WIRTSCHAFT
HOCHSCHULE MAINZ
UNIVERSITY OF
APPLIED SCIENCES

Bachelor-Arbeit

BA WI Dual

Bedeutung von APIs und API-Management für Unternehmen: Ein Verfahren für die funktionale Bewertung und Vergleich von API-Management-Plattformen

Eine Untersuchung am Beispiel der Heraeus Holding GmbH

Hochschule Mainz
University of Applied Sciences
Fachbereich Wirtschaft

Vorgelegt von:	Michael Rudl
	Nordstraße 12, 63505 Langenselbold
	941069
Vorgelegt bei:	Marc Aurel Schubert
Eingereicht am:	30.09.2022

Erklärung

Ich erkläre, dass ich die vorliegende Bachelor-Arbeit

„Bedeutung von APIs und API-Management für Unternehmen: Ein Verfahren für die funktionale Bewertung und Vergleich von API-Management-Plattformen. Eine Untersuchung am Beispiel der Heraeus Holding GmbH"

selbstständig und ohne fremde Hilfe angefertigt habe. Ich habe dabei nur die in der Arbeit angegebenen Quellen und Hilfsmittel benutzt.

Zudem versichere ich, dass ich weder diese noch inhaltlich verwandte Arbeiten als Prüfungsleistung in anderen Fächern eingereicht habe oder einreichen werde.

Langenselbold, 30.09.2022 Unterschrift

Inhaltsverzeichnis

1 Einleitung 1

1.1 Zielsetzung 2

1.2 Methodisches Vorgehen 2

2 Ausgewählte Grundlagen APIs 4

2.1 Funktionsweise einer (Web-)API 4

2.2 Abgrenzung von Web-APIs zu anderen API-Arten 6

2.3 Eigenschaften von APIs 7

2.4 Betriebswirtschaftliche Potentiale durch den Einsatz von APIs 8

3 Definition API-Management und Implikationen für Unternehmen 10

3.1 Fokusbereiche des API-Managements 10

3.1.1 Fokusbereich API-Lifecycle-Management 13

3.1.2 Fokusbereich Security 13

3.1.3 Fokusbereich Performance 14

3.1.4 Fokusbereich Observability 14

3.1.5 Fokusbereich Community 15

3.1.6 Fokusbereich Commercial 16

3.1.7 Abgrenzung API-basierte Integration 16

3.2 Implikationen für Unternehmen 18

4 Architektur und Funktionsweise von API-Management-Plattformen 19

5 Funktionales Bewertungs- und Vergleichsverfahren von API-Management-Plattformen 22

5.1 Ziele formulieren 22

5.2 Bewertungskriterien definieren 23

5.3 Alternativen auswählen 25

5.4 Gewichtungsfaktoren ermitteln 27

5.5 Ermittlung der Zielerfüllungsfaktoren 28

5.6 Ergebnisphase 31

6 Anwenden und evaluieren des Verfahrens bei der Heraeus Holding GmbH 33

7 Fazit und Ausblick 39

8 Retrospektive 41

Literaturverzeichnis V

Anhangsverzeichnis VIII

Abbildungsverzeichnis

Abbildung 1 Client Server Modell Web API in Anlehnung an (Masse, 2011, S. 1–11).....4
Abbildung 2 Beispiel JSON-Objekt 5
Abbildung 3 Beispiel einer URI einer Rest-API in Anlehnung an (Masse, 2011, S. 11–23) 5
Abbildung 4 Beispiel für HATEOAS-Prinzip von Rest-APIs 6
Abbildung 5 Google Suchanfragen zu API-Standards. In Anlehnung an (Weir, 2019, S. 164) 6
Abbildung 6 Begriffsdefinitionen API-Management aus Sicht der Praktik „Implementierung von Schutzmaßnahmen vor DoS Attacken“.....12
Abbildung 7 API-basierte Integration in Anlehnung an (Brajesh, 2017, S. 81–105) 17
Abbildung 8 Architektur von API-Management-Plattformen in Anlehnung an (Brajesh, 2017, S. 15–29). 19
Abbildung 9 Ablauf Nutzwertanalyse in Anlehnung an (Herbig, 2016) 22
Abbildung 10 Magic Quadrant for Full Life Cycle API Management (Pillai & Iijima, 2021) 26
Abbildung 11 Exemplarische Ermittlung der Gewichtungsfaktoren durch paarweisen Vergleich in Anlehnung an (Herbig, 2016) 28
Abbildung 12 Durchschnittliche Zielerfüllungsfaktoren je Fokusbereich nach verglichenen Anbietern 37

Tabellenverzeichnis

Tabelle 1 API-Management-Praktiken der Fähigkeit des API-Management Sicherheitslücken zu erkennen und zu schließen in Anlehnung an (Mathijssen et al., 2022, S. 7) 11
Tabelle 2 Fähigkeiten des API-Managements nach Fokusbereichen in Anlehnung an (Mathijssen et al., 2022, S. 7).....13
Tabelle 3 Bewertungskriterien zur funktionalen Bewertung von API-Management-Plattformen 25
Tabelle 4 Exemplarische Ermittlung hierarchischer Gewichtungsfaktoren in Anlehnung an (Herbig, 2016) 28
Tabelle 5 Direkte Ermittlung der Zielerfüllungsfaktoren durch Überprüfung der implementierbaren Praktiken.....29
Tabelle 6 Indirekte Ermittlung der Zielerfüllungsfaktoren durch Überprüfung der implementierbaren Praktiken.....30
Tabelle 7 Exemplarische Ermittlung der Teil- und Gesamtnutzen 31
Tabelle 8 Zusammengefasste Ergebnisse Paarweiser Vergleich der Stakeholder.....34
Tabelle 9 Berechnung Teil und Gesamtnutzen der Anbieter.....38
Tabelle 10 Berechnung Teil und Gesamtnutzen der Anbieter bei Gleichgewichtung der Fokusbereiche.....38

1 Einleitung

Die in Hanau ansässige Heraeus Holding GmbH bietet über ihre Webanwendung myHeraeus Geschäftskunden die Möglichkeit Edelmetalle zu handeln und Recyclingprozesse abzuwickeln. Die Kunden dieser Anwendung wünschen sich zunehmend die Bereitstellung sogenannter Application Programming Interfaces (APIs). Bei APIs handelt es sich um definierte Systemschnittstellen über die programmatisch auf die Daten und Funktionen einer Anwendung zugegriffen werden kann (Jacobson et al., 2011, S. 1–11). Noch sind diese Anforderungen in der Evaluationsphase. Es fehlt deshalb eine Strategie wie die entwickelten APIs bereitgestellt, betrieben und verwaltet werden sollen.

In Zusammenhang damit wurde innerhalb der Heraeus Holding GmbH bereits die Einführung einer API-Management-Plattform diskutiert. API-Management-Plattformen sind eigenständige Softwaresysteme, die Unternehmen bei der Umsetzung von API-Management unterstützen (Brajesh, 2017, S. 15–29). API-Management wird umgesetzt, indem konkrete API-Management-Praktiken implementiert werden. API-Management-Praktiken sind implementierbare Maßnahmen, die die Nutzung von APIs im Unternehmen fördern und verwalten sollen (Mathijssen et al., 2020, S. 12, 2022, S. 2). Die API-Management-Praktiken sollen idealerweise für die gesamte API-Landschaft konsistent implementiert werden (Medjaoui et al., 2021, S. 7 f.). Die API-Landschaft beschreibt dabei alle vom Unternehmen veröffentlichten und eingesetzten APIs (Medjaoui et al., 2018, S. 2).

Eine Studie aus dem Jahr 2016 befürwortet die Implementierung von API-Management-Praktiken in Unternehmen, mit mehr als einer halben Milliarden Dollar Jahresumsatz (Coleman Parks Research, 2016). Durch eine Befragung von 1770 Entscheidungsträgern von Unternehmen, mit mindestens einer halben Milliarden Dollar Jahresumsatz, wurde herausgefunden, dass 88% dieser Unternehmen APIs im Einsatz haben. Die Unternehmen die API-Management-Praktiken implementierten, wiesen Verbesserungen für Kennzahlen wie IT-bezogene Kosten oder Kundenzufriedenheit auf (Coleman Parks Research, 2016, S. 13). Anzumerken ist jedoch, dass diese Studie von CA Technologies, einem Anbieter einer API-Management-Plattform (Broadcom, o. J.), in Auftrag gegeben wurde (Coleman Parks Research, 2016, S. 17).

API-Management-Praktiken können, müssen aber nicht, von API-Management-Plattformen implementiert werden (Mathijssen et al., 2022, S. 6). Es eignet sich daher eine getrennte Betrachtung zwischen API-Management und API-Management-Plattformen.

1.1 Zielsetzung

Um eine Basis für das Verständnis der Themen dieser Arbeit zu schaffen, werden initial ausgewählte technische Grundlagen des Konzeptes von APIs beschrieben sowie die betriebswirtschaftlichen Potentiale durch den Einsatz von APIs erläutert. Anschließend wird der Rahmen des Themengebietes des API-Managements konkretisiert. Dazu werden einzelne Teilbereiche des Themengebietes vorgestellt und näher beleuchtet. Es wird betrachtet, wie API-Management-Plattformen bei der Umsetzung von API-Management unterstützen können.

Daraus wird die Leitfrage dieser Arbeit abgeleitet, wie ein Verfahren zur funktionalen Bewertung und zum Vergleich von API-Management-Plattformen, unter Berücksichtigung der Anforderungen eines Unternehmens, quantifizierbar gestaltet werden kann. Diese Arbeit soll einen Beitrag zur Beantwortung dieser Fragestellung leisten.

Ein entsprechendes Verfahren soll unter Einbezug der erarbeiteten Erkenntnisse im Themengebiet API-Management ausgearbeitet werden. Unternehmen wird durch dieses Verfahren ermöglicht, API-Management-Plattformen funktional zu bewerten und so miteinander vergleichen zu können. Verschiedene funktionale Aspekte sollen durch die Unternehmen gewichtet werden können, um das Verfahren auf die Anforderungen des Unternehmens anpassen zu können.

Das erarbeitete Verfahren wird anschließend exemplarisch auf die Heraeus Holding GmbH angewandt, um Stärken und Schwächen des Verfahrens zu identifizieren. Abschließend werden die daraus resultierenden Ergebnisse kritisch evaluiert sowie ein Fazit gezogen.

1.2 Methodisches Vorgehen

Diese Arbeit baut auf einer theoretischen Auseinandersetzung mit bestehender Fachliteratur auf. Grundlage für das Verständnis von API-Management bot dabei eine systematische Literatur-Recherche aus dem Jahr 2020 von Mathijssen et al. (Mathijssen et al., 2020). Die referenzierte Literatur dieser Arbeit wurde im Schneeballverfahren analysiert. Durch dieses Verfahren wurden weitere Quellen identifiziert. In einer vorwärtsgewandten Suche ausgehend von der

systematischen Literatur-Recherche von Mathijssen et al. wurde die Quelle „*API-m-FAMM: A focus area maturity model for API Management*“ (Mathijssen et al., 2022) identifiziert. In diesem Journalartikel konkretisieren Mathijssen et al. die Inhalte ihrer systematischen Literatur-Recherche und betrachten, wie der API-Management-Reifegrad von Unternehmen bestimmt werden kann. Der API-Management-Reifegrad betrachtet dabei, welche API-Management-Praktiken ein Unternehmen implementiert hat.

Begriffe, die in der Literatur aufkommen, jedoch dort nicht weiter definiert werden, wurden durch Begriffssuche über Google Scholar, wissenschaftlichen Metasuchmaschine wie HSMeta oder dem OPAC-Bibliothekskatalog konkretisiert.

Zur Erstellung des Bewertungs- und Vergleichsverfahrens von API-Management-Plattformen, wurden Methoden aus der Erstellung von Nutzwertanalysen angewandt. Dazu wurden zwei Quellen zu dem Thema identifiziert, aus denen passende Methoden für das Vorgehen extrahiert wurden. Zum einen „*Scoring und Nutzwertanalysen*“ von Jörg B. Kühnapfel (Kühnapfel, 2021) zum anderen „*Nutzwertanalyse: eine Methode zur Bewertung von Lösungsalternativen und zur Entscheidungsfindung*“ von Nobert Herbig (Herbig, 2016).

Das erstellte Bewertungs- und Vergleichsverfahren wurde exemplarisch auf die Heraeus Holding GmbH angewandt. Während der Umsetzung des Bewertungs- und Vergleichsverfahrens mussten verschiedene Bewertungskriterien entsprechend den Anforderungen der Heraeus Holding GmbH in ihrer Bedeutung gewichtet werden. Für diese Gewichtung wurden firmeninterne Stakeholder miteinbezogen. Die Stakeholder wurden dabei von dem Bereichsleiter der Abteilung „Applications“ der Heraeus Consulting & IT Solutions GmbH (HCS), dem internen IT-Dienstleister der Heraeus Holding GmbH, nominiert.

2 Ausgewählte Grundlagen APIs

In diesem Kapitel werden ausgewählte Grundlagen zu APIs, insbesondere Web-APIs, beschrieben. Dazu werden Web-APIs von anderen Arten von APIs abgegrenzt. Es wird beschrieben, welche technischen Eigenschaften APIs aufweisen. Auf Basis dieser Eigenschaften wird analysiert, welche betriebswirtschaftlichen Potentiale durch die Verwendung von Web-APIs in Unternehmen entstehen können.

2.1 Funktionsweise einer (Web-)API

Jacobson et al. definieren eine API als einen Weg über den zwei Computerapplikationen über das Netzwerk in einer von beiden Parteien verständlichen Sprache kommunizieren können (Jacobson et al., 2011, S. 1–11). Diese Definition ist anwendbar auf sogenannte Web-APIs (Frank et al., 2021, S. 34). Web-APIs folgen einer Client-Server-Architektur (Masse, 2011). Der Client kann durch Anfragen auf die Daten und Funktionalitäten der vom Server bereitgestellten Anwendung zugreifen (vgl. Abbildung 1).

Abbildung 1 Client Server Modell Web API in Anlehnung an (Masse, 2011, S. 1–11)

Für diese Kommunikation wird der Web Standard des Hypertext Transfer Protocol (HTTP) verwendet (Masse, 2011, S. 1–11). HTTP ist ein Request-Response-Protokoll (Nielsen et al., 1999, S. 12). Demnach sendet der Client eine Anfrage an den Server, welche vom Client bereitgestellte Informationen beinhaltet. Der Server verarbeitet diese Informationen und sendet dem Client nach Abschluss der Verarbeitung eine Antwort (Nielsen et al., 1999, S. 12). Bei Web-APIs ist das Format der ausgetauschten Informationen klar definiert. Dazu werden Datenformate wie beispielsweise die Javascript Object Notation (JSON) verwendet (Frank et al., 2021, S. 20). JSON ist ein Textformat für die Serialisierung von strukturierten Daten (Bray, 2014). Die serialisierten strukturierten Daten können über das HTTP-Protokoll zwischen Client und Server ausgetauscht werden. Ein Beispiel für ein serialisiertes JSON-Objekt ist in Abbildung 2 dargestellt. Eine Alternative zu JSON ist beispielsweise die Extensible Markup Language (XML).

Abbildung 2 Beispiel JSON-Objekt

Wie bereits beschrieben muss die Struktur der übertragenen Daten klar definiert sein, damit Client bzw. Server sie korrekt verarbeiten können. Nach Jacobson et al. gleicht eine Web-API somit einem Vertrag zwischen dem Server als API-Bereitsteller und dem Client als API-Konsument (Jacobson et al., 2011, S. 1–11). Zum Definieren dieser Verträge gibt es verschiedene Standards. Ein häufig verwendeter Standard für Web-APIs sind Representational State Transfer-APIs (Rest-APIs) (Frank et al., 2021, S. 35). Rest-APIs verwenden Uniform Resource Identifiers (URI), um Ressourcen eindeutig zu identifizieren und zu adressieren (Masse, 2011, S. 11–23). Ein Beispiel für eine URI ist in Abbildung 3 dargestellt. Über Anfragen an die URI einer Resource können API-Konsumenten den Status einer Ressource abrufen oder diese ändern (Masse, 2011, S. 1–11).

```
http://api.example.restapi.org/france/paris/louvre/leonardo-da-vinci/mona-lisa
```

Abbildung 3 Beispiel einer URI einer Rest-API in Anlehnung an (Masse, 2011, S. 11–23)

Rest-APIs folgen weitergehend dem Prinzip Hypermedia as the engine of application state (HATEOAS) (Masse, 2011, S. 1–11). HATEOAS beschreibt, dass die Repräsentation einer Ressource die URIs von verwandten Ressourcen beinhaltet. API-Konsumenten können demnach über die, in den Antworten bereitgestellten, URIs zu verwandten Ressourcen navigieren. Ein Beispiel für die Umsetzung von HATEOAS ist in Abbildung 4 dargestellt.

```
{
    "Vorname": "Max",
    "Nachname": "Mustermann",
    "Geburtsdatum": "01.01.1970",
    "Verfasste_Bücher": [
        "/books/musterbuch_1",
        "/books/musterbuch_2"
    ],
    "Aktionen": {
        "Folgen": "/autoren/max_mustermann/folgen",
        "Liken": "/autoren/max_mustermann/liken"
    }
}
```

Abbildung 4 Beispiel für HATEOAS-Prinzip von Rest-APIs

Neben Rest, als dem weitesten verbreiteten Standard, existieren weitere API-Standards, die in den letzten 15 Jahren an Beliebtheit dazugewonnen bzw. abgenommen haben (vgl. Abbildung 5)

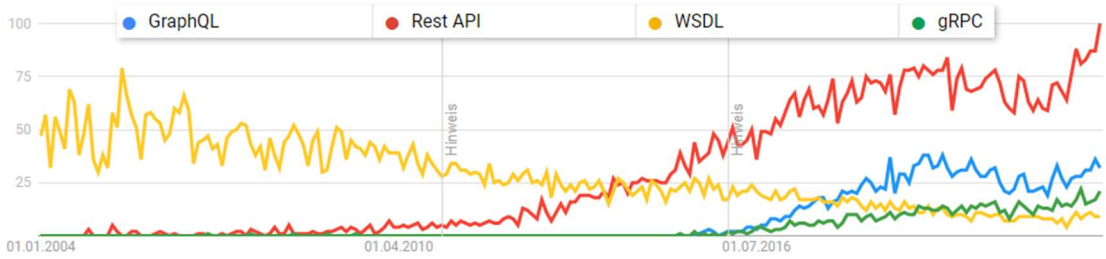

Abbildung 5 Google Suchanfragen zu API-Standards. In Anlehnung an (Weir, 2019, S. 164)

Die Standards unterscheiden sich in verschiedenen Kriterien wie beispielsweise Umsetzung von Versionierung oder Performance (Weir, 2019, S. 194–203). Dabei hat jede Lösung ihre Vor- und Nachteile. Eine Allzwecklösung gibt es nach Weir nicht (Weir, 2019, S. 194). Eine genaue Differenzierung der einzelnen Standards liegt aufgrund der angestrebten Kürze außerhalb des Umfangs dieser Arbeit.

2.2 Abgrenzung von Web-APIs zu anderen API-Arten

Auch außerhalb des Kontexts von Web-APIs wird der Begriff API verwendet. Die erste Benutzung des Begriffs API kann auf eine Publikation aus dem Jahr 1968 zurückgeführt werden (Weir, 2019, S. 158). Die Publikation beschreibt ein Vorgehen zum Ansteuern von Computerdisplays über eine hardwareunabhängige Schnittstelle (Cotton & Greatorex, 1968, S. 543). Auch heute wird das Vorgehen, Hardwarekomponenten durch APIs zu abstrahieren, eingesetzt. Als Beispiel bietet das mobile Betriebssystem Android seinen App-Entwicklern verschiedene APIs, beispielsweise zum Steuern der Kamera oder des Vibrationsmotors, an (Android API

reference, o. J.). Entwickler können so Software für das Android Betriebssystem entwickeln ohne genaue Spezifikationen der verbauten Hardwarekomponenten zu kennen. Die Kommunikation läuft in diesem Fall anders als bei Web-APIs nicht über das Netzwerk, sondern fungiert als Abstrahierung der Hardwarekomponenten. In dieser Arbeit wird sich fortlaufend nur noch auf Web-APIs bezogen. Falls nicht gegenteilig beschrieben, wird der Term API von nun an mit dem der Web-API gleichgesetzt.

2.3 Eigenschaften von APIs

Nach Frank et al. (2021) weisen APIs die Eigenschaften der Modularität, Interoperabilität und Kapselung auf (Frank et al., 2021, S. 24).

