet, um das neuartige System zu entwickeln. Die Tabelle 3 fasst den den wesentlichen Forschungsprozess mit ihren Phasen und Ergebnissen, die im Rahmen dieser Arbeit gesammelt wurden zusammen. Das Ziel des Systems war Support-Mitarbeitern dabei zu unterstützen, realistische Aussagen über MTTR in PIM Prozessen und Informationen über die Auslastung von PIA-Warteschlangen in verteilten Umgebungen zu erhalten. Wir stützen unser Design auf die Analyseergebnisse der PIM-Infrastruktur der BSH, einem der größten Gerätehersteller weltweit.

Im Kapitel 1 wurde in dieser Arbeit die Motivation, Problemstellung und Zielsetzung vorgestellt. Im darauffolgenden Kapitel 2 wurden dem Leser alle theoretischen Grundlagen vermittelt, welche für die darauffolgenden Abschnitte für das Verständnis notwendig sind. In Kapitel 3 führten wir eine Anforderungsanalyse einer Visualisierungslösung als WMS durch. Unter Berücksichtigung der zuvor definierten Anforderungen an das zu entwickelnde System, wurde das WMS System, im Kapitel 4 als Prototyp entwickelt. Anschließend wird der Leser im Kapitel 5 in die technischen Rahmenbedingung der Kibana-Plattform eingeführt. Die Architektur von Kibana wird analysiert. Nachfolgend werden auf Basis der zuvor durchgeführten Analyse Integrationsoptionen vorstellt und iterativ erprobt. Beginnend von der geringsten Integrationskomplexität wurde eine Integration des zuvor entwickelten Visualisierungssystem vorgenommen. Wir präsentieren die Ergebnisse der Prototyp-Evaluierung

mit Anwendern aus der realen Welt. In jeder Entwicklungsiteration wurden die Integrationen über Feedback-Gespräche mit einem Domäennutzer evaluiert und optimiert. Anschließend wurde das WMS final in die Infrastruktur der BSH integriert und in Produktion von Anwendern getestet. Die Ergebnisse dieser Arbeit wurden in Kapitel 6 dokumentiert. Hierbei wurden auch die Herausforderungen, welche während der Umsetzung der Integrationsarchitektur auftraten, dargelegt. Daraufhin folgte eine Überprüfung der im Kapitel 3 definierten Anforderungen an die umgesetzte Integrationsarchitektur. Dabei zeigte sich, dass die integrierte Plugin-Architektur im Wesentlichen die geforderten Anforderungen erfüllt. Weiterhin konnte festgestellt werden, dass die zu Beginn dieser Arbeit definierten Ziele (vgl. Abschnitt 1.3) vollumfänglich erreicht wurden.

Zusammenfassend lässt sich sagen, dass das WMS nun über eine funktionale, performante und einfach zu bedienende Benutzeroberfläche verfügt. Die vorliegende Arbeit zeigt, dass die Entwicklung einer WMS Lösung für verteilte Umgebungen und ihre Integration in Kibana, möglich ist.

7.2 Ausblick

Im Hinblick auf die Weiterentwicklung des Kibana Plugins steht zunächst die Implementierung weiterer Funktionen im Vordergrund. Neben den Optimierungsvorschlägen aus den Interviews mit Endnutzern (siehe Kapitel 6) können weitere Funktionserweiterungen am System zukünftig vorgenommen werden. Zur Erhöhung der Anpassungsfähigkeit auf Änderungen der abzubildenden Systemlandschaft und somit schnell und einfach auf Veränderungen reagieren zu können, ist es sinnvoll das Plugin mit folgenden Funktionen auszustatten. Dem Benutzer sollte die Möglichkeit gegeben werden die Reihenfolge und Anzahl der Anwendungssysteme in der Visualisierung einfach anzupassen. Es sollten beliebig viele Prozessoren in beliebiger Position angelegt, Prozessoren gelöscht oder modifiziert werden können. Namen der Prozessoren oder der gewünschten Warteschlangen können bearbeitet werden. Ebenfalls würden Modifikationen die Position des Systems betreffen. Zur Realisierung einer

