Untersuchung und Vergleich verschiedener Ansätze zur Implementierung von Microfrontends

Masterarbeit zur Erlangung des Master-Grades
Master of Science im Studiengang Medieninformatik
an der Fakultät für Informatik und Ingenieurwissenschaften
der Technischen Hochschule Köln

vorgelegt von:	Javad Alamdar
Matrikel-Nr.:	11136142
E-Mail:	javad.alamdar@smail.th-koeln.de

eingereicht bei:	Prof. Christian Noss
Zweitgutachter/in:	Prof. Dr. Hoai Viet Nguyen

Düsseldorf, 11.03.2024

Erklärung

Ich versichere, die von mir vorgelegte Arbeit selbstständig verfasst zu haben. Alle Stellen, die wörtlich oder sinngemäß aus veröffentlichten oder nicht veröffentlichten Arbeiten anderer oder der Verfasserin/des Verfassers selbst entnommen sind, habe ich als entnommen kenntlich gemacht. Sämtliche Quellen und Hilfsmittel, die ich für die Arbeit benutzt habe, sind angegeben. Die Arbeit hat mit gleichem Inhalt bzw. in wesentlichen Teilen noch keiner anderen Prüfungsbehörde vorgelegen.

Düsseldorf, 11.03.2024

Ort, Datum

Rechtsverbindliche Unterschrift

Danksagung

Ich möchte mich herzlich bei all jenen bedanken, die mich während der Erstellung dieser Masterarbeit unterstützt und motiviert haben.

Ein besonderer Dank gilt meinen Betreuern, Herrn Prof. Noss und Herrn Prof. Dr. Nguyen, für ihre hervorragende Unterstützung und ihr wertvolles Feedback. Ebenso möchte ich Frau Dr. Mirjam Berg für ihre hilfreiche Unterstützung bei der Anleitung meiner Masterarbeit danken.

Zudem widme ich einen herzlichen Dank an meine Frau. Ihre Liebe und Unterstützung waren die Antriebskraft hinter meinem akademischen Erfolg.

Hinweis

Die vorliegende Masterarbeit wurde in der deutschen Sprache verfasst, um an der Technischen Hochschule Köln vorgestellt zu werden. Da ich die deutsche Sprache als Zweitsprache erlernt habe, wurde nach Rücksprache mit den Betreuern Herrn Prof. Christian Noss und Prof. Dr. Hoai Viet Nguyen die Genehmigung zur Verwendung von von KI-Tools wie ChatGPT und Bard zur sprachlichen und stilistischen Verbesserung des Texts sowie zur Rechtschreibprüfung erteilt.

Zur besseren Lesbarkeit wird in dieser Masterarbeit das generische Maskulinum verwendet. Die in dieser Arbeit verwendeten Personenbezeichnungen beziehen sich – sofern nicht anders kenntlich gemacht – auf alle Geschlechter.

Abstract

Microfrontends are an architectural model for frontend development based on similar principles to microservices. It allows monolithic frontends to be split into independent, smaller applications that are used by large companies. For this reason, a basic understanding of the implementation is crucial.

This master's thesis examines and compares various approaches for implementing micro frontends in web-based applications to determine when their use is appropriate. The following questions are addressed: What approaches exist, and how do they differ? What are the advantages and disadvantages of micro frontends? Due to the relatively limited research on this topic, a descriptive study was conducted.

The results indicate Microfrontends expedite development and enhance scalability. Various implementation approaches exist, including client-side, server-side, and edge-side, each with its own set of advantages and disadvantages. Choosing the appropriate approach based on project requirements is crucial. Thorough research is necessary to explore this emerging topic.

The frontend community is actively seeking solutions to provide comprehensive support for microfrontends by integrating new tools. Careful examination of all aspects and consideration of future development is essential before starting the project.

Kurzfassung

Microfrontends sind ein Architekturmodell für die Frontend-Entwicklung, das auf ähnlichen Prinzipien wie Microservices basiert. Es ermöglicht die Aufteilung monolithischer Frontends in unabhängige, kleinere Anwendungen, die von großen Unternehmen eingesetzt werden. Aus diesem Grund ist ein grundlegendes Verständnis der Implementierung entscheidend.

Diese Masterarbeit untersucht und vergleicht verschiedene Ansätze zur Implementierung von Microfrontends in webbasierten Anwendungen, um herauszufinden, wann ihr Einsatz sinnvoll ist. Dabei werden die folgenden Fragen gestellt: Welche Ansätze gibt es und wie unterscheiden sie sich? Was sind die Vor- und Nachteile von Microfrontends? Aufgrund des vergleichsweise geringen Forschungsstandes zu diesem Thema wurde eine deskriptive Untersuchung durchgeführt.

Die Ergebnisse zeigen, dass Microfrontends die Entwicklung beschleunigen und die Skalierbarkeit verbessern. Es wurde festgestellt, dass verschiedene Implementierungsansätze wie Client-, Server- und Edge-Side existieren, die jeweils ihre eigenen Vor- und Nachteile haben. Die Wahl des geeigneten Ansatzes hängt von den Projektanforderungen ab. Weitere Forschung ist erforderlich, um dieses aufkommende Thema im Detail zu untersuchen.

Die Frontend-Gemeinschaft sucht aktiv nach Lösungen, um durch die Integration neuer Werkzeuge eine umfassende Unterstützung für Microfrontends zu schaffen. Es wird betont, dass vor Projektbeginn alle Aspekte sorgfältig untersucht und die zukünftige Entwicklung des Projekts berücksichtigt werden sollten.

Inhaltsverzeichnis

Tabellenverzeichnis

Abbildungsverzeichnis

Abkürzungsverzeichnis

AJAX Asynchronous JavaScript and XML

CDN Content Delivery Network

CSR Client Side Includes

CSR Client Side Rendering

DDD Domain Driven Design

DOM Document Object Model

ESI Edge Side Includes

MPA Multi Page Application

REST Representational State Transfer

SPA Single Page Application

SEO Search Engine Optimization

SSR Server Side Rendering

HTTP Hypertext Transfer Protocol

1 Einleitung

Webanwendungen sind zu einem wesentlichen Bestandteil des alltäglichen Lebens geworden und spielen eine entscheidende Rolle in Bereichen wie unter anderem E-Commerce, Unternehmensanwendungen sowie sozialen Medien. Die Webanwendungen müssen nicht nur von hoher Qualität und Benutzerfreundlichkeit sein, sondern auch skalierbar, wartbar, flexibel und schnell laden. Aus diesem Grund lässt sich momentan eine Veränderung der Webarchitekturen feststellen [1] (vgl. Geers 2020: 1). Diese Veränderungen lassen sich auf die Implementierung neuer Trends und Technologien wie Microservices im Backend und Microfrontends im Frontend zurückführen, die eine Revolution in der Entwicklung und Verwaltung von Webanwendungen hervorrufen und die Trennung von Frontend und Backend ermöglichen [2] (vgl. Wanjala 2022: 337).