Modularität beschreibt eine Eigenschaft, bei der ein System in einzelne Module aufgeteilt werden kann. Nach James Hannan (1987) wird ein Modul als *„… eine in sich geschlossene Einheit …, die eine bestimmte Aufgabe oder eine Anzahl von Aufgaben erfüllt, um die Arbeit des gesamten Systems zu unterstützen*“ (Hannan, 1987, S. 63) definiert. Durch die feste Definition von Modulschnittstellen, können die einzelnen Module unabhängig voneinander entwickelt, getestet und verändert werden (Hannan, 1987, S. 63).

Interoperabilität beschreibt die *„… Fähigkeit eines Systems, eine Integration mit anderen gegenwärtigen oder zukünftigen Systemen einschränkungsfrei zu schaffen. Die Schnittstellen des Systems werden dabei vollständig für andere Entwickler offengelegt“* (Frank et al., 2021, S. 32). Solange die Schnittstelle klar definiert ist können auch eigene Systeme mit APIs von externen Partnern kombiniert werden (Frank et al., 2021, S. 32). Erleichtert wird diese Interoperabilität durch das Befolgen von verbreiteten Standards. In Kapitel 2.1 wurde dazu der Standard der Rest-APIs bereits erläutert.

Durch Kapselung wird jede API zu einer Blackbox: *„Jede API kann genutzt werden, ohne die genauen Inhalte des Bausteins zu kennen*“ (Frank et al., 2021, S. 37). Verwendete Technologien und Details über die Verarbeitung der Anfragen bleiben für den API-Konsumenten unbekannt, was Auswirkungen durch Änderungen in der Anwendung hinter der API für den API-Konsumenten gering hält (Frank et al., 2021, S. 37).

APIs bieten durch diese Eigenschaften die Möglichkeit komplexe Softwaresysteme in weniger komplexe einzelne Module zu unterteilen und diese beliebig miteinander zu verbinden (Frank

et al., 2021, S. 26). Jedes Modul erfüllt eine spezielle Aufgabe und kann unabhängig voneinander entwickelt und getestet werden. Durch Befolgen von Standards bei der Definition der Schnittstelle wird die Anbindung und das Auswechseln von Modulen ermöglicht. Da Informationen über die Schnittstellendefinition hinaus dem Verwender nicht bekannt sind, können keine Abhängigkeiten zu internen Spezifika der Anwendung hinter der API entstehen. Die einzelnen Module können somit intern verändert werden ohne API-Konsumenten zu beeinflussen. Aus diesen Eigenschaften von APIs können Potentiale für Unternehmen entstehen.

2.4 Betriebswirtschaftliche Potentiale durch den Einsatz von APIs

Durch die in Kapitel 2.3 beschriebenen Eigenschaften entstehen durch den Einsatz von APIs betriebswirtschaftliche Potentiale für Unternehmen. Als betriebswirtschaftliche Potentiale werden Aspekte definiert durch die APIs einen Beitrag zur langfristigen Gewinnmaximierung des Unternehmens leisten können. Die entstehenden Potentiale werden nachfolgend beschrieben.

Durch die Modularität von APIs lassen sich Funktionen von Anwendungssystemen wiederverwenden (Frank et al., 2021, S. 50 f.). Die Entwicklungszeit von neuen Systemen kann durch die Wiederverwendung dieser Funktionen beschleunigt werden (Frank et al., 2021, S. 51). Diese Beschleunigung der Entwicklungszeit ermöglicht es Unternehmen kostengünstiger Prototypen zu Produktideen zu implementieren und diese Prototypen schneller in Testszenarien zu validieren (Frank et al., 2021, S. 51). Durch die Interoperabilität können auch unternehmensfremde APIs in die eigenen Anwendungssysteme integriert werden (Frank et al., 2021, S. 32). Unternehmensfremde APIs werden dabei von Partnerunternehmen oder von externen Quellen öffentlich zugänglich am Markt angeboten (vgl. (*ProgrammableWeb*, 2022)). Das Angebot an integrierbaren Funktionen wächst dadurch.

Unternehmen können auch selbst beginnen ihre APIs ihren Partner oder der Öffentlichkeit zugänglich am Markt anzubieten. Die Vorteile des anbietenden Unternehmens resultieren in indirekten oder direkten Umsatzsteigerungen (Weir, 2019, S. 101).

Indirekte Umsatzsteigerungen entstehen beispielsweise, wenn durch die APIs neue Vertriebskanäle für die eigenen Produkte geschaffen werden. In diesem Fall wird die Nutzung der API nicht selbst in Rechnung gestellt, sondern bietet Zugriff, Produkte und Dienstleistungen des Unternehmens zu erwerben (Weir, 2019, S. 101). Bei dem Transaction Fee Modell wird die

Nutzung der API ebenfalls nicht direkt in Rechnung gestellt. Stattdessen behält der API-Anbieter bei monetären Transaktionen einen Teil der Transaktionsgebühr ein. Dieses Vorgehen ist besonders bei Zahlungsdienstleistern wie PayPal oder Stripe üblich (Frank et al., 2021, S. 106; Weir, 2019, S. 101).

Direkte Umsatzsteigerungen entstehen, wenn Abrechnungsmodelle wie beispielsweise das Tiered oder Pay-as-you-go Modell verwendet werden. Bei dem Tiered Modell können verschiede Leistungs- und Preislevel gebucht werden, die sich beispielsweise nach Anzahl der maximalen Aufrufe pro Zeitperiode, Performance oder Supportlevel unterscheiden. Bei dem Pay-as-you-go Modell wird nach der Anzahl der getätigten Anfragen an die API gezahlt (Frank et al., 2021, S. 106; Weir, 2019, S. 101). Über ein Freemium Modell kann eine limitierte Anzahl an Zugriffen oder limitierter Zugriff auf die Funktionen der API kostenlos angeboten werden (Weir, 2019, S. 101). Die Abrechnungsmodelle können verwendet werden, um bei der Bereitstellung von internen APIs an eigene Entwickler eine Verrechnung über Kostenstellen hinweg zu erlauben (Weir, 2019, S. 101).

Ob und welches Abrechnungsmodell für eine bestimmte API anwendbar ist, muss einzeln bewertet werden. Diese Bewertung liegt außerhalb des Umfangs dieser Arbeit.

3 Definition API-Management und Implikationen für Unternehmen

Mathijssen et al. schlagen in ihrer systematischen Literatur-Recherche eine Definition für den Begriff API-Management vor:

„*API Management is an activity that enables organizations to design, publish and deploy their APIs for (external) developers to consume. API Management capabilities such as controlling API lifecycles, access and authentication to APIs, monitoring, throttling and analyzing API usage, as well as providing security and documentation are often implemented through an integrated platform, which is supported by an API gateway.*“ (Mathijssen et al., 2020, S. 11)

Bestandteile des API-Managements sind nach Mathijssen et al. die Fähigkeiten des API-Managements (engl. API Management capabilities). Fähigkeiten des API-Managements werden definiert als „[…] *the ability to achieve a certain goal related to API Management […]*“ (Mathijssen et al., 2020, S. 12). Bei der Definition der Fähigkeiten des API-Managements tritt eine tautologische Verwendung des Begriffes API-Managements auf. Während sich die Definition von API-Management nur auf die Gestaltung, Veröffentlichung und Bereitstellung von APIs bezieht, sprechen die Fähigkeiten des API-Managements von weitreichenderen Themengebieten wie Zugriffsverwaltung oder Analyse der API-Verwendung. Damit ist die Definition des API-Managements enger gefasst als die beschriebenen Fähigkeiten des API-Managements. Außerdem bezieht sich die Definition nur auf die Verwaltung von selbst entwickelten APIs. Wie in Kapitel 2.4 beschrieben können Unternehmen auch unternehmensfremde APIs in die eigenen Softwareprodukte integrieren. Daher müssen manche Fähigkeiten des API Managements auch für die extern integrierten APIs berücksichtigt werden (vgl. Weir, 2019, S. 56–60). Weitergehend beschränkt sich API-Management nicht auf die Betrachtung einzelner APIs, sondern versucht ein Level der Konsistenz in der Verwaltung aller APIs eines Unternehmens herzustellen (Medjaoui et al., 2021, S. 7). Medjaoui et al. sprechen in diesem Fall von der API-Landschaft eines Unternehmens (Medjaoui et al., 2018, S. 2).

3.1 Fokusbereiche des API-Managements

In einer auf der systematischen Literatur-Recherche von Mathijssen et al. aufbauenden Arbeit der gleichen Autoren werden 20 Fähigkeiten des API-Managements definiert und diese in sechs Fokusbereiche gruppiert (Mathijssen et al., 2022, S. 7). Die sechs Fokusbereiche werden

unterteilt in API-Lifecycle-Management, Security, Performance, Monitoring, Community und Commercial (vgl. Mathijssen et al., 2022, S. 7). Jedem Fokusbereich sind mehrere Fähigkeiten des API-Managements zugeordnet. Fokusbereiche gruppieren demnach Fähigkeiten des API-Managements in übergeordnete Kategorien. Fähigkeiten des API-Managements werden durch API-Management-Praktiken implementiert (Mathijssen et al., 2022, S. 7). API-Management-Praktiken sind konkrete Maßnahmen, die in Bezug auf die einzelnen Fähigkeiten des API-Managements umgesetzt werden können.

Fokusbereich	Fähigkeit des API-Managements	API-Management-Praktik	Beschreibung
Security	Sicherheitslücken erkennen und schließen	Implementierung von Listen mit erlaubten und verbotenen IP-Adressen	Die Organisation erlaubt/blockiert Zugriff von bestimmten IP-Adressen auf die eigenen APIs
		Implementierung von Schutzmaßnahmen vor Injektionsbedrohungen	Die Organisation schützt ihre APIs vor Injektionsbedrohungen wie SQL-Injektion
		Implementierung von Schutzmaßnahmen vor DoS-Attacken	Die Organisation schützt ihre APIs vor DoS-Attacken
		Implementierung von Ablaufprotokollen bei Sicherheitsverletzungen	Die Organisation automatisiert das Verfahren beim Auftreten von Sicherheitsverletzen beispielsweise durch automatische Benachrichtigung der Verantwortlichen
		Durchführen von Sicherheitsüberprüfungen	Die Organisation fordert und überprüft Sicherheitsstandards von ihren API-Konsumenten
		Implementierung von Zero Trust Network Access (ZTNA)	Die Organisation hat eine (ZTNA)-Sicherheitsarchitektur implementiert, bei der nur der Datenverkehr von authentifizierten Benutzern, Geräten und Anwendungen zu den bereitgestellten APIs erlaubt wird

Tabelle 1 API-Management-Praktiken der Fähigkeit des API-Management Sicherheitslücken zu erkennen und zu schließen in Anlehnung an (Mathijssen et al., 2022, S. 7)

API-Management-Praktiken sind somit präziser definiert als die Fähigkeiten des API-Managements. Um dieses Konzept zu verdeutlichen, wird die Fähigkeit des API-Managements Sicherheitslücken zu erkennen und zu schließen aus dem Fokusbereich Security betrachtet (vgl. Tabelle 1). Sicherheitslücken können mit verschiedenen Ursprüngen und resultierende Gefahren verbunden sein (vgl. OWASP Foundation, o. J.). Nach Weir sollten die Risiken, die durch Sicherheitslücken entstehen können, durch entsprechende Maßnahmen mitigiert werden (Weir, 2019, S. 105). Eine implementierbare Praktik für diese Fähigkeit des API-Managements ist es, Maßnahmen zum Schutz vor Denial of Service Attacken (DoS-Attacken) zu implementieren (Mathijssen et al., 2022, S. 7). Bei DoS-Attacken wird ein Dienst solange gezielt vielen Anfragen ausgesetzt bis dieser durch Überlastung zusammenbricht (Bundesamt für Sicherheit in der Informationstechnik, o. J.). Die API-Management-Praktik kann umgesetzt werden, indem ein Limit der maximalen Anfragen pro API-Konsument und Zeitperiode eingerichtet wird (Gadge & Kotwani, 2018, S. 6). Ein API-Konsument kann beispielsweise durch den Ursprung der Anfrage (beispielsweise die IP-Adresse) identifiziert werden (Gadge & Kotwani, 2018, S. 6).

Anhand dieses Beispiels verdeutlicht Abbildung 6 wie die Begriffe API-Management, Fokusbereich, Fähigkeit des API-Managements sowie API-Management-Praktiken zueinander im Verhältnis stehen.

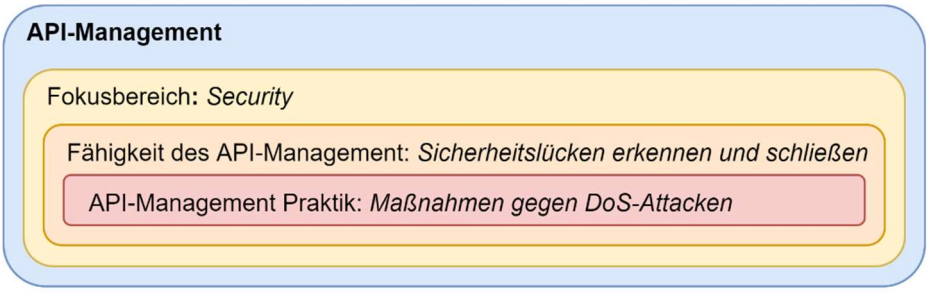

Abbildung 6 Begriffsdefinitionen API-Management aus Sicht der Praktik „Implementierung von Schutzmaßnahmen vor DoS Attacken“

Insgesamt werden den 20 Fähigkeiten des API-Managements 80 implementierbare Praktiken zugeordnet (Mathijssen et al., 2022, S. 7). Eine Auflistung der Fokusbereiche inklusive Fähigkeiten des API-Managements visualisiert die Tabelle 2. Die Beschreibung der einzelnen API-Management-Praktiken werden von Mathijssen et al. in einer weiteren Ausarbeitung definiert (Mathijssen et al., 2021, S. 9–22). In dieser Arbeit wird sich auf die Beschreibungen dieser Ausarbeitung bezogen. Eine Tabelle der API-Management-Praktiken inklusive der Beschreibungen ist in Anhang 1 zu finden.

Fokusbereich	Fähigkeit des API-Managements
API-Lifecycle-Management	**1** Versionsverwaltung **2** Versenden von Änderungsbenachrichtigungen **3** Entkopplung API & Anwendung
Security	**1** Authentifizierung **2** Autorisierung **3** Sicherheitslücken erkennen und schließen **4** Verschlüsselung
Performance	**1** Ressourcenverwaltung **2** Traffic-Management
Observability	**1** Monitoring **2** Logging **3** Analyse
Community	**1** Onboarding für Entwickler **2** Entwickler-Unterstützung **3** Dokumentation **4** Community-Management

	5 Portfolio-Management
Commercial	**1** Service-Level-Agreements **2** Monetarisierungsmodelle **3** Account-Management

Tabelle 2 Fähigkeiten des API-Managements nach Fokusbereichen in Anlehnung an (Mathijssen et al., 2022, S. 7)

3.1.1 Fokusbereich API-Lifecycle-Management

Nach Brajesh bietet API-Lifecycle-Management die Möglichkeit die Erstellung und Veröffentlichung von APIs zu kontrollieren (Brajesh, 2017, S. 15–29). Mathijssen et al. unterteilen API-Lifecycle-Management in die drei Fähigkeiten des API-Managements: (1) Versionsverwaltung, (2) Versenden von Änderungsbenachrichtigungen sowie die (3) Entkopplung zwischen API & Anwendung (Mathijssen et al., 2022, S. 8).

(1) Da sich die Geschäftsanforderungen an APIs über die Zeit ändern können, müssen häufig verschiedene Versionen einer API verwaltet werden (Brajesh, 2017, S. 15–29). Versionsverwaltung soll bei diesem Prozess unterstützen und weitergehend das Stilllegen alter Versionen ermöglichen (Brajesh, 2017, S. 15–29).
(2) Durch Versenden von Änderungsbenachrichtigungen sollen API-Konsumenten über Änderungen an der API informiert werden (Brajesh, 2017, S. 15–29).
(3) Durch Entkopplung zwischen des Datenformats der API und der dahinterliegenden Anwendung wird verhindert, dass sich die API bei neuen Versionen der Anwendung mitändert (Brajesh, 2017, S. 105–111). Diese Entkopplung kann durch Schnittstellen Übersetzung erzeugt werden. Schnittstellen Übersetzung beschreibt das Übersetzen der eingehenden Anfrage in ein für den dahinterliegenden Service verständliches Format (Brajesh, 2017, S. 15–29). So kann beispielsweise das verwendete Datenformat der Anfrage von JSON zu XML übersetzt werden (Brajesh, 2017, S. 15–29).

3.1.2 Fokusbereich Security

Der Fokusbereich Security ist verbunden mit den Fähigkeiten des API-Managements: (1) Authentifizierung, (2) Autorisierung, (3) Sicherheitslücken zu erkennen und zu schließen sowie (4) Verschlüsselung.

(1) Durch Authentifizierung soll die Identität eines API-Konsument eindeutig bestimmt und validiert werden (Brajesh, 2017, S. 15–29). Dabei können APIs unterschiedliche

Authentifizierungs-Methoden unterstützen. Die einfachste Art ist eine Nutzernamen und Password basierte Authentifizierung (Weir, 2019, S. 103 f.).

(2) Durch die auf der Authentifizierung aufbauende Autorisierung kann kontrolliert werden ob der authentifizierte API-Konsument auf die angefragte Funktion der API zugreifen darf (Brajesh, 2017, S. 15–29).

(3) Die Fähigkeit des API-Managements Sicherheitslücken zu erkennen und zu schließen wurde bereits in Kapitel 3.1 erläutert.

(4) Verschlüsselung stellt sicher, dass der Datenfluss zwischen API-Konsument und API-Anbieter verschlüsselt ist und nicht von dritten unrechtmäßig mitgelesen werden kann (Mathijssen et al., 2021, S. 6).

3.1.3 Fokusbereich Performance

Durch Maßnahmen im Fokusbereich Performance wird die Funktionalität der API auch unter steigender Last sichergestellt. Mathijssen et al. unterteilen diesen Fokusbereich in die Fähigkeiten: (1) Ressourcenverwaltung und (2) Traffic-Management (Mathijssen et al., 2022, S. 7).

(1) Ressourcenverwaltung ermöglicht Lastverteilung, automatische Skalierung der Systeme sowie Strategien zur Ausfallsicherung (Mathijssen et al., 2022, S. 7). Durch Lastverteilung werden die Anfragen an die API über mehrere identische dahinterliegende Anwendungssysteme verteilt, die Leistung des Gesamtsystems wird dadurch erhöht (Brajesh, 2017, S. 15–29). Im Falle des Ausfalls eines Anwendungssystems werden die eingehenden Anfragen auf die verbleibenden Anwendungssysteme verteilt, was die Resilienz des Gesamtsystems erhöht. Um auf Schwankungen in der Nutzung der API reagieren zu können, kann durch automatische Skalierung die Menge der verfügbaren Ressourcen erhöht oder verringert werden (Mathijssen et al., 2020, S. 18).

(2) Traffic-Management ermöglicht es, Anfrageraten zu drosseln oder Anfragen zu priorisieren (Mathijssen et al., 2022, S. 8).

3.1.4 Fokusbereich Observability

Durch Maßnahmen im Fokusbereich Observability wird die Nutzung der API messbar und auswertbar gemacht. Unterteilt werden kann er in: (1) Monitoring, (2) Logging und (3) Analyse (Mathijssen et al., 2022, S. 7).

(1) Monitoring beschreibt dabei das Sammeln von Daten in Bezug auf die Nutzung der API (Brajesh, 2017, S. 165–171). Die Daten beinhalten u.a. Anzahl der Anfragen, Antwortgeschwindigkeit oder Fehlerrate und können von den Entwicklern als Anhaltspunkt genutzt werden, um die Leistung der API zu verbessern (Brajesh, 2017, S. 165–171).
(2) Logging beschreibt das Aufzeichnen von Verhaltensmustern von API-Konsumenten (Mathijssen et al., 2021, S. 7).
(3) Auf Basis der gesammelten Daten können Analysen durchgeführt werden, um Fehlermeldungen auszuwerten (Medjaoui et al., 2021, S. 60–65) oder Stakeholdern Antworten auf Fragen in Bezug auf die Verwendung der einzelnen APIs zu beantworten (Brajesh, 2017, S. 165–171).

3.1.5 Fokusbereich Community

Der Fokusbereich Community beinhaltet Fähigkeiten des API-Managements, um Entwicklern die Arbeit mit den eigenen APIs zu erleichtern. Fähigkeiten des API-Managements dieses Fokusbereichs sind: (1) Onboarding für Entwickler, (2) Entwickler-Unterstützung, (3) Dokumentation, (4) Community-Management und (5) Portfolio-Management (Mathijssen et al., 2022, S. 7).