Phasen und Prinzipien		Artefakt
Phase 1: Problemformulierung		
Prinzip 1: Von der Praxis inspirierte Untersuchung	Die Forschung wurde angestoßen durch den Bedarf an besserer IT-Unterstützung der Produkt-Support-Spezialisten.	**Erkenntnis**: Es wurde festgestellt, dass den Unzulänglichkeiten des bestehenden Warteschlangen-Monitoring-Systems die für das effektive Vorhersagen von Produktinformations-Problembehebungen erforderliche Genauigkeit fehlt.
Prinzip 2: Theoriegeleitetes Artefakt.	Die verwendete Theorie war die Warteschlangentheorie, den daraus abgeleiteten Leistungsgrößen, Grundlagen zu PIM und Plugins.	
Phase 2: Aufbau, Intervention und Evaluation (BIE)		
Prinzip 3: Reziproke Gestaltung	Es wurde erwartet, dass die Ungenauigkeit in der Vorhersage von Produktinformation-Problembehebungen ein anhaltendes Problem sein würde. Die aufgetretenen Probleme wurden iterativ angegangen mit Praktikern als frühe Entwurfsprinzipien formuliert.	**Alpha Version**: Das Artefakt wurde als Design-Idee konzipiert. Es sollte die auf die Systemlandschaft der BSH basierenden verteilte Umgebung widerspiegeln. **Beta Version**: Das Prototyp-System wurde entworfen, um Vorhersagedaten von Produkt-Verarbeitungszeitpunkten in Warteschlangen und deren Metriken zu visualisieren.
Prinzip 4: Wechselseitig beinflussende Rollen	Im ADR Prozess wurden sowohl die Rollen des Forschers als auch die des Praktikers berücksichtigt, um theoretische, technische und praktische Perspektiven einzubeziehen.	
Prinzip 5: Authentische und begleitende Evaluation	Das Prototyp-System wurde zunächst intern und dann im weiteren Umfeld der Endbenutzer bei BSH evaluiert.	
Phase 3: Reflexion und Erkenntnis		
Prinzip 6: Geführte Emergenz	Der Gesamtcharakter des Artefakts wurde anerkannt. Darüber hinaus ergaben sich Gestaltungselemente für die IT-Komponenten und Änderungen der Annahmen in Bezug auf die Arbeitspraktiken.	**Realisierung**: Neue Anforderungen an das Artefakt aufgrund von Ergebnissen, die in der BIE-Phase entstehen. Überarbeitete Version der ursprünglichen Entwurfsprinzipien.
Phase 4: Formalisierung der Erkenntnisse		
Prinzip 7: Verallgemeinerte Ergebnisse	Es wurde eine Auswahl an Entwurfsprinzipien für WMS formuliert, die das Prototyp-System als Instanz definieren.	**Zusammenfassung**: Eine Zusammenfassung der Entwurfsprinzipien und den Vorgaben zur Entwicklung und Integration des Software-Systems in Kibana.

Tabelle 3 Zusammenfassung des ADR Prozesses im WMS Projekt

einfachen und bedienerfreundlichen Positions-Modifikations-Lösung, kann eine Drag-and-Drop-Funktion implementiert werden. Dabei wird durch das „Greifen“ und „Ziehen“ eines grafischen Objekts, beim „los lassen“ die Postion des Objekts geändert [Dra20]. Alternativ kann die Prozesslandkarten-Visualisierung durch eine Konfigurationsdatei beschreiben werden. Hierfür eignen sich Dateiformate, wie z.B. yaml[16]. Diese wird beim Anwendungsstart ausgelesen und die Beschreibung auf die Visualisierung abgebildet. Eine weitere Möglichkeit besteht im Speichern der Konfiguration als JSON-Dokument in einem Elasticsearch Index. Das reduziert weitere technologische Abhängigkeiten, jedoch werden hierfür mehr technisches Fachwissen der Anwender benötigt, da hierfür im Gegensatz zum Bearbeiten einer Textdatei ein POST-Request an den jeweiligen Index (z.B `.queues-conf`) mit der aktualisierten Beschreibung gesendet werden. Dafür kann u.A. der Editor in den Kibana Dev Tools verwendet werden. Ist vom Plattformnutzer eine Integrierbarkeit der Visualisierung in Dashboards gewünscht, so kann das Plugin als Visualization-Plugin in Kibana erweitert werden (siehe Abbildung 14 in Kapitel 4.3). Es können noch weitere Warteschlangenleistungsgrößen im WMS visualisiert werden, wie z.B. die Ankunfts- und Bedienrate (siehe Kapitel 3.1 Warteschlangentheorie). Bei der Darstellung der Eintritts- und Austrittszeitpunkte wird aktuell ausschließlich nur die Mitteleuropäische Zeitzone berücksichtigt. Im Hinblick auf eine Optimierung der Benutzerfreundlichkeit sollte die Datums- und Uhrzeit-Anzeige im Plugin, sich je nach Zeitzone des Clients, automatisch anpassen.