Microfrontends basieren auf ähnlichen Prinzipien wie Microservices und haben sich nach Pavlenko et al. (2020: 49) und Petcu et al. (2023: 323) zu der führenden Technologie für die Implementierung der Webanwendungen entwickelt [3, 10] (vgl. Pavlenko et al. 2020: 49; Petcu et al. 2023: 323). Der Grundgedanke hinter Microfrontends besteht darin, die Entwicklung, Wartung und Skalierbarkeit von Frontend-Anwendungen zu verbessern, indem sie in kleinere, handhabbare Teile aufgeteilt werden [1, 3, 10, 11] (vgl. Geers 2020: 1; Pavlenko et al. 2020: 54; Petcu et al. 2023: 323; Peltonen et al. 2021: 4). Jedes Microfrontend kann von seinem eigenen Team entwickelt, getestet und bereitgestellt werden. Dadurch können Teams unabhängig voneinander arbeiten, ohne sich gegenseitig zu beeinträchtigen [1, 3, 4] (vgl. Geers 2020: 1; Pavlenko et al. 2020: 54; Jackson 2019: o. S.).

1.1 Problemfeld und Kontext

Webentwickler stehen vor der anspruchsvollen Aufgabe, komplexe Webanwendungen zu erstellen, insbesondere wenn es darum geht, die Skalierbarkeit, die Wartbarkeit und die Flexibilität von Frontend-Anwendungen zu gewährleisten. Diese Anforderungen haben zu einer Evolution in der Art und Weise geführt, wie Frontend-Anwendungen entwickelt und bereitgestellt werden. Das Konzept der Microfrontends hat erhebliche Aufmerksamkeit erregt, da es eine vielversprechende Lösung zu sein scheint, um die Komplexität großer Anwendungen zu bewältigen, indem sie sie in kleinere, eigenständige Einheiten zerlegen [2, 10] (vgl. Wanjala 2022: 337; Petcu et al. 2023: 323).

Das hiermit verbundene Problemfeld, dem diese Untersuchung gewidmet ist, liegt in der Notwendigkeit, effiziente Strategien zur Entwicklung, Integration und Verwaltung von Microfrontends zu identifizieren und zu verstehen. Angesichts der Vielfalt der Ansätze, die derzeit auf dem Markt verfügbar sind, stehen Entwickler vor der Herausforderung, den für ihre Anforderungen geeigneten Weg zu finden, um Microfrontends in bestehende oder neue Webanwendungen zu integrieren.

Die Skalierbarkeit von Webanwendungen, insbesondere in großen Organisationen oder Projekten, kann rasch zu einer Kapazitätsbeschränkung führen. Das Management divergenter Teams, die unabhängig voneinander an unterschiedlichen Teilen der Anwendung arbeiten, kann komplex sein und zu Problemen bei der Koordination, Konsistenz und Effizienz füh-

ren [1] (vgl. Geers 2020: 4).

Zusätzlich sind Webanwendungen oft gezwungen, auf verschiedenen Plattformen oder in unterschiedlichen Technologiestapeln zu arbeiten. Dies kann zu Inkonsistenzen, Performance-Problemen und mangelnder Wartbarkeit führen [6] (vgl. Mezzalira 2021: 14).

1.2 Zielsetzung

Das Hauptziel dieser Masterarbeit ist es, verschiedene Ansätze zur Implementierung von Microfrontends zu analysieren und miteinander zu vergleichen, um fundierte Erkenntnisse und Empfehlungen für die Auswahl und Umsetzung von Microfrontend-Architekturen in modernen Webanwendungen zu liefern.

Die vorliegende Arbeit kann Entwicklern, Web-Architekten und technischen Entscheidungsträgern ein umfassendes Verständnis der Konzepte und Praktiken von Microfrontends vermitteln. Außerdem kann es helfen, die Stärken und Schwächen verschiedener Ansätze im Hinblick auf die spezifischen Anforderungen Ihres Projekts zu verstehen und den besten Ansatz zur Implementierung von Microfrontends auszuwählen.

1.3 Forschungsfragen

Nach der Darstellung des Problembereichs, des Kontexts und der angestrebten Ziele lautet die zentrale Fragestellung dieser Untersuchung: Wie können verschiedene Ansätze zur Implementierung von Microfrontends identifiziert, verstanden und miteinander verglichen werden, um Entwicklern und Technologieentscheidern die Auswahl des am besten geeigneten Ansatzes für ihre spezifischen Anforderungen zu ermöglichen?

Zu diesem Zweck werden die folgenden Forschungsfragen gestellt, die mithilfe dieser Arbeit beantwortet werden sollen.

1. Wann ist es sinnvoll, Microfrontends in Betracht zu ziehen, und was sind die Vor- und Nachteile ?
2. Inwiefern unterscheiden sich die verschiedenen Ansätze zur Implementierung von Microfrontends voneinander, und welche dieser Ansätze gelten als die effektivsten ?
3. Wie können diese Ansätze Webanwendungen im Frontend verbessern?
4. Welche Erfahrungen haben Unternehmen gemacht, die bereits Microfrontends implementiert haben, und welche Empfehlungen können daraus abgeleitet werden?

1.4 Struktur der Arbeit

Am Anfang dieser Arbeit werden in Kapitel 2 die Evolution von Webarchitekturen untersucht und legt die Grundlagen für die Klassifizierung der folgenden Kapitel, indem es sich auf die Analyse der verfügbaren Architekturen auf dem Markt konzentriert und einen kurzen Überblick über jede Architektur bietet. Mithilfe dieses Schrittes wird ein haltbareres Verständnis für verschiedene Architekturoptionen geschaffen und Entscheidungen

hinsichtlich der Migration von Unternehmen und Entwicklern zu neuen Architekturen besser durchdrungen.

Das dritte Kapitel widmet sich der Analyse der Microfrontend Architektur. In diesem Abschnitt werden die Gründe für die Verwendung der Microfrontend Architektur erläutert und die Herausforderungen diskutiert, die Entwickler durch den monolithischen Ansatz im Frontend-Bereich haben. Ferner werden die Ursachen für die Umstellung auf eine Microfrontend Architektur untersucht. Der Abschnitt 3.2 bietet ein umfassendes Verständnis der Microfrontend-Architektur, wobei der Schwerpunkt auf der grundlegenden Idee und dem Hauptziel dieser Architektur liegt. In den folgenden Abschnitten dieses Kapitels werden Entscheidungen, die vor Beginn eines Projekts mit der Microfrontend-Architektur getroffen werden müssen, beleuchtet. Ebenso werden die Aspekte, in denen die Verwendung dieser Architektur in Projekten nützlich sein kann, sowie die detaillierten Vor- und Nachteile dieser Architektur ausführlich erläutert.

Kapitel 4 untersucht und analysiert die verfügbaren Ansätze zur Entwicklung von Microfrontend Architektur. Das Hauptziel dieses Kapitels besteht darin, Architekten, Entwicklern und Unternehmen bei der Verbesserung ihres Verständnisses der Schwierigkeiten zu helfen, den für ihre Anforderungen nützlichen Weg zu finden, um Microfrontends in bestehende oder neue Webanwendungen zu integrieren. Architekten und Entwickler können in diesem Kapitel die geeigneten Ansätze auswählen, die gemäß den Zielen ihrer Projekte am besten funktionieren.

Schließlich befasst sich das fünfte Kapitel mit der Anwendung der Microfrontend Architektur in Unternehmen wie Spotify AB, Zalando SE, Netflix Inc. und IKEA. Beendet wird die Masterarbeit mit einer Diskussion mit der Architektur von Microfrontends und den identifizierten Methoden zur Umsetzung dieser Architektur, um in Kapitel 7 ein Fazit der Arbeit zu ziehen, die Forschungsfragen zu beantworten und einen Ausblick auf die zukünftigen Entwicklungen zu bieten.