(1) Onboarding für Entwickler beschreibt, dass API-Konsumenten ein einfacher Registrationsprozess bereitgestellt wird, um Zugriff auf bestimmte APIs zu erhalten (Mathijssen et al., 2021, S. 7).
(2) Entwickler-Unterstützung soll API-Konsumenten in ihrer Arbeit mit den APIs unterstützen, indem über einheitliche Kommunikationskanäle Informationen über Fehler und Probleme kommuniziert werden (Mathijssen et al., 2021, S. 7).
(3) Bereitgestellte Dokumentation soll den API-Konsumenten bei der Einarbeitung in die APIs unterstützen (Mathijssen et al., 2021, S. 7).
(4) Durch Community-Management können API-Konsumenten untereinander ihre Lernerfahrung in der Arbeit mit den APIs der API-Landschaft teilen und sich gegenseitig unterstützen. Umgesetzt werden kann dies durch ein Community Forum oder ein Entwicklerportal (Mathijssen et al., 2021, S. 7).
(5) Portfolio-Management stellt sicher, dass die verfügbaren APIs für Konsumenten durchsuchbar gemacht werden können (Mathijssen et al., 2021, S. 7).

3.1.6 Fokusbereich Commercial

Der Fokusbereich Commercial beschäftigt sich mit der Vermarktung der APIs. Die zugeordneten Fähigkeiten des API-Managements sind: (1) Verwalten von Monetarisierungsmodellen, (2) Service-Level-Agreements und (3) Account-Management.

(1) Die Nutzung der Funktionalitäten einer API kann monetarisiert werden, dabei können unterschiedliche Monetarisierungsmodelle gewählt werden (vgl. Kapitel 2.4). APIs sollten daher die Abrechnung sowie Rechnungsstellung entsprechend des gewählten Monetarisierungsmodelle implementieren (Weir, 2019, S. 101).
(2) Auf Basis der gewählten Monetarisierungsmodelle können unterschiedliche Service-Level-Agreements (SLAs) angeboten werden. SLAs sind Vereinbarungen zwischen Dienstanbieter und Kunde in Bezug auf Verpflichtungen beider Parteien (Ludwig et al., 2003). Häufig werden Vereinbarungen über minimale Verfügbarkeit oder durchschnittliche Antwortzeiten festgesetzt (Ludwig et al., 2003). Den API-Konsumenten müssen Informationen über diese SLAs bereitgestellt werden sowie deren Einhaltung an Hand von Messungen nachgewiesen werden (Mathijssen et al., 2022, S. 8).
(3) Des Weiteren brauchen API-Konsumenten die Möglichkeit ihre Rechnungsaccounts zu verwalten, um beispielsweise ihre Abonnements anzupassen (Mathijssen et al., 2022, S. 8).

3.1.7 Abgrenzung API-basierte Integration

Als weiterer Aspekt des API-Managements könnte die API-basierte Integration betrachtet werden. Der Begriff basiert auf einer Arbeit aus dem Jahr 2021, in dem die englische Bezeichnung API-led Connectivity verwendet wird (Chouhan & Mishra, 2021, S. 1363). Für die Definition der API-basierten Integration wird sich auf das von Brajesh beschriebene API-Facade-Pattern bezogen (Brajesh, 2017, S. 81–105). Das API-Facade-Pattern beschreibt ein Vorgehen, in dem eine neue API zur Integration verschiedener Systeme erstellt wird. Die neue API übersetzt die eingehende Anfrage in mehrere einzelne Anfragen an die dahinterliegenden Systeme und bietet eine aggregierte Antwort (Brajesh, 2017, S. 81–105).

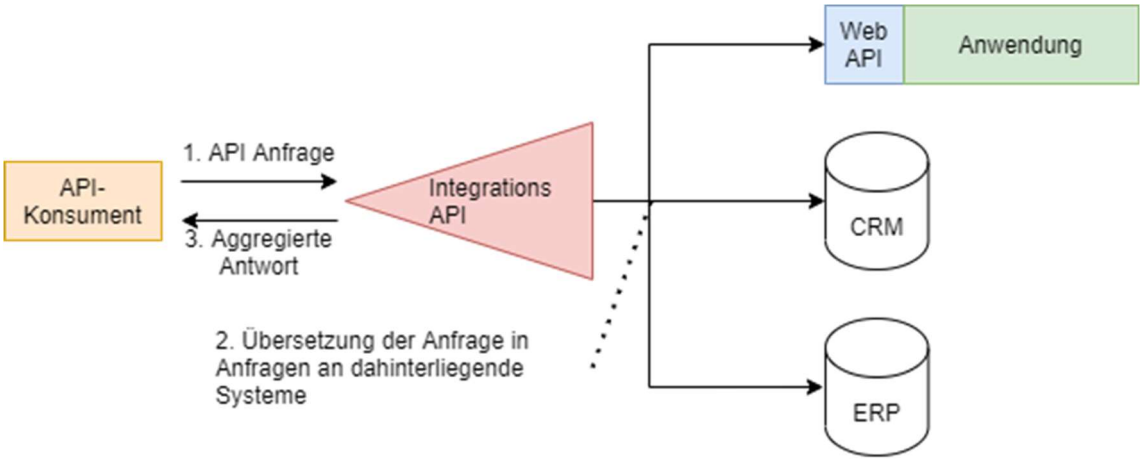

Abbildung 7 API-basierte Integration in Anlehnung an (Brajesh, 2017, S. 81–105)

Dabei müssen die dahinterliegenden Systeme selbst keine HTTP-basierten Web-APIs sein. Je nach Funktionsumfang der Integrations-API kann auch direkt auf die Datenbanken der Systeme zugegriffen werden (Preibisch, 2018, S. 137–138). Nach Brajesh ermöglicht dieses Entwurfsmuster die einfache Erstellung von APIs für komplexe Bestandssysteme die bisher noch keine APIs anbieten (Brajesh, 2017, S. 81–105). Für Unternehmen mit solchen Bestandssystemen kann dieses Entwurfsmuster ein relevanter Aspekt für die Planung und Verwaltung der eigenen API-Landschaft sein.

Diesem Fokusbereich kann die Fähigkeit des API-Managements „Dienst Orchestration“ (engl. Service Orchestration) zugewiesen werden (Mathijssen et al., 2020, S. 18). Dazu wurde sich auf eine Fähigkeit des API-Managements bezogen, die Mathijssen et al. in ihrer systematischen Literatur-Recherche aus 2020 identifiziert haben, jedoch in ihrer aufbauenden Arbeit im Jahr 2022 keinem Fokusbereich zugewiesen haben.

Dienst Orchestration beschreibt die Übersetzung einer API-Anfrage in mehrere Anfragen an dahinerliegende Systeme und die Aggregation der erhaltenden Antworten (Brajesh, 2017, S. 81–105). Die Integration wird durch eine Integrations-API umgesetzt (vgl. Abbildung 7).

Es kann diskutiert werden, ob API-basierte Integration als eigener Fokusbereich des API-Management definiert werden sollte. Anders als bei den anderen Fokusbereichen bezieht sich API-basierte Integration auf die Implementierung von Geschäftslogik durch Integration verschiedener dahinerliegender Services. Die von Mathijssen et al. (2022) beschriebenen Fokusbereiche fokussieren sich dagegen nur auf die API-Schicht, weitestgehend unabhängig der im-

plementierten Geschäftslogik. Aus diesem Grund sollte API-basierte Integration nicht direkt dem API-Management zugewiesen werden.

3.2 Implikationen für Unternehmen

Durch die Betrachtung der Funktionen des API-Managements aus den sechs Fokusbereichen konnte der Umfang von API-Management geschärft werden.

Durch Betrachtung welche API-Management-Praktiken ein Unternehmen in Verbindung mit einzelnen Fähigkeiten des API-Managements implementiert hat kann eine Aussage über den API-Management-Reifegrads des Unternehmens in den einzelnen Fokusbereichen, sowie des API-Managements ganzheitlich getroffen werden (Mathijssen et al., 2022, S. 11). In der Praxis fehlen nach Medjaoui et al. (2021) häufig die Ressourcen um alle API-Management-Praktiken in allen Fokusbereichen für jede API zu implementieren (vgl. Medjaoui et al., 2021, S. 7 f.). Unternehmen die planen APIs weitgehend in die eigene Strategie zu integrieren, sollten überlegen wie sie die einzelnen Fokusbereiche und Fähigkeiten des API-Managements auf Basis ihrer eigenen Anforderungen gewichten. Aus dieser Gewichtung kann eine Priorisierung für die Implementierung von API-Management-Praktiken abgeleitet werden.

Medjaoui et al. weisen darauf hin, dass die Implementierung von API-Management-Praktiken nicht im Umfang einzelner APIs, sondern im Kontext aller APIs eines Unternehmens betrachten werden sollte (Medjaoui et al., 2021, S. 7). Eine Herausforderung ist das Schaffen von Konsistenzen zwischen den einzelnen APIs (Medjaoui et al., 2021, S. 7). Bei wachsender Anzahl an APIs und dadurch steigender Komplexität kann die konsistente Implementierung von API-Management-Praktiken schwieriger werden (Medjaoui et al., 2021, S. 7). Das Schaffen und Gewährleisten von Konsistenzen zwischen den implementierten API-Management-Praktiken sollte demnach von Unternehmen berücksichtigt werden. Medjaoui et al. sprechen davon, dass diese Herausforderung häufig durch das Bereitstellen von Kernservices und geteilter Infrastruktur gelöst wird, die den einzelnen API-Entwicklungsteams Zugriff auf wiederverwendbare Lösungen zu verschiedenen Fokusbereichen bieten (Medjaoui et al., 2021, S. 7). Ein Lösungsansatz ist das Bereitstellen von API-Management-Plattformen, deren Architektur und Funktionsweise im nächsten Kapitel näher beschrieben werden.

4 Architektur und Funktionsweise von API-Management-Plattformen

API-Management-Plattformen sind eigenständige Softwaresysteme, die von verschiedenen Anbietern mit verschiedenen Funktionsumfängen offeriert werden. In Kapitel 3.1 wurden Fähigkeiten des API-Managements nach Fokusbereichen gruppiert beschrieben. Fähigkeiten des API-Managements können durch API-Management-Praktiken implementiert werden. API-Management-Plattformen können diese API-Management-Praktiken auf einer integrierten Plattform implementieren (Mathijssen et al., 2020, S. 11). Sie konsolidieren demnach die zu implementierenden API-Management-Praktiken mehrerer APIs an einem zentralen Ort. Nach Brajesh unterteilt sich eine API-Management-Plattform in drei Komponenten: Ein API-Gateway, ein Entwicklerportal und ein Analyse Service (vgl. Abbildung 8) (Brajesh, 2017, S. 15–29).

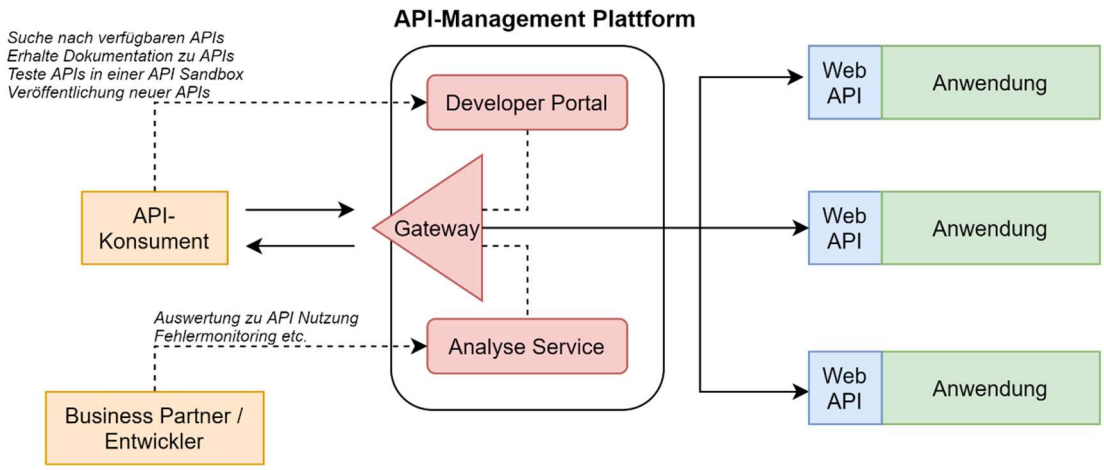

Abbildung 8 Architektur von API-Management-Plattformen in Anlehnung an (Brajesh, 2017, S. 15–29).

Eine Definition eines API-Gateways stammt von Brajesh:

”*An API gateway forms the heart of any API management solution that enables secure, flexible, and reliable communication between the back-end services and digital apps. It helps to expose, secure, and manage back-end data and services as RESTful APIs. It provides a framework to create a facade in front of the back-end services. This facade intercepts the API requests to enforce security, validate data, transform messages, throttle traffic, and finally route it to the back-end service...*” (Brajesh, 2017, S. 15–29)

API-Gateways sind demnach ein Bindeglied zwischen API-Konsumenten und mehreren APIs, die über das Gateway erreichbar sind. Anfragen an die APIs werden von dem API-Gateway abgefangen, bearbeitet und schließlich an die API weitergeleitet. Dadurch bietet das API-Gateway einen *„single point of entry"* (Brajesh, 2017, S. 81–105) zu allen bereitgestellten APIs. Das API-Gateway analysiert die eingehende Anfrage und entscheidet auf Basis von Informationen innerhalb der Anfrage (beispielsweise der URI) zu welcher dahinterliegenden API die Anfrage weitergeleitet werden muss (Weir, 2019, S. 106). Durch diese Architektur lassen sich verschiedene API-Management-Praktiken zentral innerhalb des Gateways implementieren. Zur Verdeutlichung kann sich auf das in Kapitel 3.1 beschriebene Beispiel bezogen werden. Zum Schutz vor DoS-Attacken müssten ohne ein API-Gateway entsprechende Schutzmaßnahmen in jeder API einzeln implementiert werden. In vielen API-Gateways der verschiedenen API-Management-Plattform-Anbieter sind Praktiken gegen häufig auftretende Sicherheitsschwächen, wie beispielsweise das Limitieren der maximalen Anfragen eines Nutzers in einer Zeitperiode, entweder direkt integriert oder können als Add-on hinzugefügt werden (Gadge & Kotwani, 2018, S. 6). In Kapitel 3.1.7 wurde das Konzept der API-basierten Integration vorgestellt. API-Gateways können die Funktionalität einer Integrations-API implementieren (vgl. Abbildung 7). Es existieren jedoch konträre Meinungen, ob diese Integration die Aufgabe eines API-Gateways sein sollte. Preibisch hebt die Beschleunigung der Entwicklungsgeschwindigkeit hervor, da die Integrationen nicht entwickelt, sondern in den meisten Fällen nur konfiguriert werden müssen (Preibisch, 2018, S. 138). Gadge & Kotwani behaupten stattdessen, dass Dienst Orchestration in dem API-Gateway vermieden werden sollte (Gadge & Kotwani, 2018, S. 9). Nach ihnen verletzt dieser Ansatz das Prinzip der eindeutigen Verantwortlichkeit (engl. Single-Responsibility-Prinzip). Das Prinzip der eindeutigen Verantwortlichkeit stammt aus der Softwareentwicklung und beschreibt, dass eine Komponente nur genau eine Zuständigkeit haben sollte (Abts, 2020). In Kapitel 3 wurde API-basierte Integration von den Fokusbereichen und damit den Fähigkeiten des API-Managements abgegrenzt. Daher wird auch für diese Arbeit API-basierte Integration nicht als Aufgabe des API-Gateways einer API-Management-Plattform klassifiziert.

Das Entwicklerportal ist im Kern ein Content-Management-System zur Dokumentation von APIs (Brajesh, 2017, S. 15–29). Ein Content-Management-System ist ein System zur gemeinschaftlichen Erstellung, Bearbeitung und Organisation von Inhalten (Justus-Liebig-Universität

Gießen, o. J.). Ziel des Entwicklerportals ist es Entwicklern Unterstützung bei der Arbeit mit den APIs zu liefern (Brajesh, 2017, S. 171–179). Bei Entwicklerportalen unterscheiden sich die bereitgestellten Funktionalitäten zwischen den verschiedenen Anbietern. Brajesh definiert einige Bespielfunktionalitäten eines Entwicklerportals folgendermaßen (Brajesh, 2017, S. 171–179):

- Ein API-Katalog zum Suchen und Finden veröffentlichter APIs
- Bereitstellen von Dokumentation zu den APIs
- Verwaltung von App Schlüsseln zum Steuern der API-Authentifizierung
- Ein integrierter Testbereich zum Experimentieren mit den APIs
- Codebeispiele, wie man die API aus Anwendungen heraus anspricht
- Q&A Foren, in denen sich Entwickler gegenseitig Fragen zu der Nutzung der APIs beantworten

Das Entwicklerportal und das API-Gateway sind dabei in Verbindung stehende Dienste. Änderungen im Entwicklerportal führen zu Konfigurationsänderungen im API-Gateway (Brajesh, 2017, S. 171–179). Das Entwicklerportal agiert demnach, neben seinem Zweck zum Bereitstellen von Dokumentation, als graphische Benutzeroberfläche zur Konfiguration des API-Gateways (Brajesh, 2017, S. 171–179).

Durch die Architektur des API-Gateway als single-point-of-entry eignet sich das API-Gateway zum Sammeln von Metriken und Logs über die Verwendung der APIs. Viele API-Management-Plattform-Anbieter offerieren Möglichkeiten, Daten über die Anfragen in einen Analyse Service zu speichern (Brajesh, 2017, S. 165–171). Die Auswahl, welche Daten gespeichert werden, ist dabei häufig anpassbar (Brajesh, 2017, S. 165–171). In einem Bereich zu Analyse der API-Nutzung lassen sich diese Daten in anpassbaren Berichten auswerten (Brajesh, 2017, S. 15–29).

Verfügbare Literatur zu dem Themengebiet des API-Managements wurde häufig von Autoren publiziert, die direkt oder indirekt mit kommerziellen API-Management-Plattformen in Verbindung stehen (Mathijssen et al., 2022, S. 12). Nach Mathijssen et al. sollten sich zukünftige Arbeiten auf die Fragestellung fokussieren, ob die Einführung von API-Management-Plattformen der optimale Weg sind, den API-Management-Reifegrad eines Unternehmens zu verbessern (Mathijssen et al., 2022, S. 12).

5 Funktionales Bewertungs- und Vergleichsverfahren von API-Management-Plattformen

In diesem Kapitel wird ein Verfahren für die funktionale Bewertung und den Vergleich von API-Management-Plattformen erstellt. Das erstellte Verfahren basiert auf einer Nutzwertanalyse. Nutzwertanalysen sind Verfahren in denen Alternativen mittels gewichteter Kriterien quantitativ bewertet und vergleichbar gemacht werden (Kühnapfel, 2021, S. 3).

Für die Definition des Ablaufs einer Nutzwertanalyse finden sich in der Literatur unterschiedliche Ansätze. Herbig definiert in *„Nutzwertanalyse: Eine Methode zur Bewertung von Lösungsalternativen und zur Entscheidungsfindung“* acht Schritte, die in drei Phasen unterteilt sind (vgl. Abbildung 9).

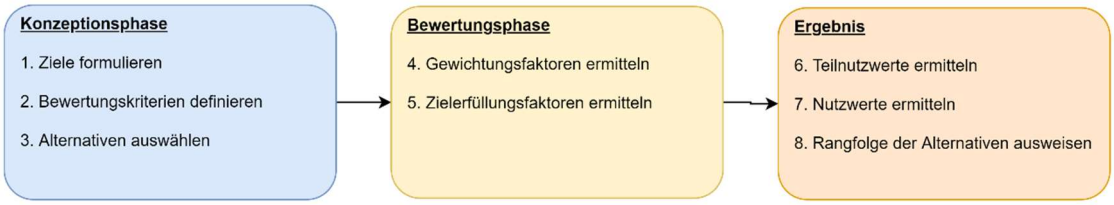

Abbildung 9 Ablauf Nutzwertanalyse in Anlehnung an (Herbig, 2016)

Kühnapfel unterteil das Vorgehen dagegen in zehn Schritte (Kühnapfel, 2021, S. 18). Beide Vorgehensmodelle überschneiden sich stark. Für diese Arbeit wird das Vorgehen von Herbig verwendet. Das Vorgehen wird mit den Erkenntnissen von Kühnapfel erweitert. Die einzelnen Prozessschritte werden dazu jeweils allgemein beschrieben und anschließend auf das Bewertungs- und Vergleichsverfahren von API-Management-Plattformen bezogen.