[16] https://yaml.org

8 Literaturverzeichnis

[Sun19] Cha S., R. N. Taylor, Kang K. 2019. Handbook of Software Engineering. Cham, Switzerland: Springer Nature Switzerland AG.

[Shu19] Shukla P., Kumar, 2019. Learning Elastic Stack 7.0. 2. Auflage. Packt Publishing.

[Mar20] BSH Hausgeräte GmbH (2020): Hausgeräte unter 11 Marken, [online] https://www.bsh-group.com/de/marken/hausgeraetemarken [15.08.2020]

[Jp12] IT Administrator (2012): Grundlagen: Monitoring, [online] https://www.it-administrator.de/themen/netzwerkmanagement/grundlagen/111034.html [01.09.2020]

[dsl20] Elasticsearch Dokumentation (2020): Query DSL, [online] https://www.elastic.co/guide/en/elasticsearch/reference/current/query-dsl.html [22.09.2020]

[Tia06] Tian, N., Zhang, Z. G. 2006. Vacation Queueing Models Theory and Applications. New York, NY: Springer.

[Kyn10] Kyne F. et al. 2010. International Technical Support Organization System z Mean Time to Recovery Best Practices. 1. Auflage. North Castle Drive, Armonk, NY: International Business Machines Corporation.

[Ber16] Bertram, M., 2016. The Strategic Role of Software Customization: Managing Customization-Enabled Software Product Development. 1. Auflage. Wiesbaden: Springer Gabler.

[Zam20] Zammetti F., 2020. Modern Full-Stack Development: Using TypeScript, React, Node.js, Webpack, and Docker. Pottstown, PA, USA: Apress Media LLC: Welmoed Spahr.

[Typ20] TypeScript (2020): TypeScript Documentation, [online]
https://www.typescriptlang.org/docs [01.09.2020].

[Hans12] Hanschke, T. (2012): Das Grundmodell der Warteschlangentheorie, [online]
https://www.mathematik.tu-clausthal.de/arbeitsgruppen/stochastik/public-relations/das-grundmodell-der-warteschlangentheorie [12.06.2020]

[Weis09] Weiß, A. (2009): Warteschlangen. Proseminar Technische Information, [online]
https://www.mi.fu-berlin.de/inf/groups/ag-tech/teaching/2009-10_WS
/S_19510b_Proseminar_Technische_Informatik/weiss10warteschlangen.pdf [07.07.2020]

[Chi19] Chircu, A. et al. 2019. Real-Time 3D Visualization of Queues with Embedded ML-based Prediction of Item Processing for a Product Information Management System.

[Dov16] Dover J. (2016): Introducing a new architecture for Kibana - Key aspects of the Kibana platform, [online] https://www.elastic.co/de/blog/introducing-a-new-architecture-for-kibana [01.09.2020]

[Hin06] Hintemann, R. (2006): Uptime ist in, Downtime ist out !, [online] https://www.bitkom.org
/sites/default/files/file/import/Hochverfuegbarkeit-Flyer-9-11-2006.pdf [20.06.2020]

[Siep20] Siepermann, M. (2020): Service Level Agreement, [online]
https://wirtschaftslexikon.gabler.de/definition/service-level-agreement-53580#references [01.07.2020]

[Bux12] Buxton I. (2012): Insellösungen bergen Risiken, [online]
https://www.computerwoche.de/a/inselloesungen-bergen-risiken,2503015 [09.09.2020]

[Lud10] Ludewig, J. 2010. Software Engineering - Grundlagen, Menschen, Prozesse, Techniken. 2. Auflage. Heidelberg: dpunkt.verlag.

[Bal09] Balzert H. et al. 2009. Lehrbuch der Softwaretechnik: Basiskonzepte und Requirements Engineering. 2. Auflage. Heidelberg: Spektrum Akademischer Verlag.

[Hel20] Helber, S. 2020. Operations Management Tutorial - Grundlagen der Modellierung und Analyse der betrieblichen Wertschöpfung. 2. Auflage. Hildesheim: Stefan Helber.