1.5 Related Work

Pavlenko et al. (2020) [3] berichten in ihrem Artikel über die Verwendung von Micro-Frontends im Konzept von Single Page Applications und haben in diesem Kontext eine Fallstudie zur Implementierung eines „Education Hub Systems“ entworfen. Hierbei handelt es sich um eine Anwendung, welche Online-Kurse von verschiedenen Anbietern zusammenfasst, um den Nutzern als zentrale Anlaufstelle und Suchmaschine für ihre individuellen Bedürfnisse und Wünsche zu dienen [3] (vgl. Pavlenko et al. 2020: 56).

Pavlenko et al. (2020) haben das Single-Spa [20] Framework verwendet, um ihre Microfrontend Architektur aufzubauen, wobei React.js [16] als SPA-Framework für die gesamte Architektur verwendet wurde. Sie berichten, dass das Routing eines der Hauptanliegen in SPA ist.

Zusätzlich wurde festgestellt, dass die Verwendung der Microfrontend-Architektur besonders für komplexe Großprojekte und große Teams erfahrener Entwickler geeignet ist, im Gegensatz zu kleinen Projekten oder solchen, an denen nur wenige Entwickler beteiligt

sind. Dies liegt daran, dass die Entwickler sich nicht auf die Wartung der Architektur, sondern sich auf die Entwicklung von Funktionen konzentrieren, was die Entwicklungszeit verkürzt.

Yang et al. (2019) [13] präsentieren in ihrem Artikel „Research and Application of Microfrontends“ ein Content Management System (CMS), welches das „Mooa-Framework“ [21] für die Microfrontendarchitektur verwendet. Das „Mooa Framework“ [21] ist ein Microfrontendframework, das Angular Dienste anbietet und die Master-Slave Architektur nutzt. Es handelt sich hierbei um eine Microfrontendlösung, die auf „Single-SPA“ [20] basiert und speziell für den IE 10 und iFrames optimiert ist. Die Forschenden berichten, dass die Trennung von Frontend- und Backend-Anwendungen eine herausragende Benutzererfahrung ermöglicht, aber dazu führt, dass eine SPA nicht gut skaliert und bereitgestellt werden kann und die Wartung schwierig ist. Bei der Bereitstellung einer neuen Version eines Microfrontends muss lediglich die apps.json-Datei der Hauptanwendung auf die neueste Konfigurationsdatei verweisen, um die Hauptanwendung über die Bereitstellung des neuen Microfrontends zu informieren.

Somit haben Yang et al. (2019) herausgefunden, dass das auf Microfrontends basierende Design des Contentmanagement Systems (CMS) Teams ermöglicht, unabhängig voneinander zu entwickeln, schnell zu deployen und individuelle Tests durchzuführen. Dies ermöglicht „continuous integration, continuous deployment, and continuous delivery“ [13] (Yang et al: 2019: 5). Außerdem postulieren sie, dass Microfrontend Technologie noch in der Entwicklungsphase und nicht ausreichend ausgereift sei. Es sei demzufolge notwendig, weitere reife praktische Anwendungen zu erforschen.

Mena et al. (2019) [14] stellen in ihrem Artikel eine Progressive Web Application (PWA) im Zusammenhang mit Microfrontend vor. Die PWA enthält einen Algorithmus, der es dem Benutzer ermöglicht, Komponenten basierend auf seinem Kontext auszuwählen. Sowohl Microservices als auch die Microfrontendarchitektur werden von Mena et al. (2019) [14] für diese Anwendung genutzt. Die hauptsächliche Konzentration lag darauf, eine Webanwendung mit Microservices zu entwickeln, anstatt dies in der Microfrontend-Architektur zu berücksichtigen. Trotzdem wurde festgestellt, dass Microfrontend die Möglichkeit bietet, die Benutzeroberfläche dynamisch zu gestalten und visuelle Elemente unabhängig voneinander zu entwickeln. Dabei wurden verschiedene Möglichkeiten gefunden, die Benutzerdaten anzuzeigen.

2 Evolution von Web Architekturen

Webarchitekturen spielen eine entscheidende Rolle bei der Entwicklung von Webanwendungen und Websites. Sie bieten eine strukturierte und organisierte Methode, um verschiedene Komponenten einer Webanwendung miteinander zu verbinden und zu koordinieren.

Dieses vorliegende Kapitel untersucht und stellt die Evolution von Webarchitekturen in den letzten Jahrzehnten vor. Sämtliche Arten der Architektur werden in diesem Kapitel bündig ausgeführt, um einen Überblick über die Entwicklungen in Webarchitekturen zu geben und die Gründe zu beschreiben, warum Forscher und Entwickler von älteren Architekturen zu neueren übergehen.

2.1 Dreischichtige Architektur

Unter der Dreischichtenarchitektur wird ein verwendetes Softwarearchitekturmodell im Bereich der Webentwicklung verstanden. Durch diese Architektur können Softwareanwendungen in drei logische und physische Schichten gegliedert werden: die Darstellungsschicht, die Anwendungsschicht und die Datenschicht. Jede Schicht hat ihre eigenen Aufgaben und Verantwortungen, was die Anwendungslogik von den Daten trennt [22] (vgl. IBM o. J.: o. S.). Die unten aufgeführte Abbildung 1 verdeutlicht diese Differenzierung.

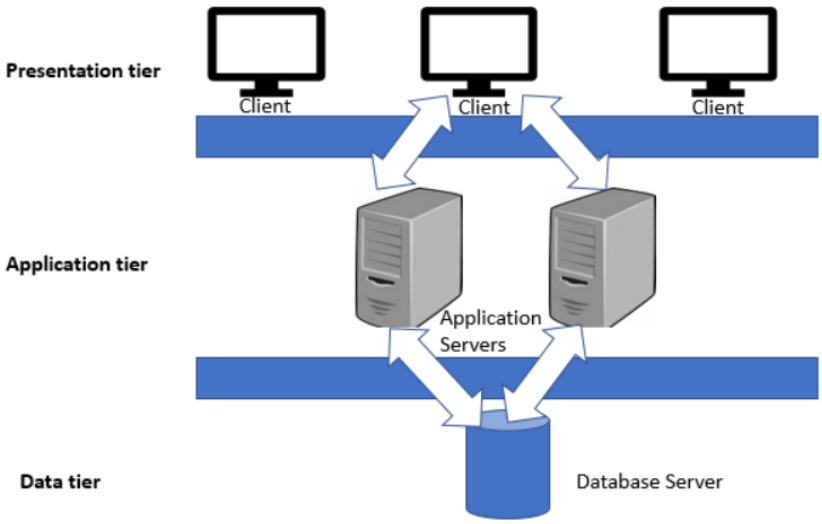

Abbildung 1 Diagramm der dreischichtigen Architektur
(Quelle: [23] O'Reilly o. J.)

Die Darstellungsschicht ist für die Veranschaulichung der Informationen und die Interaktion mit dem Benutzer verantwortlich. Sie umfasst die Benutzerschnittstelle, die für die Darstellung der Daten und die Entgegennahme der Benutzereingaben verantwortlich ist. Die Anwendungsschicht enthält die Anwendungslogik, die die spezifischen Funktionen und Geschäftsregeln der Anwendung implementiert. Hier werden die Anfragen der Präsentationsschicht verarbeitet und die entsprechenden Aktionen ausgeführt. Die Datenschicht ist für den Zugriff auf die Datenbank oder andere Datenspeicher zuständig. Sie kümmert sich

um die Speicherung, Abfrage und Aktualisierung der Daten gemäß den Anforderungen der Anwendungsschicht [22] (vgl. IBM o. J.: o. S.).