5.1 Ziele formulieren

Die Formulierung der Ziele beschäftigt sich mit der Frage warum eine Nutzwertanalyse durchgeführt werden soll (Kühnapfel, 2021, S. 23). Dazu muss eine einheitliche Zielformulierung definiert werden, auf die sich alle Teilnehmer der Nutzwertanalyse einigen können (Kühnapfel, 2021, S. 23). Dieses Vorgehen soll gewährleisten, dass es zu keinen Missverständnissen bei der Definition der Bewertungskriterien und Ermittlung der Zielerfüllungsfaktoren kommt. Die Zielformulierung soll präzise und mit möglichst wenigen Unterzielen oder Nebenbedingungen beschrieben sein (Kühnapfel, 2021, S. 23).

API-Management-Plattformen dienen der konsolidierten Implementierung von API-Management-Praktiken für die gesamte API-Landschaft eines Unternehmens (vgl. Kapitel 4). Um den API-Management-Reifegrad eines Unternehmens bewerten zu können, wird überprüft wie viele API-Management-Praktiken das Unternehmen, nach den einzelnen Fokusbereichen und Fähigkeiten des API-Managements gruppiert, implementiert hat (vgl. Kapitel 3.2). Der Einsatz von API-Management-Plattformen verspricht demnach einen positiven Einfluss auf den API-Management-Reifegrad eines Unternehmens.

Eine allgemeine Zielformulierung für die funktionale Bewertung und Vergleich einer API-Management-Plattform kann demnach definiert werden als:

„Welche API-Management-Plattform bietet durch ihre Funktionalitäten, unter Berücksichtigungen der funktionalen Anforderungen des Unternehmens, das größte Potential zur Verbesserung des API-Management-Reifegrades des Unternehmens?“

In dieser Arbeit wird sich rein auf die funktionalen Eigenschaften in Bezug auf implementierbare API-Management-Praktiken konzentriert. Für einen kompletten Auswahlprozess einer API-Management-Plattform müssten weitere Aspekte, die es bei der Auswahl von Softwaresystemen zu berücksichtigen gilt, in die Zielformulierung mit einbezogen werden. Unberücksichtigt bleiben die nach Maiden & Ncube beschriebenen „Vereinbarungen über technische Unterstützung“ (engl. Technical Support Arrangement) wie beispielsweise Lizenzierungsvereinbarungen und -kosten, Schulungsangebote oder Umfang des Herstellersupports (Maiden & Ncube, 1998, S. 54). Für diese Arbeit werden diese Aspekte abgegrenzt. Stattdessen werden API-Management-Plattform nur über ihre funktionalen Eigenschaften, in Bezug auf die Umsetzung von API-Management-Praktiken, miteinander verglichen.

5.2 Bewertungskriterien definieren

Bewertungskriterien müssen nach Kühnapfel vollständig, relevant, überschneidungsfrei, interdependenzfrei, bewertbar und reproduzierbar sein (Kühnapfel, 2021, S. 30–32). Vollständigkeit beschreibt dabei, dass möglichst alle Kriterien zur Erfüllung des Ziels angeführt werden (vgl. Herbig, 2016). Relevanz stellt sicher, dass jedes Kriterium zur Erfüllung der Zieldefinition beiträgt. Dabei kann die Relevanz auch an späterer Stelle, bei der Ermittlung der Gewichtungsfaktoren sichergestellt werden (vgl. Kühnapfel, 2021, S. 30). Überschneidungsfreiheit gewährleistet, dass Bewertungskriterien nicht den gleichen Inhalt abbilden (vgl. Herbig, 2016). Interde-

pendenzfrei definiert, dass die einzelnen Kriterien voneinander unabhängig sind (Kühnapfel, 2021, S. 31). Bewertbarkeit beschreibt eine möglichst objektive Bewertung der einzelnen Kriterien (Kühnapfel, 2021, S. 31). Durch die Reproduzierbarkeit des Bewertungs- und Vergleichsverfahren muss das Verfahren zur Bewertung eines Kriteriums beschreibbar und wiederholbar sein (Kühnapfel, 2021, S. 31).

Das übergeordnete Themengebiet API-Management wurde durch die Fokusbereiche möglichst vollständig unterteilt. Jeder Fokusbereich ist relevant für den API-Management-Reifegrad eines Unternehmens in Bezug auf API-Management. Für API-Management-Plattformen kann überprüft werden, welche API-Management-Praktiken, zu den Fähigkeiten des API-Managements und den Fokusbereichen zugeordnet, implementierbar sind. Wenn durch eine API-Management-Plattform für eine Fähigkeit des API-Managements mehr API-Management-Praktiken implementierbar sind als für eine andere, wird diese in der Umsetzung dieser Fähigkeit des API-Managements als besser betrachtet. Durch die Betrachtung einzelner API-Management-Praktiken können die Funktionalitäten von API-Management-Plattformen in Bezug auf die Fokusbereiche objektiv und reproduzierbar verglichen werden.

Die von Mathijssen et al. definierten Fähigkeiten des API-Managements sind dabei bereits weitestgehend überschneidungs- und interpendenzfrei. Nur bei Betrachtung der Fähigkeiten des API-Managements Authentifizierung und Autorisierung treten Interdependenzen auf, da die Autorisierung auf der vorangegangenen Authentifizierung des API-Konsumenten basiert (vgl. 3.1.2). Aus diesem Grund werden die beiden Fähigkeiten des API-Managements als ein gemeinsames Bewertungskriterium betrachtet.

Durch diese Konsolidierung entstehen auf Basis der 20 von Mathijssen et al. definierten Fähigkeiten des API-Managements 19 Bewertungskriterien, die den einzelnen Fokusbereichen zugeordnet sind. Nach Herbig wird vorgeschlagen, die Bewertungskriterien hierarchisch zu strukturieren (Herbig, 2016). Durch die hierarchische Strukturierung werden einzelne Bewertungskriterien in übergeordnete Hierarchiestufen unterteilt. Die hierarchische Strukturierung reduziert die Komplexität für die spätere Ermittlung der Gewichtungsfaktoren. Im Falle der funktionalen Bewertung von API-Management-Plattformen liegen auf übergeordneter Hierarchiestufe die sechs Fokusbereiche. Unter ihnen zugeordnet liegen die einzelnen Bewertungskriterien auf Basis der Fähigkeiten des API-Management (vgl. Tabelle 3).

Überkriterium	Unterkriterium
API-Lifecycle-Management	Versionsverwaltung
	Versenden von Änderungsbenachrichtigungen
	Entkopplung API & Anwendung
Security	Authentifizierung & Autorisierung
	Sicherheitslücken erkennen und schließen
	Verschlüsselung
Performance	Ressourcenverwaltung
	Traffic-Management
Observability	Monitoring
	Logging
	Analyse
Community	Onboarding für Entwickler
	Entwickler-Unterstützung
	Dokumentation
	Community-Management
	Portfolio-Management
Commercial	Service-Level-Agreements
	Monetarisierungsmodelle
	Account-Management

Tabelle 3 Bewertungskriterien zur funktionalen Bewertung von API-Management-Plattformen

5.3 Alternativen auswählen

Im letzten Schritt der Konzeptionsphase werden mögliche Alternativen zur weiteren Bewertung ausgewählt. Diese müssen dabei sinnvoll voneinander unterscheid- und vergleichbar sein (Kühnapfel, 2021, S. 24). Verglichen werden können Angebote die API-Management-Praktiken implementieren und der in Kapitel 4 beschriebenen Architektur von API-Management-Plattformen entsprechen oder ähneln. Der *„Gartner Magic Quadrant for Full Life Cycle API Management“* kann dabei einen Überblick über Anbieter von API-Management Lösungen geben (Pillai & Iijima, 2021). In der Beschreibung der Marktdefinition des Magic Quadrants werden die nach Brajesh beschriebenen Komponenten einer API-Management-Plattformen (Entwicklerportal, API-Gateway, Analytics Service) aufgelistet (Pillai & Iijima, 2021). Dennoch beziehen sich nicht alle Angebote auf die in Kapitel 4 beschriebene Definition einer API-Management-Plattform. Der Anbieter Postman beispielsweise hat in seiner Lösung selbst kein API-Gateway implementiert (Pillai & Iijima, 2021).

Der Magic Quadrant bewertet dabei die verschiedenen Anbieter danach, wie etabliert sie am Markt sind (engl. Ability to execute) und ihrem Verständnis für die Entwicklung des Marktes (engl. Completeness of Vision) (Pillai & Iijima, 2021). Die Anbieter werden auf Basis dessen in die vier Kategorien klassifiziert: Marktführer (engl. Leaders), Visionäre (engl. Visionaries), Nischenanbieter (engl. Niche Players) und Herausforderer (engl. Challengers) (Pillai & Iijima, 2021).

Abbildung 10 Magic Quadrant for Full Life Cycle API Management (Pillai & Iijima, 2021)

Gartner definiert die unterschiedlichen Gruppen wie folgt:

- *„Marktführer setzen ihre aktuelle Vision gut um und sind für morgen gut aufgestellt."*
- *„Visionäre verstehen, wohin der Markt geht, oder haben eine Vision für die Veränderung der Marktregeln, setzen sie aber noch nicht gut um."*
- *„Nischenanbieter konzentrieren sich erfolgreich "auf ein kleines Segment oder sind unfokussiert und übertreffen andere nicht."*
- *„Herausforderer führen heute gut aus oder dominieren vielleicht ein großes Segment, zeigen aber kein Verständnis für die Marktrichtung."* (übersetzt aus *Magic Quadrant Forschungsmethodik*, o. J.)

Um die Anzahl der Alternativen zu reduzieren, werden nach Kühnapfel Muss-Anforderungen definiert auf deren Basis Alternativen direkt zu Beginn herausgefiltert werden können (Kühnapfel, 2021, S. 25). Muss-Anforderungen können auf Grundlage der eigenen Enterprise Architektur definiert werden. Enterprise Architektur beschreibt die Prozesse nach denen die IT-Architektur eines Unternehmens entwickelt wird (Armour et al., 2012).

5.4 Gewichtungsfaktoren ermitteln

Anschließend müssen Gewichtungsfaktoren für die Bewertungskriterien ermittelt werden. Nach Herbig eignet sich dazu der paarweise Vergleich (Herbig, 2016). Bei diesem werden die einzelnen Bewertungskriterien in einer Matrix geordnet. Jedes Kriterium wird mit jedem anderen Kriterium verglichen, um zu beantworten, welches der beiden Kriterien aus Sicht der Bewertenden wichtiger ist. Je nach Antwort werden den Kriterien Punkte vergeben:

- 2:0 -> Kriterium 1 ist bedeutsamer als Kriterium 2
- 1:1 -> Beide Kriterien sind gleich wichtig
- 0:2 -> Kriterium 2 ist bedeutsamer als Kriterium 1

Aus den Ergebnissen kann eine Summe der Punkte je Bewertungskriterium errechnet werden. Setzt man diese Summe ins Verhältnis zur insgesamten Summe der verteilten Punkte errechnet sich ein Gewichtungsfaktor. Eine exemplarische Anwendung für die Fokusbereiche ist in Abbildung 11 visualisiert.

	Lifecycle-Management	Security	Performance	Observability	Community	Commcercial	**Summe**	Gewichtungsfaktor
Lifecycle-Management	*XXX*	0	0	0	1	2	**3**	0.1
Security	2	*XXX*	1	1	2	2	**8**	0.266667
Performance	2	1	*XXX*	1	2	2	**8**	0.266667
Observability	2	1	1	*XXX*	2	2	**8**	0.266667
Community	1	0	0	0	*XXX*	1	**2**	0.066667
Commcercial	0	0	0	0	1	*XXX*	**1**	0.033333
							30	
Security ist wichtiger als Lifecycle-Management								
Performance und Security sind gleich wichtig								
Lifecycle Management ist weniger wichtig als Performance								

Abbildung 11 Exemplarische Ermittlung der Gewichtungsfaktoren durch paarweisen Vergleich in Anlehnung an (Herbig, 2016)

Durch die in Kapitel 5.2 beschriebe hierarchische Strukturierung der Bewertungskriterien wird der paarweise Vergleich nur für die Überkriterien durchgeführt. Die ermittelten Gewichtungsfaktoren werden anschließend auf die unterliegenden Bewertungskriterien verteilt (vgl. Tabelle 4). Die Gewichtung der Unterkriterien kann dabei entweder gleich verteilt oder nach einem beliebigen Schlüssel erfolgen. Es gilt zu berücksichtigen, dass die Summe der Gewichtungsfaktoren des Unterkriteriums genau der Summe des Überkriteriums entspricht.

Überkriterium	Gewichtungsfaktor Überkriterium	Unterkriterium	Gewichtungsfaktor
Security	0.25	Authentifizierung & Autorisierung	0.2
		Erkennen und Schützen vor Sicherheitslücken	0.025
		Verschlüsselung	0.025

Tabelle 4 Exemplarische Ermittlung hierarchischer Gewichtungsfaktoren in Anlehnung an (Herbig, 2016)

5.5 Ermittlung der Zielerfüllungsfaktoren

Zur Ermittlung der Zielerfüllungsfaktoren bedarf es Bewertungsskalen (Kühnapfel, 2021, S. 56). Dabei gilt es, dass die Skalen bestimmte Voraussetzungen erfüllen. Praktikabilität beschreibt, dass die Bewertungsskala von den Teilnehmern in gleicher Form verständlich ist (Kühnapfel, 2021, S. 56). Repräsentativität gewährleistet, dass die Skala passende Bewertungsoptionen in Bezug der zu bewertenden Eigenschaft enthält (Kühnapfel, 2021, S. 56). Kühnapfel beschreibt beispielhaft, dass eine 10er Skala zur Bewertung eines binären Zustandes unpassend wäre (Kühnapfel, 2021, S. 56). Universalität beschreibt den Wunschzustand, dass alle Bewertungs-

skalen gleich skaliert sind (Kühnapfel, 2021, S. 57). Richtungsgleichheit legt fest, dass die Werteverläufe der Skalen gleich verlaufen (beispielsweise gut=10, schlecht=0) (Kühnapfel, 2021, S. 57).

Pro API-Management-Praktik kann vermerkt werden, ob bzw. zu welchem Grad diese durch die Plattform implementiert werden kann. Pro Fähigkeit des API-Managements können die einzelnen Angebote in Bezug auf ihre implementierbaren API-Management-Praktiken verglichen und auf Basis deren Zielerfüllungsfaktoren verteilt werden (vgl. Tabelle 5). Falls über eine API-Management-Plattform mehr API-Management-Praktiken einer Fähigkeit des API-Managements implementierbar sind, erhält sie für diese Fähigkeit des API-Managements eine bessere Bewertung als eine API-Management-Plattform die weniger API-Management-Praktiken implementieren kann. Zur Gewährleistung der Richtungsgleichheit der Skala wird vorgeschlagen, für mehr implementierbare API-Management-Praktiken einen höheren Zielerfüllungsfaktor zu vergeben.

Fähigkeit des API-Managements	API-Management Praktik	Implementierbar	Zielerfüllungsfaktor
Erkennen und Schützen von Sicherheitslücken	Implementierung von Listen mit erlaubten und verbotenen IP-Adressen	ja	3.5/6 -> **58.3/100**
	Implementierung von Richtlinien zum Schutz vor Injektionsbedrohungen	teilweise	
	Implementierung von Schutzmaßnahmen vor DoS Attacken	ja	
	Implementierung von Maßnahmen bei Sicherheitsverletzungen	nein	
	Durchführen von Sicherheitsüberprüfungen	nein	
	Implementierung von Zero Trust Network Access (ZTNA)	ja	

Tabelle 5 Direkte Ermittlung der Zielerfüllungsfaktoren durch Überprüfung der implementierbaren Praktiken

Für die Betrachtung der implementierbaren API-Management-Praktiken können verschiedene Quellen herangezogen werden. Von den Anbietern bereitgestellte Dokumentation kann nach den entsprechenden API-Management-Praktiken durchsucht werden. Es ist anzunehmen, dass genauere Eindrücke gesammelt werden können, wenn Interviews mit technischen Experten der Lösung geführt oder die implementierbaren API-Management-Praktiken in Feldtests betrachtet werden.

Im Idealfall lassen sich alle API-Management-Praktiken in vordefinierte Kategorien unterteilen. Die für dieses Vorgehen vorgeschlagenen Kategorien sind: „implementierbar“, „teilweise implementierbar“ und „nicht implementierbar“. Der Zielerfüllungsfaktor kann demnach direkt durch eine Betrachtung des Verhältnisses der von Mathijssen et al. definierten sowie den durch die API-Management-Plattform implementierbaren API-Management-Praktiken ermit-

telt werden. Dies gewährleistet die von Kühnapfel geforderte Praktikabilität. Als einheitliche Skala kann eine Zielerfüllungsskala von 0 bis 100 verwendet werden, die das prozentuale Verhältnis zwischen definierten und implementierten API-Management-Praktiken betrachtet. API-Management-Praktiken. die als teilweise implementierbar kategorisiert werden, werden mit dem Faktor 0.5 betrachtet (vgl. Tabelle 5).

Es wird jedoch vermutet, dass bei der Bewertung subjektive Eindrücke einen Einfluss haben könnten und demnach eine eindeutig kategorische Bewertung der API-Management-Praktiken nicht möglich ist. API-Management-Praktiken können bei verschiedenen Anbietern anders und somit aus Sicht der Bewertenden besser oder schlechter implementierbar sein. Um diese Aspekte zu berücksichtigen, könnte ein indirekteres Verfahren zur Ermittlung der Zielerfüllungsfaktoren umgesetzt werden. Statt jede API-Management-Praktik einzeln kategorisch zu bewerten, können die Erkenntnisse über die Implementierbarkeit der API-Management-Praktiken, einer Fähigkeit des API-Managements, zu einem aggregierten Meinungsbild zusammengefasst werden. Die Ermittlung der Zielerfüllungsfaktoren könnte in diesem Fall auf Basis eines Vergleichs des aggregierten Meinungsbildes je Alternative durchgeführt werden (vgl. Tabelle 6).

Fähigkeit des API-Managements	API-Management Praktik	Implementierbar	Zielerfüllungsfaktor
Erkennen und Schützen von Sicherheitslücken	Implementierung von Listen mit erlaubten und verbotenen IP-Adressen	ja, sehr einfach	2/5
	Implementierung von Richtlinien zum Schutz vor Injektionsbedrohungen	umständlicher als bei anderen Anbietern	
	Implementierung von Schutzmaßnahmen vor DoS Attacken	nicht vollständig, mehr Funktionen bei anderen Anbietern	
	Implementierung von Maßnahmen bei Sicherheitsverletzungen	nicht möglich	
	Durchführen von Sicherheitsüberprüfungen	nicht möglich	
	Implementierung von Zero Trust Network Access (ZTNA)	ja	

Tabelle 6 Indirekte Ermittlung der Zielerfüllungsfaktoren durch Überprüfung der implementierbaren Praktiken

Durch dieses Verfahren würde der Bewertungsprozesses jedoch subjektiver werden, was die Reproduzierbarkeit des Verfahrens gefährdet. Falls dieses Verfahren verwendet werden soll, muss eine praktikable Skala verwendet werden. Als praktikable Skala wird eine Ordinalskala zum Bilden einer Rangfolge der Alternativen vorgeschlagen (Kühnapfel, 2021, S. 57). Die Alternative, die die API-Management-Praktiken einer Fähigkeit des API-Managements im Vergleich zu den Alternativen am besten implementiert erhält den Rang eins und damit den höchsten Zielerfüllungsfaktor (beispielsweise 5/5 beim Vergleich von fünf Alternativen). Rang zwei erhält den zweithöchsten Zielerfüllungsfaktor (beispielsweise 4/5 beim Vergleich von fünf Alternativen). Teilen sich zwei oder mehr Alternativen einen Rang, wird das arithmetische Mittel zwi-

schen ihren belegten Zielerfüllungsfaktoren für beide verteilt (beispielsweise 2.5 für Rang 3 und 4 beim Vergleich von fünf Alternativen).

Das beschriebene Verfahren für die indirekte Ermittlung der Zielerfüllungsfaktoren durch Überprüfung der implementierbaren API-Management-Praktiken basiert auf der Vermutung, dass bei der Bewertung subjektive Eindrücke einen Einfluss haben könnten und demnach eine eindeutig kategorische Bewertung der API-Management-Praktiken nicht möglich ist. Um diese Vermutung zu überprüfen, wird für die in Kapitel 6 beschriebene exemplarische Anwendung versucht, die Implementierbarkeit der einzelnen API-Management-Praktiken kategorisch zu bewerten und von der kategorischen Bewertung die Zielerfüllungsfaktoren direkt durch die Berechnung des Verhältnisses aus definierten und implementierbaren API-Management-Praktiken abzuleiten.