[Zim14] Zimmermann, A. (2014). Leistungsbewertung - Teil 2 Modellierung, [online] https://www.tu-ilmenau.de/fileadmin/media/telematik/lehre/Leistungsbewertung/Handouts/07_Modellierung.pdf [27.07.2020]

[Dom94] Dombacher, C. (1994): Warteschlangen, [online] http://www.telecomm.at/documents/Warteschlangen.pdf [19.08.2020]

[Reu16] Reucher E. (2016): Planen mit mathematischen Modellen: Stochastische Simulation - Techniken und Anwendungen, [online] https://www.fernuni-hagen.de/BWLOR/assets/papers/iww_k00859-warteschlangen.pdf [14.08.2020]

[Mod08] Modiano, E. (2008): Lectures 5 & 6 Introduction to Queueing Theory. Massachusetts Institute of Technology, [online] http://web.mit.edu/modiano/www/6.263/lec5-6.pdf [03.09.2020]

[Wil06] Wilde, T., Hess, T. 2006: Methodenspektrum der Wirtschaftsinformatik: Überblick und Portfoliobildung. München: Institut für Wirtschaftsinformatik und Neue Medien der Ludwig-Maximilians-Universität München.

[Ken53] Kendall, D. G. 1953. Stochastic Processes Occurring in the Theory of Queues and their Analysis by the Method of the Imbedded Markov Chain. 3. Auflage. USA: Annals of Mathematical Statistics.

[Chh08] Chhajed, D., Lowe, T. J. 2008. Little's Law. Building Intuition: Insights from Basic Operations Management Models and Principles. London: Springer.

[KOZ06] Kozdron, M. (2006): Introduction to Stochastic Processes Notes, [online] http://stat.math.uregina.ca/ kozdron/Teaching/Regina/862Winter06/Handouts /revised_lecture1.pdf [01.07.2020]

[KiDemo] Elastic NV (2020): Kibana Demo, [online] https://demo.elastic.co/app/kibana# /dashboard/7adfa750-4c81-11e8-b3d7-01146121b73d?_a=(description:'Analyze %20mock%20flight%20data%20for%20ES-Air,%20Logstash%20Airways,%20Kibana %20Airlines%20and%20JetBeats',filters:!(),fullScreenMode:!f,options: (hidePanelTitles:!f,useMargins:!t),query:(language:kuery,query:''),timeRestore:!t,title: '%5BFlights%5D%20Global%20Flight%20Dashboard',viewMode:view)&_g= (filters:!(),refreshInterval:(pause:!f,value:900000),time:(from:now-13w,to:now)) [30.06.2020]

[Plus99] Plusquellic, J. M. (1999): Queuing theory, [online] http://ece-research.unm.edu/jimp/611 /slides/chap6_3.html [21.07.2020]

[Hel07] Held, P. (2007): Beschreibung von Warteschlangensystemen und deren psychologische Einflüsse auf den Menschen, [online] http://isg.cs.uni-magdeburg.de/sim/vilab/2007 /papers/06_queues_pheld.pdf [25.07.2020]

[Bsh20] BSH GmbH: Unternehmensprofil, [online] https://www.bsh-group.com/de/unternehmen /unternehmensprofil [20.06.2020]

[Gri18] Grizault, C.: Everything you need to know about PIM, Akaneo, [online] https://magento.com/sites/default/files8/2019-01/product-information-management-101.pdf [20.06.2020]

[For06] Forza C., Salvador F. (2006): Management for Mass Customization Connecting Customer, Front-office and Back-office for Fast and Efficient Customization. London, UK: Palgrave Maximillan, [online] https://pdfs.semanticscholar.org/4918
/753be9f76c6ac43e63e0797a8f5cc71d2541.pdf [10.09.2020]

[Gup15] Gupta Y. 2015. Kibana Essentials. 3.[überarbeitete] Auflage. Studium. Birmingham: Packt Publishing Ltd.

[Can20] Elastic (2020): Kibana Canvas Documentation, [online]
https://www.elastic.co/guide/en/kibana/current/canvas.html [20.08.2020]

[Dev20] Elastic (2020): Kibana Developer Guide 7.9, [online]
https://www.elastic.co/guide/en/kibana/current/development.html [01.09.2020]

[EsL20] Elastic (2020): Elasticsearch Reference 7.9, SQL Limitations, [online]
https://www.elastic.co/guide/en/elasticsearch/reference/current/sql-limitations.html [12.08.2020]

[Cha20] Chamoni, P. (2020): Gabler Lexikon: Dashboard, [online]
https://www.gabler-banklexikon.de/definition/dashboard-70726 [03.09.2020]

[Sei11] Sein, M. K. et al. (2011): Action Design Research. MIS Quarterly, [online]
https://www.researchgate.net/publication/220259912_Action_Design_Research [11.09.2020]

[KibanaFeatures20] Elastic (2020): Kibana-Funktionen, [online]
https://www.elastic.co/de/kibana/features [12.07.2020]

[Bas97] Baskerville, R. 1997. Distinguishing Action Research from Participative Case Studies. Journal of Systems and Information Technology. Bradford, UK: Emerald Publishing Limited.