In der dreischichtigen Architektur gibt es keine direkte Verbindung zwischen der Datenschicht und der Präsentationsschicht, da die gesamte Kommunikation über die Anwendungsschicht läuft. Diese Eigenschaft erhöht die Sicherheit der Architektur. Auf der anderen Seite entwickeln verschiedene Teams jede dieser Schichten unabhängig voneinander. Das arrangiert es den Teams, die für die Projektziele am haltbarsten geeigneten Programmiersprachen und Werkzeuge zu verwenden. Folglich wird nicht nur eine schnellere Entwicklung ermöglicht, sondern auch eine bessere Skalierbarkeit der Schichten ohne direkten Einfluss auf andere Schichten [22] (vgl. IBM o. J.: o. S.).

2.2 Web Serviceprotokoll

Heutzutage sind Entwickler in der Lage, Websites und Apps mit einer Vielzahl von Programmiersprachen, Architekturen und Plattformen zu erstellen. Trotzdem stellt das Teilen von Daten zwischen diesen Technologien eine große Herausforderung in diesem Bereich dar, da die Unterschiede in den Datenformaten zwischen diesen Technologien zu Problemen führen. Die Vielzahl von Datenformaten macht den Informationsaustausch komplizierter und stellt Entwickler vor Anpassungsprobleme [25] (vgl. Amazon Web Services o. J.: o. S.).

So scheint Web Service Protokoll eine der wichtigsten Lösungen im Bereich der Informationstechnologie zu sein, um die Probleme des Datenaustauschs zu bewältigen [25] (vgl. Amazon Web Services o. J.: o. S.). Es gibt diverse Arten dieser Architektur wie exemplarisch RPC (gRPC), REST, SOAP und GraphQL [26] (vgl. Kornienko et al. 2021: 3).

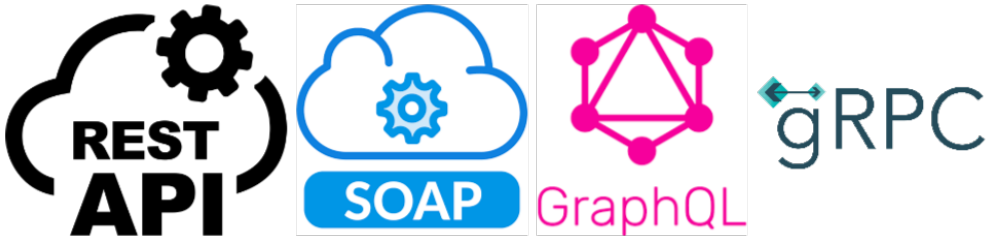

Abbildung 2 Typen des Webserviceprotokolls

Das Hauptziel aller oben genannten Methoden besteht darin, Informationen mithilfe von APIs zu kommunizieren. „APIs are mechanisms that enable two software components to communicate with each other using a set of definitions and protocols.“[27] (vgl. Amazon Web Services o. J.: o. S.)

2.2.1 REST-ful HTTP

Fielding (2000: o. S.) stellt den Representational State Transfer (REST) in seiner Dissertation mit dem Titel „Architectural Styles and the Design of Network-based Software Architectures“ vor [28] (Fielding 2000: o. S.). Gemäß Sunyaev (2020: 184) „Representational State Transfer (REST) describes an architectural style for designing loosely coupled

applications over HTTP.“[29] (Sunyaev 2020: 184)

Das Ziel von Fielding (2000) bei der Erstellung von REST war es, einen Standard zu schaffen, der es Servern ermöglichte, von verschiedenen Orten aus miteinander zu kommunizieren. Vor der Entwicklung und Einführung von REST waren die Entwickler angehalten, APIs mit anderen Techniken wie SOAP-Webservices zu integrieren [30] (vgl. Gillis 2021: o. S.).

„When REST is used als Basis for Web Services, these Web Services are referred to as RESTful.“[29] (Sunyaev 2020: 184) Ein RESTful-Web-Service wird hingegen ebenso als RESTful API oder auch REST API genannt [30] (vgl. Gillis 2021: o. S.).

Client-Server, Stateless, Cache, Uniform Interface, Layered System und Code-on-Demand sind sechs von Fielding (2000) beschriebene Einschränkungen, die helfen, den REST-Architekturstil zu definieren und RESTful-Web-Services zu beschreiben [25, 26, 28, 29] (vgl. Aws-Amazon o. J.: o. S.; Kornienko et al. 2021: 3; Fielding 2000: o. S.; Sunyaev 2020: 185).

Dieser Architekturstil, der auf den Prinzipien des HTTP-Protocols basiert, hat sich nach der Vorstellung von REST durch Fielding (2000) zu einem der am häufigsten verwendeten Stilen in der Entwicklung von Web-Services und APIs entwickelt. Tatsächlich war Fielding an der Entwicklung des HTTP-Protocols beteiligt und wird als einer der Erfinder betrachtet [26] (vgl. Kornienko et al. 2021: 4).

Die Anfragen werden in einer RESTful-API an eine URI gesendet, indem HTTP Standardmethoden verwendet werden, um zwischen Server und Client zu kommunizieren [25, 29] (vgl. Aws-Amazon o. J.: o. S.; Sunyaev 2020: 186).

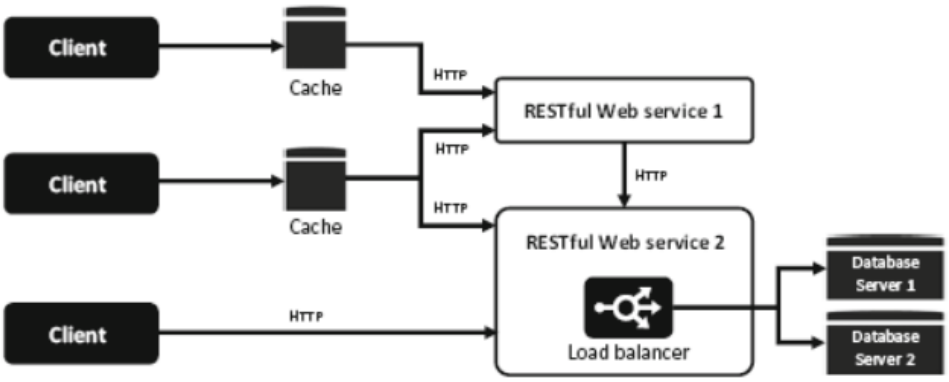

Abbildung 3 Konzeptionelle Architektur von RESTful Web Services
(Quelle: [29] Sunyaev 2020: 186)

In der Regel werden diese HTTP-Methoden in vier Abschnitte unterteilt [29] (vgl. Sunyaev 2020: 187): GET dient dem Abrufen von Ressourcen, POST dem Erstellen neuer Ressourcen. PUT ermöglicht die Aktualisierung oder Änderung des Zustands einer Ressource und kann auch zur Erstellung neuer Ressourcen angewandt werden. DELETE wird verwendet,

um Ressourcen zu löschen.