In jedem Fall sollte sich für ein einheitliches gewähltes Verfahren entschieden werden, welches für alle Fähigkeiten des API-Managements gleich angewandt werden kann. Werden die Verfahren mir ihren unterschiedlichen Bewertungsskalen vermischt, wäre das ein Verstoß gegen die von Kühnapfel geforderte Universalität.

5.6 Ergebnisphase

Durch Ermittlung der Gewichtungs- sowie Zielerfüllungsfaktoren können anschließend Teil- sowie Gesamtnutzwerte analysiert werden (Herbig, 2016). Der Gesamtnutzwert ist dabei die Summe aller Teilnutzwerte. Die Teilnutzwerte belaufen sich dabei auf das Produkt der einzelnen Zielerfüllungs- und Gewichtungsfaktoren (vgl. Tabelle 7)

Überkriterium	Unterkriterium	Gewichtungsfaktor	Zielerfüllungsfaktor	Teilnutzen
Security	Authentifizierung & Autorisierung	0.2	50	10
	Erkennen und Schützen vor Sicherheitslücken	0.025	25	0.625
	Verschlüsselung	0.025	80	2
...				
GESAMTNUTZEN:		1		**57.3835**

Tabelle 7 Exemplarische Ermittlung der Teil- und Gesamtnutzen

Durch den Vergleich der Gesamtnutzen ergibt sich eine Rangordnung der Alternativen. In einer Sensitivitätsanalyse werden die Ergebnisse der Nutzwertanalyse durch beispielsweise Anpassung der Gewichtungsfaktoren auf ihre Robustheit überprüft (Kühnapfel, 2021, S. 81 ff.). Es wird überprüft, ob eine Anpassung der Gewichtungsfaktoren ein anderes Ergebnis in Bezug auf

das Ergebnis des Bewertungs- und Vergleichsverfahrens produziert. Sofern sich die Rangfolge der Alternativen nicht ändert, ist das Ergebnis belastbar und reproduzierbar (Kühnapfel, 2021, S. 86). Eine veränderte Reihenfolge weist auf nicht signifikante Unterschiede zwischen den Alternativen hin, was eine relevante Information für weitere Entscheidungen sein kann (Kühnapfel, 2021, S. 86).

6 Anwenden und evaluieren des Verfahrens bei der Heraeus Holding GmbH

Die Heraeus Holding GmbH ist ein weltweit agierendes Unternehmen mit über 16000 Mitarbeitern. Das Unternehmen ist in mehrere selbstständig agierende Geschäftseinheiten unterteilt, die in unterschiedlichen Märkten agieren. Die Heraeus Consulting & IT Solutions GmbH (HCS) ist der interne IT-Dienstleister und Beratungspartner rund um Digitalisierungsprojekte innerhalb des Konzerns. Der Bereich „Applications“ der HCS entwickelt und betreut Anwendungen außerhalb des SAP-Umfelds, die an interne oder externe Kunden bereitgestellt werden. Der Bereich erhält vermehrt Anfragen die Funktionalitäten der betreuten Anwendungen als API bereitzustellen. Aus diesem Grund wurde bereits die Einführung einer API-Management-Plattform diskutiert. Um den Funktionsumfang von verschiedenen Alternativen bewerten zu können wurde das in Kapitel 5 erstellte Bewertungs- und Vergleichsverfahren angewandt.

Als formuliertes Ziel wurde sich auf das in Kapitel 5.1 beschriebene Ziel zur funktionalen Bewertung und Vergleich von API-Management-Plattformen bezogen. Für die Auswahl der Bewertungskriterien wurde die in Kapitel 5.2 beschriebene hierarchische Strukturierung der Bewertungskriterien auf Basis der Fähigkeiten des API-Managements und der Fokusbereiche umgesetzt. Aus Gründen des Arbeitsaufwands der Bewertung musste die Anzahl an zu bewertenden Alternativen eingegrenzt werden. Es wurden aus dem „*Gartner Magic Quadrant for Full Life Cycle API-Management*“ die Mulesoft Anypoint Plattform und die Apigee Plattform von Google ausgewählt. Im Magic Quadrant wurden diese beide Plattformen am besten bewertet (vgl. Abbildung 10).

In einem anderen Bereich der HCS wird bereits eine Lösung von SAP mit dem Namen SAP Process Integration (SAP-PI) verwendet. Diese Lösung wird zur Integration zwischen Systemen eingesetzt und vermarktet sich selbst nicht als API-Management-Plattform. Neben dem Synchronisieren von Daten zwischen Anwendungssystemen, wird die Lösung auch zum Bereitstellen eines zentralen Zugriffspunkt für APIs von Drittsystemen eingesetzt. Dabei werden API-Management-Praktiken aus unterschiedlichen Fokusbereichen durch diese Lösung implementiert. Außerdem bietet die SAP-PI einen zentralen Zugriffspunkt zu dahinterliegenden APIs, was dem in Kapitel 4 beschriebenen Eigenschaften eines API-Gateways gleicht. Die Lösung ähnelt

demnach in ihrer Architektur und Funktionsweise einer API-Management-Plattform. Die SAP-PI wurde deshalb als zu bewertende Alternative in das Verfahren aufgenommen. Ihre Funktionalitäten wurden mit denen der dedizierten API-Management-Plattformen von Mulesoft und Google verglichen. Dedizierte API-Management-Plattformen sind dabei Angebote, die von ihrem Anbieter als API-Management-Plattformen vermarktet werden. Durch den Vergleich wird der in Kapitel 4 beschriebene Vorbehalt adressiert: API-Management-Plattformen haben noch nicht erwiesen, der optimale Weg zu sein, den API-Management-Reifegrad eines Unternehmens zu erhöhen.

Anschließend musste die Gewichtung der Bewertungskriterien festgelegt werden (vgl. Kapitel 5.4). Um verschiedene Meinungen in die Gewichtung der Bewertungskriterien zu aggregieren, wurden die Stakeholder des Unternehmens gebeten, den in Kapitel 5.4 beschriebenen paarweisen Vergleich umzusetzen. Um ein einheitliches Verständnis über die einzelnen Fokusbereiche zu gewährleisten, wurde den einzelnen Stakeholdern ein Handout bereitgestellt. Das Handout beschreibt die einzelnen Fokusbereiche und beinhaltet eine Beschreibung des Vorgehens. Das bereitgestellte Handout ist in Anhang 2 beigelegt. Es wurden fünf Stakeholder befragt, von denen vier den paarweisen Vergleich umsetzten. Die zusammengefassten Ergebnisse sind in Tabelle 8 präsentiert. Die einzelnen ausgefüllten Matrizen sind in Anhang 3 bis Anhang 6 beigelegt.

	Lifecycle-Management	Security	Performance	Observability	Community	Commcercial	**Gesamt**
Stakeholder 1	0.1	0.3	0.1667	0.2333	0.0667	0.1333	**1**
Stakeholder 2	0	0.33	0.27	0.13	0.17	0.1	**1**
Stakeholder 3	0.10	0.27	0.30	0.23	0.10	0.00	**1**
Stakeholder 4	0.13	0.33	0.17	0.23	0.13	0.00	**1**
AVG:	0.0825	0.3067	0.2267	0.2067	0.1167	0.0583	**1**

Tabelle 8 Zusammengefasste Ergebnisse Paarweiser Vergleich der Stakeholder

Jeder Stakeholder bewertete entweder den Fokusbereich Security oder Performance als am wichtigsten. Die Fokusbereiche API-Lifecycle-Management und Commercial erhielten die geringste Gewichtung. Pro Fokusbereich wurde die durchschnittliche Gewichtung analog zu dem in Kapitel 5.4 beschriebenen Vorgehen zugeordnet und gleichmäßig auf alle Fähigkeiten des API-Managements dieses Fokusbereiches verteilt.

Für die Bestimmung der Zielerfüllungskriterien wurden pro API-Management-Plattform die implementierbaren API-Management-Praktiken je Fähigkeit des API-Managements analysiert. Für alle Alternativen wurde initial das Verfahren angewandt, die Dokumentation der Anbieter nach den einzelnen API-Management-Praktiken zu durchsuchen (vgl. 5.5).

Für die SAP-PI kam es bei der Umsetzung dieses Verfahrens zu Problemen, da die öffentlich zugängliche Dokumentation nicht detailliert genug war, um die Implementierbarkeit einzelner API-Management-Praktiken zu validieren. Daher wurde die Fachabteilung gebeten, die Implementierbarkeit der API-Management-Praktiken entsprechend ihrer Kenntnisse zu bewerten. Dabei kann als problematisch angesehen werden, dass unterschiedliche Verfahren für die Kategorisierung der Implementierbarkeit der API-Management-Praktiken unterschiedliche Ergebnisse produzieren können. Der Bedarf ein anderes Verfahren anzuwenden, entstand durch die unzureichende öffentlich zugängliche Dokumentation der SAP-PI. Dies kann als externe Gegebenheit zu Problemen bei der Anwendung des Verfahrens führen.

Die einzelnen API-Management-Praktiken wurden analog zu Kapitel 5.5 als „implementierbar“, „teilweise implementierbar“ oder „nicht implementierbar“ kategorisiert. Die Bewertung, ob einzelne API-Management-Praktiken durch die API-Management-Plattform implementierbar sind, stellte sich dabei als teilweise schwierig heraus. Manche API-Management-Praktiken sind nicht direkt als explizite Funktionalität implementiert, lassen sich allerdings durch entsprechende Konfiguration und Kombination beschriebener Funktionalitäten implementieren. Die API-Management-Praktik APIs um Versionsinformationen zu erweitern (engl. Extend API with Versioning Information (Mathijssen et al., 2021, S. 11)) beschreibt beispielsweise, dass die Antworten der API um eine Warnung erweitert werden können, die den API-Konsumenten über das baldige stilllegen dieser API-Version informiert (Mathijssen et al., 2021, S. 11). Die Mulesoft Anypoint Plattform unterstützt diese Funktionalität nicht direkt, bietet jedoch die Möglichkeit an, die Antworten der API beliebig zu konfigurieren (MuleSoft Documentation, o. J.). Eine Erweiterung um eine entsprechende Versions-Warnung könnte demnach manuell

konfiguriert werden. Solche indirekt implementierbaren API-Management-Praktiken wurden mit „teilweise implementierbar“ gekennzeichnet.

Auch sind manche API-Management-Praktiken schwer auf die Funktionalität von API-Management-Plattformen übertragbar. Ein Beispiel dafür ist die API-Management-Praktik API-Evangelisten zu nominieren (engl. Dedicate Evangelist (Mathijssen et al., 2021, S. 19)). Ein API-Evangelist ist ein Ansprechpartner im Unternehmen der Feedback von API-Konsumenten sammelt und die Nutzung der APIs durch das Erstellen von Trainingsmaterialien fördert (Mathijssen et al., 2021, S. 19). Zwar kann diese API-Management-Praktik durch Komponenten der API-Management-Plattform wie dem Entwicklerportal unterstützt werden, benötigt jedoch weitere organisatorische Prozesse und die Partizipation der Mitarbeiter. Solche API-Management-Praktiken wurden, falls nicht explizit als Funktionalität beschrieben, mit „nicht implementierbar“ gekennzeichnet. Die Bewertung der Implementierbarkeit der einzelnen API-Management-Praktiken sind pro Anbieter in Anhang 7 bis Anhang 9 dargestellt.

Aus der Kategorisierung der Implementierbarkeit der API-Management-Praktiken ableitend, konnten die Zielerfüllungsfaktoren der einzelnen Fähigkeiten des API-Managements auf einer Skala von 0 bis 100 bewertet werden (vgl. Kapitel 5.5). Die Skala beschreibt dabei das prozentuale Verhältnis zwischen implementierbaren und definierten API-Management-Praktiken. Durch die Gleichgewichtung der Fähigkeiten des API-Managements pro Fokusbereich konnte der durchschnittliche Zielerfüllungsfaktor je Fokusbereich berechnet werden. In Abbildung 12 sind die durchschnittlichen Zielerfüllungsfaktoren je Fokusbereich inklusive der durch den paarweisen Vergleich errechneten Gewichtungsfaktoren dargestellt.

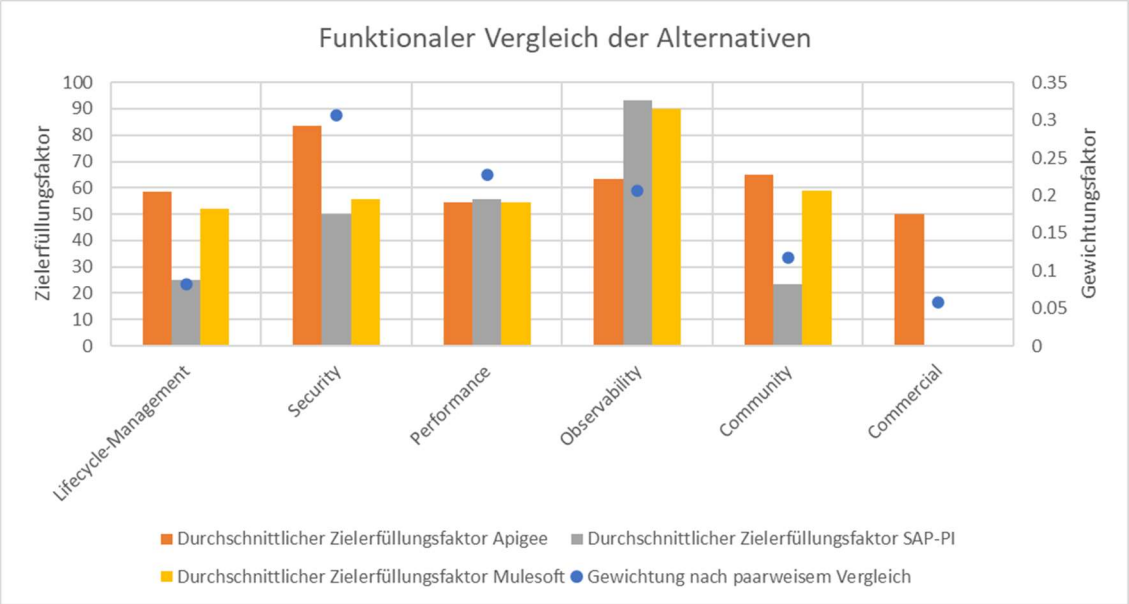

Abbildung 12 Durchschnittliche Zielerfüllungsfaktoren je Fokusbereich nach verglichenen Anbietern

Die Apigee Plattform weist in den Fokusbereichen API-Lifecycle-Management, Security, Community und Commercial die höchsten Zielerfüllungsfaktoren auf. Sie ist die einzige Plattform über die API-Management-Praktiken aus dem Fokusbereich Commercial implementierbar sind. Die guten Ergebnisse in den Bereichen API-Lifecycle-Management und Community lassen sich auf ein integrierbares und frei konfigurierbares Entwicklerportal zurückführen, über welches sich viele API-Management-Praktiken implementieren lassen (Google Cloud, o. J.).

Die SAP-PI als nicht dedizierte API-Management-Plattform weist in den Fokusbereichen Security, Performance und Observability vergleichbare Zielerfüllungsfaktoren auf. Viele API-Management-Praktiken aus den Fokusbereichen API-Lifecycle-Management, Community und Commercial sind jedoch nicht implementierbar. Dies ist darauf zurückzuführen, dass die SAP-PI kein Entwicklerportal beinhaltet.

Die Mulesoft Anypoint Plattform konnte in keinem Fokusbereich den höchsten Zielerfüllungsfaktor erreichen. Zwar bietet sie auch ein Entwicklerportal an, dieses ist jedoch nicht so frei konfigurierbar wie das der Apigee Plattform. Weder die Mulesoft Anypoint Plattform noch die SAP-PI implementieren API-Management-Praktiken aus dem Fokusbereich Commercial.

Fokusbereich	Teilnutzen Apigee	Teilnutzen SAP-PI	Teilnutzen Mulesoft
Lifecycle-Management	4.83	2.06	4.29
Security	25.56	15.34	17.07
Performance	12.36	12.58	12.36
Observability	13.09	19.29	18.6
Community	7.56	2.73	6.86
Commercial	2.92	0	0
GESAMT:	**66.32**	**52**	**59.18**

Tabelle 9 Berechnung Teil und Gesamtnutzen der Anbieter

Durch Multiplikation der Gewichtungsfaktoren (vgl. Tabelle 8) mit den Zielerfüllungsfaktoren (vgl. Abbildung 12) wurden die Teilnutzwerte je Fokusbereich berechnet. Die Teilnutzen konnten summiert werden, um den Gesamtnutzen je Plattform zu berechnen (vgl. Tabelle 9). Nach diesem Vergleich weist die Apigee Plattform den höchsten Gesamtnutzen auf. Die Mulesoft Anypoint Plattform folgt dahinter auf Platz zwei. Die SAP-PI weist den geringsten Gesamtnutzen auf. Als Sensitivitätsanalyse wurde berechnet, wie sich die Gesamtnutzen unter Annahme einer gleichen Gewichtung aller Gewichtungsfaktoren verändern (vgl. Tabelle 10). Auch bei einer gleichen Verteilung aller Gewichtungsfaktoren ändert sich die Reihenfolge der Alternativen nicht.

Fokusbereich	Teilnutzen Apigee	Teilnutzen SAP-PI	Teilnutzen Mulesoft
Lifecycle-Management	9.75	4.17	8.67
Security	13.89	8.33	9.28
Performance	9.08	9.25	9.08
Observability	10.56	15.56	15
Community	10.8	3.9	9.8
Commercial	8.33	0	0
GESAMT:	**62.41**	**41.21**	**51.83**

Tabelle 10 Berechnung Teil und Gesamtnutzen der Anbieter bei Gleichgewichtung der Fokusbereiche

Durch Anwenden des Bewertungs- und Vergleichsverfahrens wurde das Bild des Gartner Magic Quadrant bestätigt (vgl. Abbildung 10). Die im Magic Quadrant am besten bewertete Alternative Apigee konnte sowohl unter gleichgewichteter als auch unter Gewichtung der Anforderungen der Heraeus Holding GmbH, den höchsten Gesamtnutzen erzielen. Rein funktional bietet die Apigee Plattform demnach das größte Potential den API-Management-Reifegrad der Heraeus Holding GmbH in Bezug auf API-Management unter Berücksichtigung der Gewichtung der Fokusbereiche zu erhöhen. Diese Erkenntnis kann als Grundlage für weitere Entscheidungsprozesse genutzt werden. Weitere Anbieter könnten mit dem Verfahren bewertet und verglichen werden.

7 Fazit und Ausblick

In dieser Arbeit wurde die Bedeutung von APIs und API-Management für Unternehmen erläutert. Dazu wurden die Eigenschaften der Modularität, Interoperabilität und Kapselung von APIs beschrieben. Anschließend wurden die, aus diesen Eigenschaften entstehenden, betriebswirtschaftlichen Potentiale für Unternehmen analysiert. Die Potentiale belaufen sich auf die Beschleunigung von Entwicklungsprozessen, sowie die Möglichkeit Umsatzsteigerungen zu erzielen. Es wurde aufgezeigt, dass sich das Themengebiet des API-Managements in die sechs Fokusbereiche API-Lifecycle-Management, Security, Observability, Performance, Community und Commercial unterteilen lässt. Den sechs Fokusbereichen wurden 20 Fähigkeiten des API-Managements zugewiesen und diese beschrieben. Die einzelnen Fähigkeiten des API-Managements bündeln implementierbare API-Management-Praktiken (beispielsweise die Implementierung von Schutzmaßnahmen vor DoS-Attacken). Hervorgehoben wurde die Wichtigkeit, diese API-Management-Praktiken konsistent für die API-Landschaft eines Unternehmens zu implementieren. API-Management-Plattformen wurden als Möglichkeit vorgestellt, diese konsistente Implementierung zu gewährleisten.

Die Leitfrage dieser Arbeit war, wie ein Verfahren für die funktionale Bewertung und zum Vergleich von API-Management-Plattformen, unter Berücksichtigung der Anforderungen eines Unternehmens, quantifizierbar gestaltet werden kann. Ein entsprechendes Bewertungs- und Vergleichsverfahren wurde erstellt. Das Bewertungs- und Vergleichsverfahren basiert auf einer Nutzwertanalyse. Es ermöglicht die Gewichtung der einzelnen Fähigkeiten des API-Managements entsprechend der Anforderungen eines Unternehmens. Anschließend werden API-Management-Plattformen auf die Implementierbarkeit der einzelnen API-Management-Praktiken überprüft. Aus dieser Überprüfung lassen sich Zielerfüllungsfaktoren für die einzelnen Fähigkeiten des API-Managements ermitteln. Für diese Ermittlung wurden zwei Verfahren beschrieben. Das erste Verfahren leitet die Zielerfüllungsfaktoren durch eine Kategorisierung der Implementierbarkeit der einzelnen API-Management-Praktiken ab. Als Kategorien für die Implementierbarkeit wurden „implementierbar“, „teilweise implementierbar“ und „nicht implementierbar“ definiert. Das zweite Verfahren ermöglicht eine subjektivere Bewertung der Implementierbarkeit und leitet die Zielerfüllungsfaktoren anschließend durch das Bilden einer Rangordnung der verglichenen Alternativen ab. Durch Betrachtung der Gewichtung und der

Zielerfüllungsfaktoren können Nutzwerte berechnet werden. Anhand der errechneten Nutzwerte lassen sich verschieden Alternativen quantitativ bewerten und vergleichen.