[Kre20] Kreutzer, R. T. (2020): Gabler Lexikon: Time-to-Market, [online] https://wirtschaftslexikon.gabler.de/definition/time-market-54271 [05.08.2020]

[Cham74] Chamberlin D. D., Boyce, R. F., 1974: Sequel: A Structured English Query Language. IBM Research Laboratory, San Jose, California, [online] https://researcher.watson.ibm.com/researcher/files/us-dchamber/sequel-1974.pdf [19.08.2020]

[Esql20] Elastic (2020): ELASTICSEARCH SQL Dokumentation, [online] https://www.elastic.co/guide/en/elasticsearch/reference/current/xpack-sql.html [02.09.2020]

[Dra20] HTML Drag and Drop API, [online] https://www.w3schools.com/html/html5_draganddrop.asp [19.09.2020]

[css20] W3Schools (2020): CSS Introduction, [online] https://www.w3schools.com/css/css_intro.asp [17.09.2020]

[Plug20] Kibana Dokumentation: Known, [online] https://www.elastic.co/guide/en/kibana/current /known-plugins.html [15.09.2020]

A Anhang

```
const { Client } = require('@elastic/elasticsearch')
const client = new Client({ node: 'http://localhost:9200' })

// promise API
const result = await client.search({
 index: 'my-index',
 body: {
  query: {
   match: { hello: 'world' }
  }
 }
})

// callback API
client.search({
 index: 'my-index',
 body: {
  query: {
   match: { hello: 'world' }
  }
 }
}, (err, result) => {
 if (err) console.log(err)
})
```

Quelltext 1 Beispiel einer Regristrierung einer Route mit Ressourcenabfrage

```
router.post({
 path: '/api/item',
 validate: {
  body: schema.object({
   item: schema.string(),
   name: schema.string(),
   tier: schema.string()
  }),
 }
},
async (context, request, response) => {
 const data = await client.search({
  index: 'queues',
  body: {
   "_source": ["timestamp", "name", "size", "tier"],
   "query": {
    "bool": {
     "must": [
     { "match": { "items": request.body.item }},
     { "match": { "tier": request.body.tier }},
     { "match": { "name": request.body.name }}
     ]
    }
   },
   "aggs": {
    "queue_enter" : {
     "top_hits": {
      "size": 1,
      "sort": [ { "timestamp": { "order": "asc" } } ],
      "_source": { "includes": [ "timestamp", "name", "size", "tier" ] }
     }
    },
    "queue_left" : {
     "top_hits": {
```

```
35          "size": 1,
36          "sort": [ { "timestamp": { "order": "desc" } } ],
37          "_source": { "includes": [ "timestamp", "name", "size", "tier" ] }
38        }
39       }
40      }
41     },
42     "size": 0
43    });
44    return response.ok({
45     body: {
46      data: data,
47     },
48    });
49   });
```

Quelltext 2 Beispiel einer Regristrierung einer Route mit Ressourcenabfrage

```
1 const handler = () => {
2  const body = { item: item, name: name, tier:tier};
3  http.post("/api/item", { body: JSON.stringify(body) }).then(res => {
4   if (res) {
5    setTimestamps(res.data.aggregations);
6   }
7 });
```

Quelltext 3 Beispiel Ressourcenabfrage an Route

```
{
 "mappings": {
  "_doc": {
   "properties": {
    "items": {
     "type": "text"
    },
    "name": {
     "type": "text",
     "fields": {
      "keyword": {
       "type": "keyword",
       "ignore_above": 256
      }
     }
    },
    "size": {
     "type": "text",
     "fields": {
      "keyword": {
       "type": "keyword",
       "ignore_above": 256
      }
     }
    },
    "timestamp": {
     "type": "date"
    }
   }
  }
 }
}
```