Nachdem die Kommunikation über das HTTP-Protocol abgeschlossen ist, kann eine RESTful API die Daten als Antwort in verschiedenen Formaten wie XML und JSON übertragen [25, 26, 29] (vgl. Aws-Amazon o. J.: o. S.; Kornienko et al. 2021: 4; Sunyaev 2020: 186).

2.2.2 SOAP

Ein SOAP-Webdienst, auch als großformatiger Webdienst bekannt, fungiert als Kommunikationsprotokoll zwischen Anwendungen über verschiedene Netzwerke einschließlich des Internets. Es ist ein anerkannter Industriestandard des W3C, der XML (eXtensible Markup Language) als interoperables Format verwendet. Die verfügbaren Funktionen und Ressourcen werden durch maschinenlesbare Dienstbeschreibungen veröffentlicht, die auf der Web Services Description Language (WSDL) basieren. SOAP nutzt die Protokolle HTTP, SMTP und FTP für seine Funktionen [26, 29] (vgl. Kornienko et al. 2021: 4; Sunyaev 2020: 177).

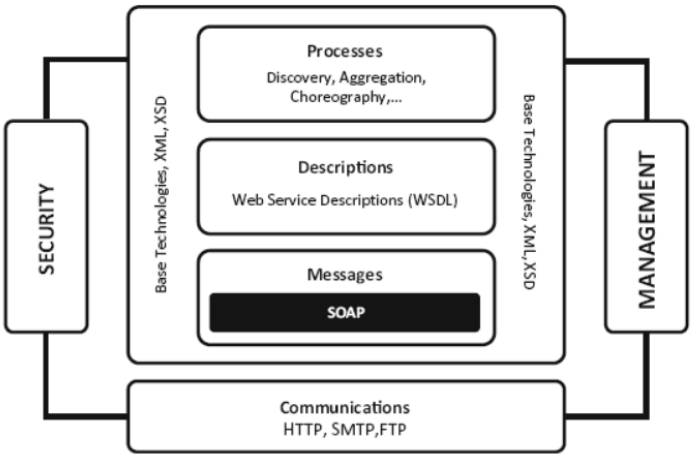

Abbildung 4 Die Architektur Stapel für SOAP Web service
(Quelle: [29] Sunyaev 2020: 177)

Die Architektur eines SOAP-Webdienstes unterstützt synchrone und asynchrone Interaktionen zwischen Clients und Diensten und bietet zusätzliche Funktionen wie Routenführung und Transaktionsverarbeitung. Das SOAP-Nachrichtenprotokoll wurde ursprünglich als Simple Object Access Protokoll konzipiert und wird vorwiegend in Unternehmen eingesetzt, um zuverlässigen Zugriff innerhalb des Unternehmens oder direkt mit einem Client zu ermöglichen. Die ausgeprägten Sicherheitsfunktionen machen SOAP insbesondere im Unternehmens- und Finanzsektor zur bevorzugten Wahl [26, 29] (vgl. Kornienko et al. 2021: 4; Sunyaev 2020: 177).

2.2.3 gRPC

gRPC handelt es sich um ein modernes, quelloffenes und hochleistungsfähiges Remote Procedure Call (RPC) Framework, das in jeder Umgebung laufen kann [31] (vgl. grpc.io o. J.:

o. S.). Es wurde im Jahr 2015 von der Firma Google entwickelt und erhält häufig Anerkennung als Open Source Framework, was bedeutet, dass jeder den Quellcode anpassen oder verbessern kann, um seine Bedürfnisse und Ziele zu erfüllen [31] (vgl. grpc.io o. J.: o. S.). Andererseits können Entwickler aufgrund der Standards, die es bei der Programmierung bietet, signifikant Zeit einsparen [32] (vgl. IONOS 2020: o. S.). gRPC basiert auf HTTP/2 und verwendet Protokoll Buffers zur Serialisierung von Daten. Üblich werden diese Protokoll-Buffers in einer Textdatei mit der Dateiendung „.proto“ gespeichert [33] (vgl. groc.io o. J.: o. S.).

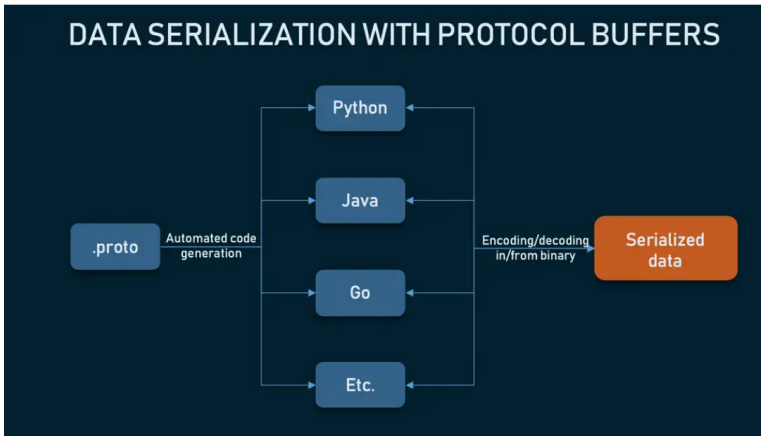

Abbildung 5 Protokollpuffer als eine beliebte Serialisierungsmethode außerhalb von gRPC Anwendungsfällen
(Quelle: [34] altexsoft o. J.: o. S.)

gRPC kann von einer Vielzahl von Programmiersprachen unterstützt und entwickelt werden, darunter Java, Kotlin, C++, Node, Pyton und andere [33] (vgl. groc.io o. J.: o. S.). Viele der größten Unternehmen weltweit, wie Netflix, Cisco, Square, CoreOS und Cockroach Labs, verwenden gRPC [31](vgl. grpc.io o. J.: o. S.).

2.3 Backend Architekturen

2.3.1 Monolithische Architektur

"Monolithic Architecture (MA), instead, was the traditional approach to software development, used in the past by large companies like Amazon and Ebay."[35] (Lauretis 2019: 93). Eine Anwendung in der monolithischen Architektur entwickelt sich zu einer einzigen umfassenden und zusammenhängenden Einheit, die von anderen Anwendungen unabhängig ist [36] (vgl. Chandler o. J.: o. S.).

Die gesamte Anwendung ist in einer monolithischen Architektur auf einer einzigen Codebasis aufgebaut, die alle Funktionen und Komponenten beinhaltet, und als Ganzes bereitstellt [39] (vgl. Ren et al. 2018: 1). Ahmad (2023: o. S.) zufolge sind in einer monolithischen Architektur die Komponenten unabhängig voneinander verbunden [40] (vgl. 2023: o. S.). Daher ist es nicht möglich, die Module separat voneinander auszuführen [41] (vgl. Dragoni et al. 2017: 196). Diese Art der Architektur ist eng verbunden, und die gesamte Logik zur Bearbeitung einer Anfrage wird in einem einzigen Prozess durchgeführt [42] (vgl. Blinowski et al. 2022: 20358).

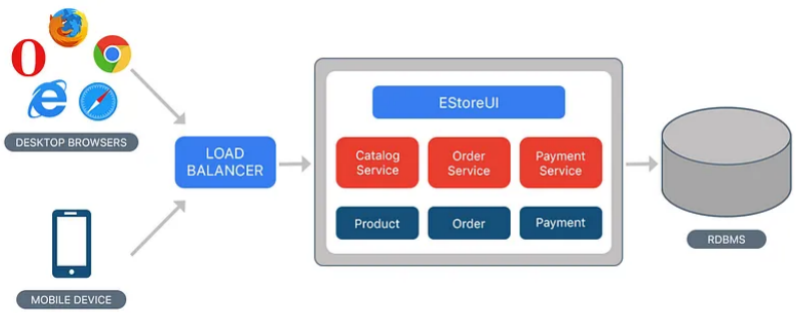

Abbildung 6 Monolithische Architektur
(Quelle: [56] Haq 2018: o. S.)