Um das Bewertungs- und Vergleichsverfahren evaluieren zu können, wurde es exemplarisch bei der Heraeus Holding GmbH angewandt. Durch die Betrachtung einzelner API-Management-Praktiken ließ sich ein detailliertes Bild über funktionale Stärken und Schwächen der verglichenen API-Management-Plattformen in Bezug auf die Fähigkeiten des API-Managements zeichnen. Dadurch wurde der Vergleich der Funktionen verschiedener API-Management-Plattformen quantifizierbar und nachvollziehbar. Für die Heraeus Holding GmbH konnte so eine wichtige Grundlage für die Auswahl einer API-Management-Plattform gelegt werden. Zur Ermittlung der Zielerfüllungsfaktoren wurde das zuerst beschriebene Verfahren angewandt. API-Management-Praktiken können bei unterschiedlichen Anbietern mit unterschiedlichem Arbeitsaufwand implementierbar sein. Eine klare Einteilung der Implementierbarkeit der API-Management-Praktiken in die drei benannten Kategorien stellte sich daher teilweise als schwierig heraus. In zukünftigen Arbeiten sollte auch das zweite beschriebene Verfahren zur Ermittlung der Zielerfüllungsfaktoren angewandt werden. Die Ergebnisse der beiden Verfahren könnten so verglichen werden.

In dieser Arbeit wurden Alternativen, auf Basis ihrer Fähigkeit API-Management-Praktiken aus möglichst allen Fokusbereichen zu implementieren, miteinander verglichen. Denkbar wäre auch ein Ansatz, in dem mehrere einzelne Systeme miteinander kombiniert werden. In dieser Arbeit wurde das Bewertungs- und Vergleichsverfahren neben dedizierten API-Management-Plattformen auf ein Heraeus-Bestandssystem angewandt. Dem Heraeus-Bestandssystem fehlte es an einem Entwicklerportal als Komponente einer API-Management-Plattform. Dennoch ließen sich API-Management-Praktiken über das System implementieren. Durch die fehlende Komponente schnitt das Heraeus-Bestandssystem vergleichsweise schlecht in den Fokusbereichen API-Lifecycle-Management, Community und Commercial ab. In den Fokusbereichen Security, Performance und Observability konnte es vergleichbare Nutzwerte erzielen. Zukünftige Analysen könnten überprüfen, ob sich einzelne Systeme gut miteinander ergänzen und kombinieren lassen. Dies wäre eine denkbare Alternative zum Einsatz von API-Management-Plattformen. Zum Vergleich des Funktionsumfangs der Gesamtarchitektur kann das erstellte Bewertungs- und Vergleichsverfahren als Grundlage genommen werden.

8 Retrospektive

Diese Arbeit war die erste vollständig allein durchgeführte wissenschaftliche Arbeit des Autors in diesem Umfang. Eine Herausforderung war dabei die strikte zeitliche Begrenzung der Bearbeitungszeit auf acht Wochen. Der Autor versuchte dieses Problem anzugehen, indem er einen Zeitplan erstellte, der das Fertigstellen einzelner Kapitel als Meilensteine datierte. Dieses Verfahren wies das Problem auf, dass Kapitel auch nach den datierten Fertigstellungsterminen noch weitere kleinere Anpassungen benötigten. Dennoch hat der Zeitplan geholfen, sich während der Bearbeitungszeit nicht in einzelnen Kapiteln zu verrennen. Entstehender Zeitdruck bei der Erstellung der späteren Kapitel konnte so vermieden werden.

Durch die Arbeit wurde das methodische Vorgehen des Autors zum Erstellen von wissenschaftlichen Arbeiten gestärkt. Während des Bearbeitungsprozesses wurden verwendete Quellen hinterfragt und auf ihre argumentative Tragbarkeit bewertet. Quellen, deren Argumentationen nicht nachvollziehbar waren, wurden aus dem Literaturverzeichnis dieser Arbeit entfernt. Es wurde davon abgesehen Literatur zu referenzieren, ohne die Methodik und den Ursprung der jeweiligen Quelle zu betrachten. Weitergehend wurde der Autor in der konsistenten Definition und Verwendung von Begrifflichkeiten geschult. Dieses Vorgehen war nötig, um das erarbeitete Bewertungs- und Vergleichsverfahren von API-Management-Plattformen durch Wiederverwendung der vorher eingeführten Begriffe prägnant beschreiben zu können.

Herausforderungen traten insbesondere bei der Anwendung des erarbeiteten Verfahrens in der Praxis auf. Aus Gründen der zeitlichen Restriktion konnte kein Kontakt zu den jeweiligen Anbietern der API-Management-Plattformen aufgebaut werden. Dadurch musste die Bewertung der Implementierbarkeit der einzelnen API-Management-Praktiken auf Basis der vom Anbieter bereitgestellten Dokumentation durchgeführt werden. Jedoch stellte sich auch dieses Verfahren als zeitintensiv heraus. Es wird erwartet, dass eine genauere Betrachtung der einzelnen API-Management-Praktiken ohne eine solch strikte zeitliche Restriktion zu belastbareren Ergebnissen bei der Kategorisierung der Implementierbarkeit führt. Dennoch konnte das erarbeitete Bewertungs- und Vergleichsverfahren komplett umgesetzt werden und aufzeigen, wie sich API-Management-Plattformen durch das Anwenden des Bewertungs- und Vergleichsverfahrens funktional und quantifizierbar vergleichen lassen. *(Wörter nach Zählung in Microsoft Word: 9793)*

Literaturverzeichnis

Die O‘reilly Mediathek bietet keine Seitenzahlen in ihren E-Books. Die referenzierten Kapitel wurden über Google Books in Seitenbereiche unterteilt.

Abts, D. (2020). Die SOLID-Prinzipien. In D. Abts, *Grundkurs JAVA* (S. 549–566). Springer Fachmedien Wiesbaden. https://doi.org/10.1007/978-3-658-30494-2_32

Android API reference. (o. J.). *Android API reference | Android Developers*. Abgerufen 12. Juli 2022, von https://developer.android.com/reference

Armour, F., Kaisler, S., & Huizinga, E. (2012). Introduction to Business and Enterprise Architecture: Processes, Approaches and Challenges Minitrack. *2012 45th Hawaii International Conference on System Sciences*, 4229–4229. https://doi.org/10.1109/HICSS.2012.342

Brajesh, D. (2017). *API Management: An Architect’s Guide to Developing and Managing APIs for Your Organization*. https://learning.oreilly.com/library/view/api-management-an/9781484213056/

Bray, T. (2014). *The JavaScript Object Notation (JSON) Data Interchange Format* (Request for Comments RFC 7159). Internet Engineering Task Force. https://doi.org/10.17487/RFC7159

Broadcom. (o. J.). *Continuous API Management*. Abgerufen 26. September 2022, von https://www.broadcom.com/products/software/api-management

Bundesamt für Sicherheit in der Informationstechnik. (o. J.). *DoS- und DDoS-Attacken*. Bundesamt für Sicherheit in der Informationstechnik. Abgerufen 19. August 2022, von https://www.bsi.bund.de/DE/Themen/Verbraucherinnen-und-Verbraucher/Cyber-Sicherheitslage/Methoden-der-Cyber-Kriminalitaet/DoS-Denial-of-Service/dos-denial-of-service.html?nn=132356

Chouhan, M. R. S., & Mishra, S. (2021). *API-led Connectivity Using Mulesoft For Healthcare*. *23*(6), 10.

Coleman Parks Research. (2016). *APIs: Building A Connected Business in the App Economy*. 17.

Cotton, I. W., & Greatorex, F. S. (1968). Data structures and techniques for remote computer graphics. *Proceedings of the December 9-11, 1968, Fall Joint Computer Conference, Part I on - AFIPS ’68 (Fall, Part I)*, 533. https://doi.org/10.1145/1476589.1476661

Frank, R., Strugholz, S., & Meise, F. (2021). *Bausteine der digitalen Transformation*. https://link.springer.com/book/10.1007/978-3-658-35107-6

Gadge, S., & Kotwani, V. (2018). *Microservice Architecture: API Gateway Considerations*. 13.

Google Cloud. (o. J.). *Developer portal solutions | Apigee X*. Google Cloud. Abgerufen 14. September 2022, von https://cloud.google.com/apigee/docs/api-platform/publish/intro-portals

Hannan, J. (1987). Grundelemente der modularen Programmierung. In J. Hannan (Hrsg.), *Ein praktischer Führer für das Management in der Computer-Programmierung* (S. 63–87). Vieweg+Teubner Verlag. https://doi.org/10.1007/978-3-663-14168-6_4

Herbig, N. (2016). *Nutzwertanalyse: Eine Methode zur Bewertung von Lösungsalterativen und zur Entscheidungsfindung*.

Jacobson, D., Brail, G., & Woods, D. (2011). *APIs: A Strategy Guide*. https://learning.oreilly.com/library/view/apis-a-strategy/9781449321628/

Justus-Liebig-Universität Gießen. (o. J.). *Content Management Systeme (CMS)*. Justus-Liebig-Universität Gießen. Abgerufen 19. August 2022, von https://www.uni-giessen.de/fbz/svc/hrz/org/mitarb/abt/3/zms/schulung/webtechniken/cms

Kühnapfel, J. B. (2021). *Scoring und Nutzwertanalysen: Ein Leitfaden für die Praxis*. Springer Fachmedien Wiesbaden. https://doi.org/10.1007/978-3-658-34810-6

Ludwig, H., Keller, A., & Dan, A. (2003). *Web Service Level Agreement (WSLA) Language Specification*. http://cliplab.org/Projects/S-CUBE/papers/ludwig03:wsla-ibm.pdf

Magic Quadrant Forschungsmethodik. (o. J.). Gartner. Abgerufen 23. August 2022, von https://www.gartner.de/de/methoden/magic-quadrants

Maiden, N. A., & Ncube, C. (1998). *Acquiring COTS Software Selection Requirements*. 12.

Masse, M. (2011). *REST API Design Rulebook*. https://learning.oreilly.com/library/view/rest-api-design/9781449317904/

Mathijssen, M., Jansen, S., & Overeem, M. (2022, Februar). *API-m-FAMM: A focus area maturity model for API Management | Elsevier Enhanced Reader*. https://doi.org/10.1016/j.infsof.2022.106890

Mathijssen, M., Overeem, M., & Jansen, S. (2020). *Identification of Practices and Capabilities in API Management: A Systematic Literature Review* (arXiv:2006.10481). arXiv. http://arxiv.org/abs/2006.10481

Mathijssen, M., Overeem, M., & Jansen, S. (2021). *Source Data for the Focus Area Maturity Model for API Management* (arXiv:2007.10611). arXiv. http://arxiv.org/abs/2007.10611

Medjaoui, M., Wilde, E., Mitra, R., & Amundsen, M. (2018). *Continuous API Management*. 66.

Medjaoui, M., Wilde, E., Mitra, R., & Amundsen, M. (2021). *Continuous API Management, 2nd Edition*. https://learning.oreilly.com/library/view/continuous-api-management/9781098103514/

MuleSoft Documentation. (o. J.). *To Configure Headers | MuleSoft Documentation*. Abgerufen 8. September 2022, von https://docs.mulesoft.com/apikit/4.x/configure-headers4-task

Nielsen, H., Mogul, J., Masinter, L. M., Fielding, R. T., Gettys, J., Leach, P. J., & Berners-Lee, T. (1999). *Hypertext Transfer Protocol – HTTP/1.1* (Request for Comments RFC 2616). Internet Engineering Task Force. https://doi.org/10.17487/RFC2616

OWASP Foundation. (o. J.). *OWASP Top Ten | OWASP Foundation*. Abgerufen 11. August 2022, von https://owasp.org/www-project-top-ten/

Pillai, S., & Iijima, K. (2021, September 28). *Magic Quadrant for Full Life Cycle API Management*. https://www.gartner.com/doc/reprints?id=1-27K3L6E5&ct=210929&st=sb

Preibisch, S. (2018). *API Development: A Practical Guide for Business Implementation Success*. Apress. https://doi.org/10.1007/978-1-4842-4140-0

ProgrammableWeb. (2022). ProgrammableWeb. https://www.programmableweb.com/

Weir, L. (2019). *Enterprise API Management: Design and Deliver Valuable Business APIs*. Packt Publishing. https://search.ebscohost.com/login.aspx?direct=true&db=nlebk&AN=2206051&lang=de&site=eds-live

Anhangsverzeichnis

Anhang 1 API-Management-Praktiken in Anlehnung an Mathijssen et al., 2021 S. 9-22 IX
Anhang 2 Handout API-Management-Platform evaluation process XV
Anhang 3 Paarweiser Vergleich Stakeholder 1 XVII
Anhang 4 Paarweiser Vergleich Stakeholder 2 XVII
Anhang 5 Paarweiser Vergleich Stakeholder 3 XVII
Anhang 6 Paarweiser Vergleich Stakeholder 4 XVIII
Anhang 7 Apigee: Bewertung der Implementierbarkeit durch Dokumentation des Anbieters XIX
Anhang 8 Mulesoft: Bewertung der Implementierbarkeit durch Dokumentation des Anbieters XXI
Anhang 9 SAP PI: Bewertung der Implementierbarkeit durch Fachabteilung..... XXIII

Anhang 1 API-Management-Praktiken in Anlehnung an Mathijssen et al., 2021 S. 9-22

Fokusbereich	Fähigkeit des API-Managements	API-Management-Praktik	Beschreibung
API-Lifecycle-Management	Versionsverwaltung	Implementierung einer evolutionären API-Strategie	Die Organisation implementiert eine evolotionäre API-Strategie. Dabei werden Änderungen so umgesetzt, dass sie keine Auswirkung auf bestehende API-Konsumenten haben.
		Implementierung einer Strategie zur API-Versionierung	Die Organisation ist in der Lage mehrere Versionen der gleichen API bereitzustellen. API-Konsumenten können über bspw. die URI ansteuern auf welche Version sie Zugreifen wollen
		Implementierung eines Plans zum Stillegen von APIs	Die Organisation implementiert ein Verfahren über welches veraltete API-Versionen stillgelegt werden können.
		Überprüfung von Rückwärtskompatibilität neuer API-Versionen	Die Organisation kann neue API Versionen automatisch auf Rückwärtskompatibilität zu alten Versionen überprüfen
	Entkopplung API & Anwendung	Entkopplung der API- und Software Version	Die Organisation hat die Version ihrer APIs von ihrer Softwareimplementierung entkoppelt.
		Entkopplung des internen und externen Datenmodells	Die Organisation hat die intern und extern verwendeten Datenmodelle voneinander entkoppelt.
		Entkopplung des internen und externen Datenformats	Die Organisation hat die intern und extern verwendeten Datenformate voneinander entkoppelt.
		Entkopplung des internen und externen Transportprotokolls	Die Organisation hat die intern und extern verwendeten Transportprotokolle voneinander entkoppelt.
	Versenden von Änderungsbenachrichtigungen	Änderungsbenachrichtigungen versenden	Die Organisation versendet (automatisch) E-Mails, um Änderungsbenachrichtigungen über APIs an API-Konsumenten zu verteilen.
		Änderungsbenachrichtigungen über einheitliche Kanäle veröffentlichen	Die Organisation hat die Möglichkeit, Änderungsbenachrichtigungen API-Konsumenten über etablierte Kommunikationskanäle wie bspw. dem Entwicklerportal zu verteilen
		Erweiterung der API um Versionswarnungen	Die Organisation kann zeitnah stillzulegende API Versionen mit Warnungen für API-Konsumenten erweitern
		Veröffentlichung einer Versionierungsroadmap	Die Organisation kann eine Roadmap der geplanten zukünfigen Änderungen an der API veröffentlichen

Security	Authentifizierung	Implementierung von Basis Authentifzierung	Die Organisation kann API-Konsumenten bspw. Durch API-Keys authentifzieren
		implementierung von Authentifizierungsprotokollen	Die Organisation kann API-Konsumenten über Authentifizierungsprotokolle wie OAuth2 authentifizieren
		Implementierung von Single Sign-On	Die Organisation hat Single Sign-on (SSO) implementiert, eine Authentifizierungsmethode, die es API-Konsumenten ermöglicht, sich mit einem einzigen Satz von Anmeldedaten sicher bei mehreren Anwendungen und Websites zu authentifizieren.
	Authorisierung	Implementierung von Zugriffkontrolle	Die Organisation kann die Zugriffsrechte von API-Konsumenten auf APIs steuern
		Implementierung von Token Management	Die Organisation bietet API-Konsumenten die Möglichkeit ihre API-Keys zu verwalten
		Implementierung von Authorisierungsprotokollen	Die Organisation hat ein branchenweit standardisiertes Autorisierungsprotokoll implementiert, z. B. das OAuth 2.0 Autorisierungsprotokoll.
		Implementierung von Autorisierungsbereichen	Die Organisation hat einen Mechanismus für Autorisierungsbereiche implementiert, wie bspw. den OAuth 2.0 Scopes Mechanismus
	Sicherheitslücken erkennen und schließen	Implementierung von Listen mit erlaubten und verbotenen IP-Adressen	Die Organisation erlaubt/blockiert Zugriff von bestimmten IP-Adressen auf die eigenen APIs
		Implementierung von Schutzmaßnahmen vor Injektionsbedrohungen	Die Organisation schützt ihre APIs vor Injektionsbedrohungen wie SQL-Injektion
		Implementierung von Schutzmaßnahmen vor DoS-Attacken	Die Organisation schützt ihre APIs vor DoS-Attacken
		Implementierung von Ablaufprotokollen bei Sicherheitsverletzungen	Die Organisation automatisiert das Verfahren beim Auftreten von Sicherheitsverletzen beispielsweise durch automatische Benachrichtigung der Verantwortlichen
		Durchführen von Sicherheitsüberprüfungen	Die Organisation fordert und überprüft Sicherheitsstandards von ihren API-Konsumenten
		Implementierung von Zero Trust Network Access (ZTNA)	Die Organisation hat eine (ZTNA)-Sicherheitsarchitektur implementiert, bei der nur der Datenverkehr von authentifizierten Benutzern, Geräten und Anwendungen zu den bereitgestellten APIs erlaubt wird
	Verschlüsselung	Implementierung von Verschlüsselung auf Transportebene	Die Organisation verschlüsselt den Datenfluss zu ihren APIs über Verschlüsselungsprotokolle wir TLS
		Implementierung einer Zertifikatsverwaltung	Die Organisation kann die für die Verschlüsselung verwendeten Zertifikate zentral verwalten und monitoren

Performance	Ressourcenverwaltung	Implementierung von Lastverteilung	Die Organisation vewendet Lastverteilung, um den API-Verkehr auf mehrere API-Instanzen zu verteilen
		Implementierung von Skalierung	Die Organisation ist in der Lage, die Menge der verfügbaren Ressourcen je nach Datenverkehr und API-Nutzung reaktiv zu erhöhen oder zu verringern.
		Implementierung von Failover-Strategien	Die Organisation ist in der Lage, auf Ausfälle durch die Implementierung von Failover-Richtlinien zu reagieren. Dies kann durch die automatische Bereitstellung eines Dienstes in einem Standby-Rechenzentrum erfolgen, wenn das primäre System ausfällt oder für Wartungsarbeiten abgeschaltet wird.
		Implementierun von prädiktiver Skalierung	Die Organisation ist in der Lage, die Menge der verfügbaren Ressourcen in Erwartung von steigendem Datenverkerhr proaktiv zu erhöhen oder zu verringern.
	Traffic Management	Implementierung von maximalen Anfragedauern	Die Organisation ist in der Lage die maximale Anfragedauer zu konfigurieren. Anfragen die die maximale Anfragedauer überschreiten werden abgebrochen
		Implementierung von Regeln zur Zwischenspeicherung von Anfrageantworten	Die Organisation implementiert Regeln zur Zwischenspeicherung von Anfrageantworten. Wenn API-Konsumenten Anfragen an denselben URI stellen, kann die zwischengespeicherte Antwort zur Beantwortung verwendet werden, anstatt diese Anfragen erneut zu bearbeiten.
		Implementierung von Limitierungen der Anfragehäufigkeit	Die Organisation implementiert einen Mechanismus, mit dem die Anfragerate von API-Konsumenten gesteuert werden kann
		Implementierung von Anfragedrosslung	Die Organisation verfügt über einen Mechanismus, mit dem API-Anfragen gedrosselt werden können, ohne dass die Verbindung geschlossen wird. Der API-Konsument muss länger auf eine Antwort der API warten
		Implementierung von maximalen Anfragekontingenten	Die Organisation implementiert Regeln, die festlegen wie viele Anfragen ein API-Konsument in einer bestimmten Zeitperiode tätigen darf
		Implementierung von maximalen Antwortgrößen	Die Organisation verfügt über einen Mechanismus, mit dem die Datenmenge, die API-Konsumenten mit einem einzigen Aufruf abrufen dürfen, begrenzt werden kann
		Implementierung von Maßnahemen zum Priorisieren von Datenverkehr	Die Organisation ist in der Lage, der Verarbeitung von API-Aufrufen eine höhere Priorität einzuräumen, basierend auf bestimmten Kundenmerkmalen und/oder -klassen.