Quelltext 4 Mapping des Elasticsearch Index

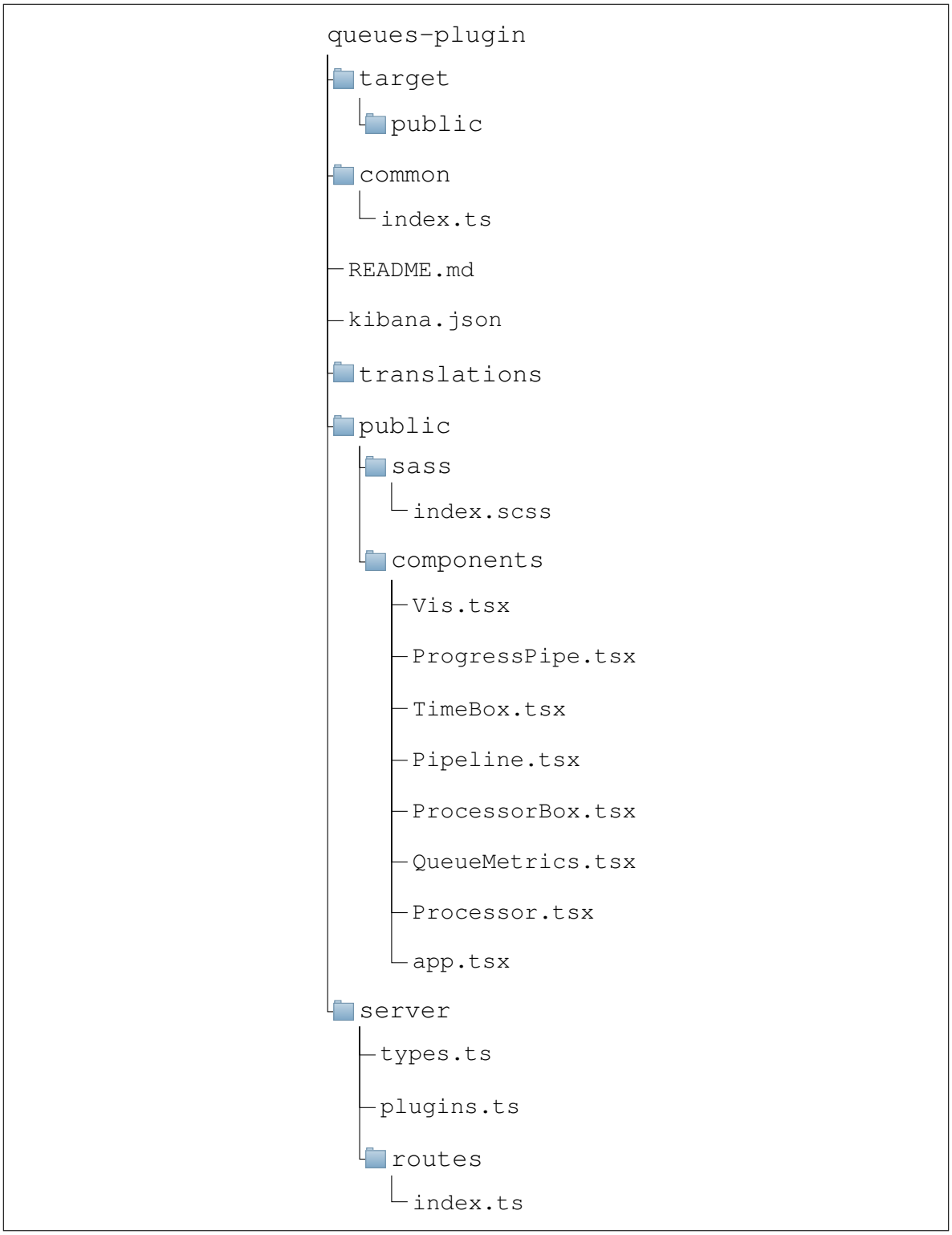

Abbildung I Ordnerstruktur Kibana Plugin App

Ehrenwörtliche Erklärung

Hiermit versichere ich, dass ich die vorliegende Bachelor- oder Masterarbeit mit dem Titel „Aufbau einer Integrationsarchitektur für ein Auslastungsanalysetool in Kibana“ ohne Hilfe Dritter und ohne Zuhilfenahme anderer als der angegebenen Quellen und Hilfsmittel angefertigt habe. Die den benutzten Quellen wörtlich oder inhaltlich entnommenen Stellen sind als solche kenntlich gemacht. Diese Arbeit hat in gleicher oder ähnlicher Form noch keiner Prüfungsbehörde vorgelegen.

BURAK KAYAALP

....

Ort, Datum Unterschrift