Wenn sie nicht kompliziert sind, haben monolithische Anwendungen ihre eigenen Stärken, wie die Einfachheit bei der Entwicklung, dem Testen und der Bereitstellung [35] (vgl. Lauretis 2019: 93). Wenn sie jedoch zu groß oder kompliziert werden, kann dies die Wartung und Bereitstellung beeinträchtigen [43] (vgl. Abgaz et al. 2023: 4214).

Anwendungsfälle

Gemäß Kalske et al. (2018: 35) und dem Avenga Team (2022: o. S.) ist die monolithische Architektur aufgrund ihrer Einfachheit und leichten Umsetzbarkeit eine geeignete Option, um mit der Entwicklung von Anwendungen zu beginnen [37, 45] (vgl. Kalske et al. 2018: 35; Avenga Team 2022: o. S.). Des Weiteren, wie Benavente et al. (2022) in ihrem Artikel erklären, wird eine monolithische Architektur normalerweise für Anwendungen verwendet, die eine begrenzte Menge an Informationen verarbeiten und eine einzige Programmiersprache oder Datenbank verwenden [38] (vgl. Benavente et al. 2022: 177). Daher werden in diesem Abschnitt einige Anwendungsfälle vorgestellt, in denen eine monolithische Architektur eine

nützliche Option sein könnte.

Nach Villamizar et al. (2015) stellt die monolithische Architektur eine große Herausforderung für die Skalierbarkeit dar [44] (vgl. Villamizar et al. 2015: 583). Aus diesem Grund ist sie für kleinere Projekte geeignet, die eine begrenzte Skalierbarkeit benötigen [46, 47, 48] (vgl. Beznos 2023: o. S.; Ozkaya 2023: o. S.; Sarwar 2020: o. S.).

In Fällen, in denen das Projekt so schnell wie möglich abgeschlossen werden muss und eine schnelle Markteinführung wichtig ist, kann die monolithische Architektur eine geeignete Option sein [45, 46, 47] (vgl. Avenga Team 2022: o. S.; Beznos 2023: o. S.; Ozkaya 2023: o. S.).

Die monolithische Architektur ist eine geeignete Wahl für Projekte, die klein und einfach sind, begrenzte funktionale Anforderungen haben und das Entwicklungsteam klein ist und das Projekt später keine Upgrade-Anforderungen hat, wie das Hinzufügen neuer Funktionen oder das Aktualisieren auf neue Technologien [45, 47, 48] (vgl. Avenga Team 2022: o. S.; Beznos 2023: o. S.; Ozkaya 2023: o. S.).

Laut Lauretis (2019: 93) ist es sinnvoll, die monolithische Architektur für Anwendungen zu verwenden, die nicht sehr komplex sind. Die monolithische Architektur ermöglicht eine einfachere Entwicklung, Testung und Bereitstellung [35] (vgl. Lauretis 2019: 93).

Vor- und Nachteile

Obwohl die monolithische Architektur ein traditionelles Modell darstellt, behält sie aufgrund ihrer zahlreichen Vorteile in der Softwareentwicklung nach wie vor ihre Relevanz. Dennoch sind ihre Nachteile insbesondere in Bezug auf die Skalierbarkeit unbestreitbar. Um ein tieferes Verständnis dieser Architektur zu erlangen, werden im Folgenden die Vor- und Nachteile genauer untersucht.

Die monolithische Architektur bietet zwei wesentliche Vorteile: Bereitstellung und Entwicklung. In diesem Ansatz wird die Anwendung auf einer alleinigen Codebasis aufgebaut, was den Entwicklungsprozess aufgrund der geringeren Modularität vereinfacht. Aufgrund der reduzierten Komplexität des Programms, bedingt durch weniger Komponenten und einer einzigen Datei für das gesamte Programm, entfällt die Notwendigkeit der Verwaltung mehrerer Bereitstellung. Das Kopieren der gepackten Anwendung auf einen Server gestaltet sich einfach, was ein unkompliziertes und schnelles Bereitstellung ermöglicht [36, 45, 46, 48, 49, 52] (vgl. Harris o. J.: o. S.; Avenga Team 2022: o. S.; Beznos 2023: o. S.; Sarwar 2020: o. S.; Davis: 2023: o. S.; Kharenko 2015: o. S.).

Wie zuvor erwähnt, basiert eine Anwendung in einer monolithischen Architektur auf einer einzigen Codebasis. Diese Konsolidierung aller Codes an einem Ort vereinfacht den Debugging-Prozess erheblich und ermöglicht eine schnelle Durchführung von End-to-End-Tests [36, 45, 46, 48, 49, 52] (vgl. Harris o. J.: o. S.; Avenga Team 2022: o. S., Beznos 2023: o. S.; Sarwar 2020: o. S.; Davis: 2023: o. S.,; Kharenko 2015: o. S.).

Die monolithische Architektur könnte aufgrund der oben genannten Anwendungsfälle und Vorteile eine geeignete Option für kleine Programme sein. Jedoch treten Probleme auf,

wenn eine Anwendung wächst und es eine Herausforderung wird, eine effektive Skalierbarkeit zu erreichen. Ein weiterer Nachteil dieser Architektur wird im Folgenden erläutert.

Es ist möglich, dass die gesamte Anwendung erneut integriert, neu erstellt, erneut getestet und nach der Bereitstellung neu gestartet werden muss, wenn sich ein Modul in einer monolithischen Struktur ändert [43] (vgl. Abgaz et al. 2023: 4214). Dies macht Aktualisierungen schwierig und zeitintensiv [36] (vgl. Harris o. J.: o. S.). Die gesamte Anwendung muss skaliert oder neu bereitgestellt werden, wenn eine einzelne Komponente aktualisiert oder skaliert werden muss [40] (vgl. Ahmad 2023: o. S.).

Die Skalierung einer monolithischen Anwendung ist immer schwieriger als die Skalierung von Microservices, da die gesamte Anwendung und die gesamte Codebasis skaliert und bereitgestellt werden müssen, anstatt nur ein Teil der Anwendung, welcher mehr Ressourcen benötigt [44] (vgl. Villamizar et al. 2015: 583).

2.3.2 Microservices Architektur

Das vorliegende Kapitel befasst sich mit den Konzepten und Anforderungen von Microservices. Bevor der architektonische Stil von Microservices analysiert werden kann, ist es notwendig, zu definieren, was genau ein Microservice ist. Die Idee des Begriffs Microservice wurde im Jahr 2011 von einer Gruppe von Softwarearchitekten entwickelt und ein Jahr später offiziell auf der 33. Konferenz in Krakau vorgestellt [42] (vgl. Blinowski et al. 2022: 20357). Kritiker argumentieren, dass das Konzept lediglich eine Neupositionierung von SOA sei [51] (vgl. Richardson 2015: o. S.).