Community	Developer Onboarding	API-Konsumenten Registration für die Nutzung von APIs ermöglich	Die Organisation verfügt über einen Mechanismus, mit dem sich API-Konsumenten bei der API registrieren können, um Zugangsdaten zu erhalten.
		Bereitstellen von SDK-Support	Die Organisation bietet API-Konsumenten die Möglichkeit, entweder clientseitige SDKs für die API herunterzuladen oder das SDK selbst aus Standard-API-Definitionsformaten wie OpenAPI zu erstellen.
		Implementierung einer interaktiven API-Konsule	Die Organisation stellt API-Konsumenten eine interaktive Konsole zur Verfügung. Mit dieser Konsole können API-Konsumenten das Verhalten einer API testen.
		Bereitstellung von Sandbox-Umgebungen	Die Organisation stellt API-Konsumenten eine Umgebung zur Verfügung, mit der sie die Merkmale der Produktionsumgebung nachahmen können,
	Support	Einrichtung von Kommunikationskanälen	Die Organisation hat einen Kommunikationskanal zwischen dem API-Anbieter und den API-Konsumenten eingerichtet, über den Unterstützung angeboten werden kann.
		Bearbeitung von Support-Problemen	Die Organisation ist in der Lage, Support-Anfragen entgegenzunehmen und zu bearbeiten
		Bereistellen eines Support-Teams	Die Organisation stellt ein Support-Team bereit, dass bei Fragen oder Problemen mit APIs kontaktiert werden kann
	Documentation	Bereitstellung von Referenzdokumentation	Die Organisation stellt den Konsumenten ihrer APIs eine grundlegende Referenzdokumentation zur Verfügung. Diese Dokumentation sollte jeden Parameter und Anworten der API-Anfragen behinhalten
		Bereistellung von Erklärung zu Schlüsselkonzepten	Bereistellung von Erklärung von Schlüsselkonzepten, indem sie die Referenzdokumentation zusammenfasst und dadurch das Verständnis beschleunigt
		Erstelung von Videotutorials	Die Organisation ist in der Lage, Video-Tutorials zu erstellen, um den Verbrauchern visuelle Informationen über die Verwendung der API und ihre Integration in ihre Anwendungen zu geben.

Community	Community Engagement	Fördern der Social Media Präsenz	Die Organisation ist in der Lage, ihre Präsenz in den sozialen Medien auf Plattformen wie Facebook oder Twitter zu pflegen. Dies kann Aktivitäten wie die Berichterstattung über den Status der API, die Ankündigung von Neuigkeiten und Aktualisierungen, die Beantwortung von Fragen oder die Reaktion auf Feedback umfassen.
		Bereistellen eines Community Forums	Die Organisation bietet (potenziellen) Verbrauchern ihrer API's ein Gemeinschaftsforum. Dieses Forum kann den Aufbau und die Vernetzung einer Entwicklergemeinschaft unterstützen, indem es ihnen eine zentrale Anlaufstelle bietet, über die sie miteinander und mit der Organisation kommunizieren können
		Bereitstellen eines Entwicklerportals	Die Organisation stellt ihren API-Konsumenten ein Entwicklerportal zur Verfügung, über das Ressourcen und Funktionalitäten an einem zentral Ort zugänglich gemacht werden. (Bspw. Dokumentation, User Registration etc.)
		Organisation von Events	Die Organisation ist aktiv an der Organisation von oder der Teilnahme an Veranstaltungen beteiligt, die darauf abzielen, die Entwicklergemeinschaft zu motivieren, ihre API's in ihre Anwendungen einzubinden.
		Nominieren von API-Evangalisten	Die Organisation beschäftigt einen engagierten API-Evangelisten. Diese Person untersützt die Verbreitung der API, indem sie Kundenfeedback einholt und die API's der Organisation bewirbt
	Portfolio Management	Ermöglichung der Entdeckung von APIs	Die Organisation stellt potenziellen Nutzern ihrer APIs einen Mechanismus zur Verfügung, mit dem sie Informationen, wie Dokumentation und Metadaten, über ihre APIs erhalten können.
		Bereistellen eines API-Katalogs	Die Organisation stellt API-Kunden einen API-Katalog zur Verfügung. Dies ist ein durchsuchbarer Katalog von APIs.
		Bündeln von APIs	Die Organisation ist in der Lage, zwei oder mehr APIs zu einem Bündel zusammenzufassen. Dabei handelt es sich um eine Sammlung von APIs, die den Entwicklern als Gruppe präsentiert wird.

Commercial	Service Level Agreements	Veröffentlichung von informellen & einfachen SLAs	Die Organisation hat die Möglichkeit informelle und einfache SLAs für die API-Konsumenten zu veröffentlichen
		Veröffenltichung von formellen & ausführlichen SLAs	Die Organisation ist in der Lage, eine formelle, ausführliche SLAs mit den Nutzern ihrer API(s) zu vereinbaren. Es bestehen Prozesse was bei einem Bruch von SLAs geschieht
		Überwachung der Einhaltung von SLAs	Die Organisation ist in der Lage, proaktiv Metriken zu überwachen, die relevant sind, um zu prüfen, ob die mit API-Kunden getroffenen SLAs eingehalten werden.
		Personalisierte SLAs ermöglichen	Die Organisation hat die Möglichkeit, den Nutzern ihrer APIs personalisierte SLAs anzubieten.
	Monetization Strategy	Unterstützung eines Subscription-based Monetarisierungsmodell	Die Organisation hat ein Monetarisierungsmodell eingeführt, das auf einer Abonnementbasis basiert
		Unterstützung eines Tier-Based Monetarisierungsmodells	Die Organisation hat ein Monetarisierungsmodell eingeführt, das auf einem abgestuften Zugang basiert.
		Unterstützung eines Freemium Monetization Monetarisierungsmodells	Die Organisation hat ein Monetarisierungsmodell gewählt, das auf Freemium-Funktionalitäten und -Zugang basiert.
		Unterstützung eines Metering-Based Monetarisierungsmodells	Die Organisation verwendet ein Monetarisierungsmodell, das auf dem Metering basiert
	Account Management	Implementierung eins Systems zur Subscription Management Systems	Die Organisation verfügt über ein System, API-Konsumenten bestehende Abonnements verwalten können.
		Erstellung von Berichten über API Geschäftswert	Die Organisation ist in der Lage, Berichte über den Geschäftswert ihrer API(s) zu erstellen.
		Bereistellung eines Subscription Reports für Kunden	Die Organisation ist in der Lage, Abonnementberichte für die Nutzer ihrer APIs zu erstellen. Diese Berichte enthalten Metriken, die durch interne Überwachung und Analyse gesammelt wurden. Dazu gehören die Anzahl der getätigten Anrufe, die Leistung und der Status der verbleibenden Kontingente.
		Bereistellung von Vorschlägen für API-Konsumenten	Die Organisation ist in der Lage, Kunden zu schulen und ihnen dabei zu helfen, ihre API(s) so gut und effizient wie möglich zu nutzen.

Anhang 2 Handout API-Management-Platform evaluation process

Handout API-Management-Platform evaluation process

To compare and evaluate different offers of API-Management-Platforms we need to align on which functionality of an API-Management-Platform are value drivers for our API-Strategy.

The functionalities can be grouped into focus areas. A focus area consists of capabilities to API-Management Platforms. In literature six API-Management focus areas are described. We want to collect opinions from our internal experts to weight the importance of those different focus areas in relations to our needs and strategy.

Please read this document to understand the aspects of the different focus areas. Please perform the pairwise comparison of the focus areas afterwards by using the excel file handed in with this document.

Focus Areas:

1. Lifecycle Management:

APIs undergoes several stages over the course of its lifetime; creation, publication, realization, maintenance and retirement. By implementing API-Lifecycle-Management practices, the organization can manage and maintain the versions the API goes through as it evolves over time.

1.1 Version Management:
- Managing multiple versions of APIs
- Deprecation and retirement processes for APIs

1.2 Decoupling API & Application:
- Interface translation between API and back-end service (e.g. data format: Json->XML, protocol conversion: REST->SOAP)

1.3 Update Notification:
- Distribute change logs
- Using consolidated communication channels to distribute change logs (e.g. automatic mails, developer portal)

2. Security

Secure the access to data and assets exposed via API. Protect the publicly exposed APIs from various kinds of attacks

2.1 Authentication:
- Determine the validity of the identity of an API-Client
- Provide different consolidated authentication mechanism (Basic Auth, OAuth2, SSO)

2.2 Authorization:
- Configure level of access to specific APIs and API functionalities

2.3 Threat Detection & Protection:
- Detect malformed request to protect against frequent attack patterns (DoS-Attack, SQL-Injection etc.)

2.4 Encryption:
- Central TLS/SSL offloading for all APIs

3. Performance

Implement consolidated practices to secure the response time and availability of APIs

3.1 Resource Management:
- Load Balancing between API instances
- Automatic scaling of key resources
- Failover policies

3.2 Traffic Management
- Define Caching Policies for APIs
- Perform Rate limiting

4. Observability:

Collect information about API health, performance and resource usage. Aggregate those information's to support decisions processes.

4.1 Monitoring:
- Collect and monitor metrics and variables concerning the exposed API

4.2 Logging:
- Perform logging of consumer behavior and activities

4.3 Analytics:
- Analyze the metrics and variables collected through monitoring and logging

5. Community

Implement practices to engage and support the community that exists around the APIs.

5.1 Developer Onboarding:
- Allow developers to request access to APIs
- Support them with resources like automatic generated SDKs or testing tools like a API console

5.2 Support:
- Provide communication channels to inform about issues

5.3 Documentation:
- Provide central repository for API documentations

5.4 Community Management:
- Allow developers to share their API usage learning using a community forum or developer portal

5.5 Portfolio Management:
- Provide a discoverable catalogue of APIs for potential consumers

6. Commercial

Handle commercial aspects of the API when exposed to third-party consumers or for internal allocation via cost centers.

6.1 Service Level Agreements:
- Manage SLA configurations
- Automatically check whether SLA were broken

6.2 Monetization Strategy:
- Configure different monetization models (Pay-Per-Use, Subscription etc.)
- Execute automatic billing & invoicing

6.3 Account Management:
- Allow consumers to manage their accounts (Subscription level, billing & invoicing etc.)
- Provide consumers with reports on their API usage

Anhang 3 Paarweiser Vergleich Stakeholder 1

	Lifecycle-Management	Security	Performance	Observability	Community	Commcercial	**Summe**	Gewichtungsfaktor
Lifecycle-Management	*XXX*	0	1	0	1	1	**3**	0.1
Security	2	*XXX*	2	1	2	2	**9**	0.3
Performance	1	0	*XXX*	1	2	1	**5**	0.166667
Observability	2	1	1	*XXX*	2	1	**7**	0.233333
Community	1	0	0	0	*XXX*	1	**2**	0.066667
Commcercial	1	0	1	1	1	*XXX*	**4**	0.133333
Gesamt:							**30**	1

Anhang 4 Paarweiser Vergleich Stakeholder 2

	Lifecycle-Management	Security	Performance	Observability	Community	Commcercial	**Summe**	Gewichtungsfaktor
Lifecycle-Management	*XXX*	0	0	0	0	0	**0**	0
Security	2	*XXX*	2	2	2	2	**10**	0.33
Performance	2	0	*XXX*	2	2	2	**8**	0.27
Observability	2	0	0	*XXX*	1	1	**4**	0.13
Community	2	0	0	1	*XXX*	2	5	0.17
Commercial	2	0	0	1	0	*XXX*	**3**	0.1
GESAMT:							**30**	1

Anhang 5 Paarweiser Vergleich Stakeholder 3

	Lifecycle-Management	Security	Performance	Observability	Community	Commcercial	**Summe**	Gewichtungsfaktor
Lifecycle-Management	*XXX*	0	0	0	1	2	**3**	0.10
Security	2	*XXX*	1	1	2	2	**8**	0.27
Performance	2	1	*XXX*	2	2	2	**9**	0.30
Observability	2	1	0	*XXX*	2	2	**7**	0.23
Community	1	0	0	0	*XXX*	2	**3**	0.10
Commercial	0	0	0	0	0	*XXX*	**0**	0.00
GESAMT:							**30**	1.00

Anhang 6 Paarweiser Vergleich Stakeholder 4

	Lifecycle-Management	Security	Performance	Observability	Community	Commcercial	**Summe**	Gewichtungsfaktor
Lifecycle-Management	*XXX*	0	1	0	1	2	**4**	0.13
Security	2	*XXX*	2	2	2	2	**10**	0.33
Performance	1	0	*XXX*	1	1	2	**5**	0.17
Observability	2	0	1	*XXX*	2	2	**7**	0.23
Community	1	0	1	0	*XXX*	2	**4**	0.13
Commercial	0	0	0	0	0	*XXX*	**0**	0
GESAMT:							**30**	1

Anhang 7 Apigee: Bewertung der Implementierbarkeit durch Dokumentation des Anbieters

Fokusbereich	Fähigkeit des API-Managements	Praktik	Implemented?	Quelle
API-Lifecycle-Management	Versionsverwaltung	Implementierung einer evolutionären API-Strategie	implementierbar	https://cloud.google.com/apigee/docs/api-hub/versions
		Implementierung einer Strategie zur API-Versionierung	implementierbar	https://cloud.google.com/apigee/docs/api-hub/versions
		Implementierung eines Plans zum Stillegen von APIs	teilweise implementierbar	https://cloud.google.com/apigee/docs/api-hub/lifecycle-stages
		Überprüfung von Rückwärtskompatibilität neuer API-Versionen	nicht implementierbar	
	Entkopplung API & Anwendung	Entkopplung der API- und Software Version	nicht implementierbar	
		Entkopplung des internen und externen Datenmodells	implementierbar	https://cloud.google.com/apigee/docs/api-platform/integration/data-mapping-overview
		Entkopplung des internen und externen Datenformats	implementierbar	https://cloud.google.com/apigee/docs/api-platform/reference/policies/json-xml-policy
		Entkopplung des internen und externen Transportprotokolls	teilweise implementierbar	https://www.googlecloudcommunity.com/gc/Apigee/JDBC-drivers-in-apigee-X/td-p/389168
	Versenden von Änderungsbenachrichtigungen	Änderungsbenachrichtigungen versenden	teilweise implementierbar	https://cloud.google.com/apigee/docs/api-platform/publish/intro-portals
		Änderungsbenachrichtigungen über einheitliche Kanäle veröffentlichen	teilweise implementierbar	https://cloud.google.com/apigee/docs/api-platform/publish/intro-portals
		Erweiterung der API um Versionswarnungen	teilweise implementierbar	https://cloud.google.com/apigee/docs/api-platform/reference/policies/assign-message-policy
		Veröffentlichung einer Versionierungsroadmap	teilweise implementierbar	https://cloud.google.com/apigee/docs/api-platform/publish/intro-portals
Security	Authentifizierung & Authorisierung	Implementierung von Basis Authentifizierung	implementierbar	https://cloud.google.com/apigee/docs/api-platform/reference/policies/basic-authentication-policy?hl=de
		Implementierung von Authentifizierungsprotokollen	implementierbar	https://cloud.google.com/apigee/docs/api-platform/reference/policies/oauthv2-policy?hl=de
		Implementierung von Single Sign-On	implementierbar	https://cloud.google.com/apigee/docs/api-platform/reference/policies/saml-assertion-policy?hl=de
		Implementierung von Zugriffkontrolle	implementierbar	https://cloud.google.com/apigee/docs/api-platform/reference/policies/oauthv2-policy?hl=de
		Implementierung von Token Management	implementierbar	https://docs.apigee.com/api-platform/publish/creating-apps-surface-your-api#manage-credentials
		Implementierung von Authorisierungsprotokollen	implementierbar	https://docs.apigee.com/api-platform/reference/policies/verify-jwt-policy
		Implementierung von Autorisierungsbereichen	implementierbar	https://docs.apigee.com/api-platform/security/oauth/working-scopes#howarescopesenforced
	Erkennen und Schützen vor Sicherheitslücken	Implementierung von Listen mit erlaubten und verbotenen IP-Adressen	implementierbar	https://cloud.google.com/apigee/docs/api-platform/reference/policies/access-control-policy?hl=de
		Implementierung von Richtlinien zum Schutz vor Injektionsbedrohungen	implementierbar	https://cloud.google.com/apigee/docs/api-platform/reference/policies/json-threat-protection-policy?hl=de
		Implementierung von Schutzmaßnahmen vor DoS Attacken	implementierbar	https://cloud.google.com/apigee/docs/api-platform/system-administration/preventing-dos?hl=de
		Implementierung von Maßnahmen bei Sicherheitsverletzungen	nicht implementierbar	
		Durchführen von Sicherheitsüberprüfungen	nicht implementierbar	
		Implementierung von Zero Trust Network Access (ZTNA)	nicht implementierbar	
	Verschlüsselung	Implementierung von Verschlüsselung auf Transportebene	implementierbar	https://docs.apigee.com/api-platform/system-administration/configuring-ssl-cloud-based-edge-installation https://docs.apigee.com/api-platf
		Implementierung einer Zertifikatsverwaltung	implementierbar	https://cloud.google.com/apigee/docs/api-platform/system-administration/options-configuring-tls?hl=de
Performance	Ressourcenverwaltung	Implementierung von Lastverteilung	implementierbar	https://cloud.google.com/apigee/docs/api-platform/deploy/load-balancing-across-backend-servers?hl=de
		Implementierung von Skalierung	teilweise implementierbar	https://cloud.google.com/apigee/docs/hybrid/v1.8/scale-and-autoscale
		Implementierung von Failover-Strategien	nicht implementierbar	
		Implementierun von prädiktiver Skalierung	nicht implementierbar	
	Traffic Management	Implementierung von maximalen Anfragedauern	implementierbar	https://cloud.google.com/apigee/docs/api-platform/reference/endpoint-properties-reference?hl=de
		Implementierung von Regeln zur Zwischenspeicherung von Anfrageantworten	implementierbar	https://docs.apigee.com/api-platform/cache/caching-edge#:~:text=Apigee%20Edge%20provides%20caching%20for,is%20useful%20for%20most
		Implementierung von Limitierungen der Anfragehäufigkeit	implementierbar	https://cloud.google.com/apigee/docs/api-platform/reference/policies/quota-policy
		Implementierung von Anfragedrosslung	implementierbar	https://cloud.google.com/apigee/docs/api-platform/reference/policies/spike-arrest-policy
		Implementierung von maximalen Anfragekontingenten	implementierbar	https://docs.apigee.com/api-platform/reference/policies/quota-policy#dynamic-quota
		Implementierung von maximalen Antwortgrößen	nicht implementierbar	
		Implementierung von Maßnahemen zum priorisieren von Datenverkehr	nicht implementierbar	