Dieser architektonische Stil existierte bereits, obwohl er unter verschiedenen Namen bekannt war [41] (vgl. Dragoni et al. 2017: 201). Netflix begann 2009 mit Microservices, als der Begriff „Microservice“ noch nicht existierte [42] (vgl. Blinowski et al. 2022: 20357), von einer monolithischen Architektur zu einer Microservices Architektur, die sie als „fine-grained Service-Oriented Architecture“ bezeichnete, zu migrieren [53] (Wang und Tonse 2013: o. S.).

Lewis und Fowler (2014: o. S.) beschreiben den architektonischen Stil der Microservices in ihrem Blogbeitrag und erklärten diesen neuen architektonischen Begriff als: „an approach to developing a single application as a suite of small services, each running in its own process and communicating with lightweight mechanisms, often an HTTP resource API.“[50] (Lewis und Fowler 2014: o. S.)

Tatsächlich führte diese Definition von Lewis und Fowler im Jahr 2014, dass Microservices in großen Organisationen wie Amazon, eBay, Zalando, Spotify, Uber, Airbnb, LinkedIn, Twitter, Groupon und Coca-Cola Beachtung fand. Dies wiederum führte dazu, dass große Unternehmen von einer monolithischen Architektur zu einer Microservices Architektur migrierten, um die Beliebtheit dieses Architekturmodells zu steigern [42] (vgl. Blinowski et al. 2022: 20357 ff.).

Laut Ponce et al. (2019: 1) und Kalske et al. (2018: 32) können traditionelle monolithische Architekturen die Anforderungen an Skalierbarkeit, Code-Eigentümerschaft und schnelle

Entwicklungszyklen nicht erfüllen, was zu der Neigung großer Organisationen zu Microservices führt [54, 37] (vgl. Ponce et al. 2019: 1; Kalske et al. 2018: 32). Microservices hingegen ermöglichen eine effektivere Verwaltung komplexer Anwendungen [38] (vgl. Benavente et al. 2022: 177). Darüber hinaus hat das Microservices Architekturmuster einen erheblichen Einfluss auf die Beziehung zwischen Anwendungen und Datenbanken. Jeder Microservice hat seine eigene Datenbankstruktur anstelle einer gemeinsamen Datenbankstruktur mit anderen Diensten [52] (vgl. Kharenko 2015: o. S.).

Abbildung 7 Microservices Architectur
(Quelle: [56] Haq 2018: o. S.)

In seinem Blog betont Kharenko (2015: o. S.) auch, dass jeder Service eine eigene Datenbankstruktur benötigt, um Microservices effektiv zu nutzen. Dieser Schritt gewährleistet eine Loose Coupling [52] (vgl. Kharenko 2015: o. S.). Die Verwendung einer Loose Coupling ermöglicht es, Änderungen in den Services autark voneinander vorzunehmen, ohne dass dies Auswirkungen auf den übrigen Code hat, da Microservices nicht miteinander verbunden sind und somit unabhängig voneinander skaliert und erweitert werden können [37] (vgl. Kalske et al. 2018: 33).

Gemäß Nunes et al. (2019: 38 ff.) müssen Organisationen drei Schritte durchführen, um von einer monolithischen Architektur zu einer Microservices Architektur zu migrieren. Diese umfassen die Datensammlung, die Identifizierung der Microservices und deren Visualisierung. In der Datensammlungsphase müssen Informationen über das System gesammelt werden, das die Migration zur Microservices-Architektur plant. Die Datensammlung kann entweder manuell oder automatisch erfolgen. Bei der automatischen Datensammlung werden Techniken wie statische und dynamische Analyse verwendet, um verschiedene Arten von Informationen zu erhalten. In der Identifikationsphase werden die gesammelten Daten verwendet, um Kandidaten für Microservices auszuwählen. Nach der Identifikation von Microservices wird eine Metrik für die gesammelten Daten definiert. Diese Metrik wird verwendet, um ein Maß der Ähnlichkeit zwischen den Systemkomponenten zu bestimmen. Nach der Identifizierung der Microservices ermöglicht die Visualisierungsphase eine visuelle Darstellung, basierend auf verschiedenen Perspektiven der Microservices und ihrer Beziehungen [55] (vgl. Nunes et al. 2019: 38 ff.).

Obwohl es Unsicherheiten bezüglich der Microservices Architektur gibt, hat diese Architektur große Vorteile, insbesondere in Zeiten, in denen agile Entwicklung und die Bereit-

stellung komplexer Unternehmensanwendungen wichtig sind [51] (vgl. Richard-son 2015: o. S.). Darüber hinaus hat dieser Architekturstil die Herausforderungen der monolithischen Architektur teilweise gelöst und zu einer Umstellung von monolithischen Architekturen auf Microservices geführt [35] (vgl. Lauretis 2019: 94). Jedoch sollten ebenso die Nachteile dieser Architektur nicht ignoriert werden. Um ein besseres Verständnis von Microservices zu gewinnen, wird im Folgenden eine kurze Erläuterung der Vor- und Nachteile gegeben.

Vor- und Nachteile

Da jeder Service unabhängig entwickelt wird, ermöglichen Microservices auch eine agile Entwicklung. Entwickler können die Technologien auswählen, die den Anforderungen eines bestimmten Microservices am besten entsprechen. Dies ermöglicht die Integration einer Vielzahl von Programmiersprachen, Frameworks und Datenbanken in einer einzigen Anwendung [36, 49, 52] (vgl. Harris o. J.: o. S.; Davis 2023: o. S.; Kharenko 2015: o. S.). Wenn der Service die API-Vereinbarung einhält, haben die Programmierer die freie Wahl zwischen verschiedenen Technologien [51] (vgl. Richardson 2015: o. S.).

Microservices ermöglichen es, jeden Service unabhängig voneinander zu skalieren. Dies reduziert die Ressourcennutzung und ermöglicht die Skalierung der Anwendung auf die notwendigen Komponenten. Anstatt jeden Service und die gesamte Anwendung währenddessen herunterzufahren, können einzelne Services nach Bedarf angepasst und aktualisiert werden [36, 49, 51, 52] (vgl. Harris o. J.: o. S.; Davis 2023: o. S.; Richardson 2015: o. S., Kharenko 2015: o. S.).

Die Wartbarkeit und Testbarkeit von Microservices sind zwei der wichtigsten Vorteile [36] (vgl. Harris o. J.: o. S.). Anwendungen werden durch die Verwendung der Microservices Architektur in kleine, unabhängige Services aufgeteilt. Diese Services sind klein, um Entwicklern die Entwicklung und das Verständnis des Codes zu erleichtern. Diese Aufteilung verbessert die Wartbarkeit des Codes in Microservices [35] (vgl. Lauretis 2019: 94). Teams können auch neue Funktionen ausprobieren und bei Bedarf ohne Schwierigkeiten zur vorherigen Version zurückkehren [36] (vgl. Harris o. J.: o. S.).

Ein weiterer Vorteil von Microservices ist ihre Fähigkeit, autark voneinander bereitgestellt zu werden. Diese Architektur ermöglicht es, jeden Microservice unabhängig von einem bestimmten Team anzubieten. Durch diese Methode ist eine schnelle und einfache Bereitstellung individueller Funktionen möglich, was Continuous Delivery und Deployment komplexer Anwendungen ermöglicht [36, 51, 52, 56] (vgl. Harris o. J.: o. S.; Richardson 2015: o. S.; Kharenko 2015: o. S.; Haq 2018: o. S.).