Observability	Monitoring	Überwachen des API-Heath-States	implementierbar	https://cloud.google.com/apigee/docs/api-platform/debug/health-check
		Überwachen der API-Performance	implementierbar	https://docs.apigee.com/api-platform/analytics/api-proxy-performance-dashboard
		Überwachen des Ressourcenverbrauchs	implementierbar	https://docs.apigee.com/api-platform/analytics/api-proxy-performance-dashboard
	Logging	Fehler protokollieren	implementierbar	https://docs.apigee.com/api-monitoring/investigate
		Zugriffsversuche protokollieren	implementierbar	https://docs.apigee.com/api-monitoring/logs-api
		API-Anfragen protokollieren	nicht implementierbar	
		Auswertung von Aktivitätsmuster der API-Konsumenten	nicht implementierbar	
	Analytics	Melden von Fehlern	nicht implementierbar	
		Veröffentlichung des API-Health-States an API-Konsumenten	nicht implementierbar	
		Erstellung von benutzerdefinierten Analyseberichten	implementierbar	https://docs.apigee.com/api-platform/analytics/create-custom-reports
		Konfigurieren von Warnungsmeldungen	implementierbar	https://docs.apigee.com/api-monitoring/alerts-notifications
		Implementierung von prädiktiver Analyse	nicht implementierbar	
Community	Developer Onboarding	API-Konsumenten Registration für die Nutzung von APIs ermöglich	implementierbar	https://docs.apigee.com/api-platform/publish/portal/configure-register-sign-in
		Bereitstellen von SDK-Support	teilweise implementierbar	https://cloud.google.com/apigee/docs/api-hub/dependencies
		Implementierung einer interaktiven API-Konsule	implementierbar	https://docs.apigee.com/api-platform/publish/portal/publish-apis#smartdocs
		Bereitstellung von Sandbox-Umgebungen	implementierbar	https://cloud.google.com/apigee/docs/api-hub/deployments
	Support	Einrichtung von Kommunikationskanälen	teilweise implementierbar	https://cloud.google.com/apigee/docs/api-platform/publish/intro-portals
		Bearbeitung von Support-Problemen	teilweise implementierbar	https://cloud.google.com/apigee/docs/api-platform/publish/intro-portals
		Bereistellen eines Support-Teams	nicht implementierbar	https://cloud.google.com/apigee/docs/api-platform/publish/intro-portals
	Documentation	Bereitstellung von Referenzdokumentation	implementierbar	https://cloud.google.com/apigee/docs/api-platform/tutorials/create-api-proxy-openapi-spec?hl=de
		Bereitstellung von Erklärung zu Schlüsselkonzepten	implementierbar	https://docs.apigee.com/api-platform/publish/portal/publish-apis#smartdocs
		Erstelung von Videotutorials	teilweise implementierbar	https://cloud.google.com/apigee/docs/api-platform/publish/intro-portals
	Community Engagement	Fördern der Social Media Präsenz	nicht implementierbar	
		Bereistellen eines Community Forums	teilweise implementierbar	https://cloud.google.com/apigee/docs/api-platform/publish/intro-portals
		Bereitstellen eines Entwicklerportals	teilweise implementierbar	https://cloud.google.com/apigee/docs/api-platform/publish/intro-portals
		Organisation von Events	nicht implementierbar	
		Nominieren von API-Evangalisten	nicht implementierbar	
	Portfolio Management	Ermöglichung der Entdeckung von APIs	implementierbar	https://cloud.google.com/apigee/docs/api-hub/apis-list
		Bereistellen eines API-Katalogs	implementierbar	https://cloud.google.com/apigee/docs/api-hub/apis-list
		Bündeln von APIs	implementierbar	https://cloud.google.com/apigee/docs/api-platform/publish/what-api-product
Commercial	Service Level Agreements	Veröffentlichung von informelle & einfachen SLAs	nicht implementierbar	
		Veröffenltichung von formellen & ausführlichen SLAs	nicht implementierbar	
		Überwachung der Einhaltung von SLAs	nicht implementierbar	
		Personalisierte SLAs ermöglichen	nicht implementierbar	
	Monetization Strategy	Unterstützung eines Subscription-based Monetarisierungsmodell	implementierbar	https://cloud.google.com/apigee/docs/api-platform/monetization/overview?hl=de
		Unterstützung eines Tier-Based Monetarisierungsmodells	implementierbar	https://cloud.google.com/apigee/docs/api-platform/monetization/overview?hl=de
		Unterstützung eines Freemium Monetization Monetarisierungsmodells	implementierbar	https://cloud.google.com/apigee/docs/api-platform/monetization/overview?hl=de
		Unterstützung eines Metering-Based Monetarisierungsmodells	implementierbar	https://cloud.google.com/apigee/docs/reference/apis/apigee/rest/v1/organizations.apiproducts.rateplans#consumptionpricingtype
	Account Management	Implementierung eins Systems zur Subscription Management Systems	implementierbar	https://cloud.google.com/apigee/docs/api-platform/monetization/purchase-api-products-api
		Erstellung von Berichten über API Geschäftswert	implementierbar	https://cloud.google.com/apigee/docs/api-platform/monetization/generate-reports?hl=de
		Bereistellung eines Subscription Reports für Kunden	nicht implementierbar	
		Bereistellung von Vorschlägen für API-Konsumenten	nicht implementierbar	

Anhang 8 Mulesoft: Bewertung der Implementierbarkeit durch Dokumentation des Anbieters

Fokusbereich	Fähigkeit des API-Managements	Praktik	implementierbar?	Quelle
API-Lifecycle-Management	Versionsverwaltung	Implementierung einer evolutionären API-Strategie	implementierbar	https://docs.mulesoft.com/exchange/manage-versions
		Implementierung einer Strategie zur API-Versionierung	implementierbar	https://docs.mulesoft.com/exchange/to-change-raml-version#api-version-in-application-url
		Implementierung eines Plans zum Stillegen von APIs	teilweise implementierbar	https://docs.mulesoft.com/exchange/manage-versions#change-lifecycle-state-for-a-version-of-an-asset https://docs.mulesoft.com/api-manager/2.x/deprecate-api-latest-task
		Überprüfung von Rückwärtskompatibilität neuer API-Versionen	nicht implementierbar	
	Entkopplung API & Anwendung	Entkopplung der API- und Software Version	nicht implementierbar	
		Entkopplung des internen und externen Datenmodells	implementierbar	https://docs.mulesoft.com/studio/7.5/transform-message-component-concept-studio
		Entkopplung des internen und externen Datenformats	implementierbar	https://docs.mulesoft.com/dataweave/2.3/dataweave-cookbook-perform-basic-transformation
		Entkopplung des internen und externen Transportprotokolls	implementierbar	https://docs.mulesoft.com/mule-runtime/3.9/transports-reference
	Versenden von Änderungsbenachrichtigungen	Änderungsbenachrichtigungen versenden	nicht implementierbar	
		Änderungsbenachrichtigungen über einheitliche Kanäle veröffentlichen	nicht implementierbar	
		Erweiterung der API um Versionswarnungen	teilweise implementierbar	https://docs.mulesoft.com/apikit/4.x/configure-headers4-task
		Veröffentlichung einer Versionierungsroadmap	nicht implementierbar	
Security	Authentifizierung & Authorisierung	Implementierung von Basis Authentifizierung	implementierbar	https://docs.mulesoft.com/policies/policies-included-basic-auth-simple https://docs.mulesoft.com/policies/policies-included-basic-auth-ldap
		Implementierung von Authentifizierungsprotokollen	implementierbar	https://docs.mulesoft.com/policies/policies-included-pingfederate-oauth-token-enforcement
		Implementierung von Single Sign-On	nicht implementierbar	
		Implementierung von Zugriffkontrolle	implementierbar	https://docs.mulesoft.com/policies/policies-included-pingfederate-oauth-token-enforcement
		Implementierung von Token Management	nicht implementierbar	
		Implementierung von Authorisierungsprotokollen	implementierbar	https://docs.mulesoft.com/policies/policies-included-pingfederate-oauth-token-enforcement
		Implementierung von Autorisierungsbereichen	implementierbar	https://docs.mulesoft.com/policies/policies-included-pingfederate-oauth-token-enforcement
	Erkennen und Schützen vor Sicherheitslücken	Implementierung von Listen mit erlaubten und verbotenen IP-Adressen	implementierbar	https://docs.mulesoft.com/policies/policies-included-ip-allowlist | https://docs.mulesoft.com/policies/policies-included-ip-blocklist
		Implementierung von Richtlinien zum Schutz vor Injektionsbedrohungen	implementierbar	https://docs.mulesoft.com/policies/policies-included-xml-threat-protection https://docs.mulesoft.com/policies/policies-included-json-threat-protection
		Implementierung von Schutzmaßnahmen vor DoS Attacken	implementierbar	https://docs.mulesoft.com/anypoint-security/dos-policy
		Implementierung von Maßnahmen bei Sicherheitsverletzungen	nicht implementierbar	
		Durchführen von Sicherheitsüberprüfungen	nicht implementierbar	
		Implementierung von Zero Trust Network Access (ZTNA)	nicht implementierbar	
	Verschlüsselung	Implementierung von Verschlüsselung auf Transportebene	implementierbar	https://docs.mulesoft.com/gateway/1.1/flex-local-tls-config
		Implementierung einer Zertifikatsverwaltung	nicht implementierbar	
Performance	Ressourcenverwaltung	Implementierung von Lastverteilung	implementierbar	https://docs.mulesoft.com/runtime-manager/dedicated-load-balancer-tutorial
		Implementierung von Skalierung	teilweise implementierbar	https://docs.mulesoft.com/gateway/1.1/flex-gateway-replicas
		Implementierung von Failover-Strategien	nicht implementierbar	
		Implementierun von prädiktiver Skalierung	nicht implementierbar	
	Traffic Management	Implementierung von maximalen Anfragedauern	implementierbar	https://docs.mulesoft.com/mule-runtime/4.2/global-settings-configuration
		Implementierung von Regeln zur Zwischenspeicherung von Anfrageantworten	implementierbar	https://docs.mulesoft.com/policies/policies-included-http-caching
		Implementierung von Limitierungen der Anfragehäufigkeit	implementierbar	https://docs.mulesoft.com/api-manager/2.x/rate-limit-1.2.0-task
		Implementierung von Anfragedrosslung	implementierbar	https://docs.mulesoft.com/policies/policies-mule3-rate-limiting-and-throttling#throttling-retries
		Implementierung von maximalen Anfragekontingenten	implementierbar	https://docs.mulesoft.com/api-manager/2.x/defining-sla-tiers#defining-a-tier
		Implementierung von maximalen Antwortgrößen	nicht implementierbar	
		Implementierung von Maßnahmen zum priorisieren von Datenverkehr	nicht implementierbar	

Observability	Monitoring	Überwachen des API-Heath-States	implementierbar	https://docs.mulesoft.com/api-functional-monitoring/
		Überwachen der API-Performance	implementierbar	https://docs.mulesoft.com/gateway/1.1/flex-view-api-metrics
		Überwachen des Ressourcenverbrauchs	implementierbar	https://docs.mulesoft.com/gateway/1.1/flex-view-api-metrics
	Logging	Fehler protokollieren	implementierbar	https://docs.mulesoft.com/gateway/1.1/flex-conn-message-log
		Zugriffsversuche protokollieren	implementierbar	https://docs.mulesoft.com/mule-runtime/4.3/logger-component-reference
		API-Anfragen protokollieren	implementierbar	https://docs.mulesoft.com/mule-runtime/4.3/logger-component-reference
		Auswertung von Aktivitätsmuster der API-Konsumenten	implementierbar	https://docs.mulesoft.com/mule-runtime/4.3/logger-component-reference
	Analytics	Melden von Fehlern	implementierbar	https://docs.mulesoft.com/monitoring/reports#failures
		Veröffentlichung des API-Health-States an API-Konsumenten	teilweise implementierbar	https://docs.mulesoft.com/monitoring/alerts#access-the-alerts-page
		Erstellung von benutzerdefinierten Analyseberichten	implementierbar	https://docs.mulesoft.com/api-manager/2.x/viewing-api-analytics#creating-custom-charts
		Konfigurieren von Warnungsmeldungen	implementierbar	https://docs.mulesoft.com/gateway/1.1/flex-use-api-alerts
		Implementierung von prädiktiver Analyse	nicht implementierbar	
Community	Developer Onboarding	API-Konsumenten Registration für die Nutzung von APIs ermöglich	implementierbar	https://docs.mulesoft.com/api-community-manager/manage-members#member-registration-and-activation
		Bereitstellen von SDK-Support	nicht implementierbar	
		Implementierung einer interaktiven API-Konsule	implementierbar	https://docs.mulesoft.com/api-community-manager/api-console
		Bereitstellung von Sandbox-Umgebungen	implementierbar	https://docs.mulesoft.com/access-management/environments#types-of-environments https://docs.mulesoft.com/api-community-manager/sandboxes
	Support	Einrichtung von Kommunikationskanälen	teilweise implementierbar	https://docs.mulesoft.com/api-community-manager/
		Bearbeitung von Support-Problemen	nicht implementierbar	
		Bereitstellen eines Support-Teams	nicht implementierbar	
	Documentation	Bereitstellung von Referenzdokumentation	implementierbar	https://docs.mulesoft.com/api-manager/2.x/proxy-deploy-raml-oas-proxy#select_an_api (RAML und OAS)
		Bereistellung von Erklärung zu Schlüsselkonzepten	teilweise implementierbar	https://docs.mulesoft.com/api-community-manager/publish-content
		Erstelung von Videotutorials	teilweise implementierbar	https://docs.mulesoft.com/api-community-manager/publish-content
	Community Engagement	Fördern der Social Media Präsenz	nicht implementierbar	
		Bereistellen eines Community Forums	implementierbar	https://docs.mulesoft.com/api-community-manager/moderate-forums
		Bereitstellen eines Entwicklerportals	implementierbar	https://docs.mulesoft.com/api-community-manager/
		Organisation von Events	nicht implementierbar	
		Nominieren von API-Evangalisten	nicht implementierbar	
	Portfolio Management	Ermöglichung der Entdeckung von APIs	implementierbar	https://docs.mulesoft.com/api-community-manager/site-structure
		Bereistellen eines API-Katalogs	implementierbar	https://docs.mulesoft.com/api-community-manager/site-structure
		Bündeln von APIs	implementierbar	https://docs.mulesoft.com/api-manager/2.x/api-groups-landing-page
Commercial	Service Level Agreements	Veröffentlichung von informelle & einfachen SLAs	nicht implementierbar	
		Veröffenltichung von formellen & ausführlichen SLAs	nicht implementierbar	
		Überwachung der Einhaltung von SLAs	nicht implementierbar	
		Personalisierte SLAs ermöglichen	nicht implementierbar	
	Monetization Strategy	Unterstützung eines Subscription-based Monetarisierungsmodell	nicht implementierbar	
		Unterstützung eines Tier-Based Monetarisierungsmodells	nicht implementierbar	
		Unterstützung eines Freemium Monetization Monetarisierungsmodells	nicht implementierbar	
		Unterstützung eines Metering-Based Monetarisierungsmodells	nicht implementierbar	
	Account Management	Implementierung eins Systems zur Subscription Management Systems	nicht implementierbar	
		Erstellung von Berichten über API Geschäftswert	nicht implementierbar	
		Bereistellung eines Subscription Reports für Kunden	nicht implementierbar	
		Bereistellung von Vorschlägen für API-Konsumenten	nicht implementierbar	

Anhang 9 SAP PI: Bewertung der Implementierbarkeit durch Fachabteilung

Fokusbereich	Fähigkeit des API-Managements	Praktik	implementierbar?
API-Lifecycle-Management	Versionsverwaltung	Implementierung einer evolutionären API-Strategie	nicht implementierbar
		Implementierung einer Strategie zur API-Versionierung	nicht implementierbar
		Implementierung eines Plans zum Stillegen von APIs	nicht implementierbar
		Überprüfung von Rückwärtskompatibilität neuer API-Versionen	nicht implementierbar
	Entkopplung API & Anwendung	Entkopplung der API- und Software Version	nicht implementierbar
		Entkopplung des internen und externen Datenmodells	implementierbar
		Entkopplung des internen und externen Datenformats	implementierbar
		Entkopplung des internen und externen Transportprotokolls	implementierbar
	Versenden von Änderungsbenachrichtigungen	Änderungsbenachrichtigungen versenden	nicht implementierbar
		Änderungsbenachrichtigungen über einheitliche Kanäle veröffentlichen	nicht implementierbar
		Erweiterung der API um Versionswarnungen	nicht implementierbar
		Veröffentlichung einer Versionierungsroadmap	nicht implementierbar
Security	Authentifizierung & Authorisierung	Implementierung von Basis Authentifzierung	implementierbar
		implementierung von Authentifizierungsprotokollen	implementierbar
		Implementierung von Single Sign-On	implementierbar
		Implementierung von Zugriffkontrolle	implementierbar
		Implementierung von Token Management	implementierbar
		Implementierung von Authorisierungsprotokollen	implementierbar
		Implementierung von Autorisierungsbereichen	implementierbar
	Erkennen und Schützen vor Sicherheitslücken	Implementierung von Listen mit erlaubten und verbotenen IP-Adressen	implementierbar
		Implementierung von Richtlinien zum Schutz vor Injektionsbedrohungen	nicht implementierbar
		Implementierung von Schutzmaßnahmen vor DoS Attacken	nicht implementierbar
		Implementierung von Maßnahmen bei Sicherheitsverletzungen	nicht implementierbar
		Durchführen von Sicherheitsüberprüfungen	nicht implementierbar
		Implementierung von Zero Trust Network Access (ZTNA)	nicht implementierbar
	Verschlüsselung	Implementierung von Verschlüsselung auf Transportebene	implementierbar
		Implementierung einer Zertifikatsverwaltung	implementierbar
Performance	Ressourcenverwaltung	Implementierung von Lastverteilung	implementierbar
		Implementierung von Skalierung	nicht implementierbar
		Implementierung von Failover-Strategien	nicht implementierbar
		Implementierun von prädiktiver Skalierung	nicht implementierbar
	Traffic Management	Implementierung von maximalen Anfragedauern	implementierbar
		Implementierung von Regeln zur Zwischenspeicherung von Anfrageantworten	implementierbar
		Implementierung von Limitierungen der Anfragehäufigkeit	implementierbar
		Implementierung von Anfragedrosslung	implementierbar
		Implementierung von maximalen Anfragekontingenten	implementierbar
		Implementierung von maximalen Antwortgrößen	nicht implementierbar
		Implementierung von Maßnahemen zum priorisieren von Datenverkehr	implementierbar

Observability	Monitoring	Überwachen des API-Heath-States	implementierbar
		Überwachen der API-Performance	implementierbar
		Überwachen des Ressourcenverbrauchs	implementierbar
	Logging	Fehler protokollieren	implementierbar
		Zugriffsversuche protokollieren	implementierbar
		API-Anfragen protokollieren	implementierbar
		Auswertung von Aktivitätsmuster der API-Konsumenten	implementierbar
	Analytics	Melden von Fehlern	implementierbar
		Veröffentlichung des API-Health-States an API-Konsumenten	implementierbar
		Erstellung von benutzerdefinierten Analyseberichten	implementierbar
		Konfigurieren von Warnungsmeldungen	implementierbar
		Implementierung von prädiktiver Analyse	nicht implementierbar
Community	Developer Onboarding	API-Konsumenten Registration für die Nutzung von APIs ermöglich	nicht implementierbar
		Bereitstellen von SDK-Support	nicht implementierbar
		Implementierung einer interaktiven API-Konsule	nicht implementierbar
		Bereitstellung von Sandbox-Umgebungen	nicht implementierbar
	Support	Einrichtung von Kommunikationskanälen	nicht implementierbar
		Bearbeitung von Support-Problemen	nicht implementierbar
		Bereistellen eines Support-Teams	nicht implementierbar
	Documentation	Bereitstellung von Referenzdokumentation	teilweise implementierbar
		Bereistellung von Erklärung zu Schlüsselkonzepten	nicht implementierbar
		Erstelung von Videotutorials	nicht implementierbar
	Community Engagement	Fördern der Social Media Präsenz	nicht implementierbar
		Bereistellen eines Community Forums	nicht implementierbar
		Bereitstellen eines Entwicklerportals	nicht implementierbar
		Organisation von Events	nicht implementierbar
		Nominieren von API-Evangalisten	nicht implementierbar
	Portfolio Management	Ermöglichung der Entdeckung von APIs	nicht implementierbar
		Bereistellen eines API-Katalogs	nicht implementierbar
		Bündeln von APIs	nicht implementierbar
Commercial	Service Level Agreements	Veröffentlichung von informelle & einfachen SLAs	nicht implementierbar
		Veröffenltichung von formellen & ausführlichen SLAs	nicht implementierbar
		Überwachung der Einhaltung von SLAs	nicht implementierbar
		Personalisierte SLAs ermöglichen	nicht implementierbar
	Monetization Strategy	Unterstützung eines Subscription-based Monetarisierungsmodell	nicht implementierbar
		Unterstützung eines Tier-Based Monetarisierungsmodells	nicht implementierbar
		Unterstützung eines Freemium Monetization Monetarisierungsmodells	nicht implementierbar
		Unterstützung eines Metering-Based Monetarisierungsmodells	nicht implementierbar
	Account Management	Implementierung eins Systems zur Subscription Management Systems	nicht implementierbar
		Erstellung von Berichten über API Geschäftswert	nicht implementierbar
		Bereistellung eines Subscription Reports für Kunden	nicht implementierbar
		Bereistellung von Vorschlägen für API-Konsumenten	nicht implementierbar