Wie bereits erwähnt, helfen Microservices, große Programme in kleine Services aufzuteilen. Basierend auf diesem Prinzip kann behauptet werden, dass ein weiterer Vorteil von Microservices in ihrer Austauschbarkeit liegt. Da einzelne Microservices klein sind, ist es einfach, sie durch eine verbesserte Implementierung oder sogar durch ihre Löschung zu ersetzen. Teams, die Microservices nutzen, haben normalerweise keine Schwierigkeiten oder sogar keine Probleme, sie bei Bedarf vollständig neu zu schreiben [35] (vgl. Lauretis 2019: 94).

Einer der Hauptnachteile von Microservices liegt in ihrer Art der Kommunikation. Blinow-

ski et al. (2022: 20359) weisen in ihrem Artikel darauf hin, dass es praktisch keine Standards für die Kommunikationsmechanismen oder den Transport von Microservices gibt. Um eine Verbindung herzustellen, müssen Microservices leichtgewichtige Internetprotokolle oder Messaging-Protokolle verwenden [42] (vgl. Blinowski et al. 2022: 20359). Andererseits betonen Kalske et al. (2018: 34) in ihrem Artikel, dass Fehler bei der Kommunikation dazu führen können, dass Microservices ihre Unabhängigkeit verlieren, was die Hauptvorteile des gesamten Ansatzes verringert. Sie betonen auch, dass bei zu viel Genauigkeit in den Services die Kommunikation zwischen mehreren Microservices funktionale Probleme verursachen kann [37] (vgl. Kalske et al. 2018: 34).

Ein großer Nachteil von Microservices sind Zeit und Kosten. In der Regel erfordert die Implementierung von Microservices eine Vielzahl von Tests, Startskripten, Hosting Infrastrukturen, Überwachungstools und Ressourcen. Dies kann zu höheren Kosten führen, insbesondere bei Projekten, die kleiner sind. Die Unabhängigkeit jedes Microservices kann den Entwicklungsprozess beschleunigen, aber es kann auch seine Geschwindigkeit verlangsamen. Dies liegt daran, dass die Umsetzung einer Vielzahl von Microservices und ihrer Organisation in Gruppen erheblich mehr Zeit in Anspruch nehmen kann und eine präzise Teamkoordination erfordern kann [36, 49, 52] (vgl. Harris 2020: o. S.; Davis 2023: o. S.; Haq 2018: o. S.).

Die Komplexität von Microservices ist ein weiterer großer Nachteil, insbesondere in Bezug auf die Bereitstellung. Microservices-Anwendungen sind ein verteiltes System und müssen vor der Einführung in eine Anwendungssoftware integriert werden, was zu dieser Komplexität führt [49, 52] (vgl. Davis 2023: o. S.; Kharenko 2015: o. S.). Obwohl in vorherigen Abschnitten die Testbarkeit von Microservices als Vorteil genannt wurde, kann sie gleichzeitig eine Herausforderung darstellen. Denn das Testen in einer Microservices-Architektur ist wesentlich komplexer als in einer monolithischen Architektur. Jeder Service in einer Microservices-Anwendung muss separat getestet werden, und um dies zu ermöglichen, müssen sowohl dieser Service als auch alle davon abhängigen Services gestartet werden [49, 52, 56] (vgl. Davis 2023: o. S.; Kharenko 2015: o. S.; Haq 2018: o. S.).

2.4 Frontend Architekturen

Wie bereits in der Einleitung erwähnt, spielen Webanwendungen heute eine wichtige Rolle und sind ein integraler Bestandteil unseres täglichen Lebens. Derzeit ist es entscheidend, eine passende Architektur für sowohl das Backend als auch das Frontend zu wählen, da die Verwendung einer angemessenen Architektur die Skalierbarkeit verbessert, die Wartung vereinfacht und die Flexibilität der Anwendungen erhöht. Dieses Unterkapitel untersucht Frontend-Architekturen und konzentriert sich auf Single und Multi Page Applications. Das hierauf folgende Kapitel befasst sich eingehend mit der Microfrontend Architektur.

Vor der Einführung der Microfrontend Architektur im Jahr 2016 verwendeten die meisten Entwickler Architekturen für Single oder Multi Page Application, um Webanwendungen zu erstellen [58] (vgl. Ly 2022: o. S.). Diese Architekturen haben sich zu Standards für die Entwicklung und Organisation von Frontend-Webanwendungen entwickelt. Microfrontends basieren auf ähnlichen Prinzipien wie Microservices, da sie es ermöglichen, die Benutzeroberfläche in separate und unabhängige Teile zu unterteilen. Jeder Abschnitt kann durch diese Unterteilungen unabhängig entwickelt, getestet und implementiert werden [3] (vgl. Pavlenko 2020: 54).

Obwohl SPA heutzutage in der Webentwicklung und der Anwendung sehr beliebt sind, kann man nicht eindeutig behaupten, dass sie für alle Projekte die beste Wahl sind [58] (vgl. Ly 2022: o. S.). Viele Faktoren beeinflussen die Entscheidung zwischen diesen Architekturen. Die Auswahl einer Frontend Architektur hängt nicht nur von den spezifischen Anforderungen jedes Projekts und dessen Anwendungsbereich ab, sondern auch von den verfügbaren Ressourcen und dem Entwicklerteam. Im Laufe der Zeit wird diese richtige Entscheidung die Qualität der Anwendung und die Wartbarkeit beeinflussen [57, 58] (vgl. Kaur 2021: o. S.; Ly 2022: o. S.).

2.4.1 Multi Page Application

Multi Page Applications (MPA) sind Anwendungen, bei denen jede Seite unabhängig vom Server erstellt und an den Browser gesendet wird [57, 59] (vgl. Kaur 2021: o. S., Ghosh 2023: o. S.). Aus diesem Grund benötigt die Übertragung dieser Seiten viele Daten, wobei diese Programme viele Links und komplexe Benutzeroberflächen verfügen [57, 58] (vgl. Kaur 2021: o. S.; Ly 2022: o. S.). Der Hauptunterschied zwischen den MPA und SPA ist, dass MPAs sich auf den Server verlassen, um Seiten zu generieren und wiederherzustellen, während SPAs den JavaScript verwenden, um Seiten im Browser zu erstellen [59] (vgl. Ghosh 2023: o. S.).

In Multi Page Applications (MPAs) erfolgt die Funktionsweise so, dass der Server bei jeder Aktion eines Benutzers wie dem Klicken auf einen Link oder dem Ausfüllen eines Formulars eine neue Seite mit aktualisiertem Inhalt generiert und als Antwort an den Browser sendet. Diese ständige Verbindung zum Server ist entscheidend, da sie sicherstellt, dass Benutzer

die neuesten Informationen auf der Seite angezeigt bekommen [59] (Ghosh 2023: o. S.).

Anwendungsfälle

Webseiten, die Änderungen vornehmen müssen, wenn Benutzer von einer Seite zur nächsten navigieren, wie unter anderem Amazon oder eBay, profitieren von MPA. In solchen Websites sind die Flexibilität und die Fähigkeit, Änderungen auf den Seiten vorzunehmen, während sich die Benutzer bewegen, entscheidend. Aus diesem Grund verwenden Entwickler MPA, um diese Websites zu erstellen [58, 59] (vgl. Ly 2022: o. S.; Ghosh 2023: o. S.).

Vor- und Nachteile

Im Vergleich zu Single-Page Applications (SPAs) bieten Multi-Page Applications (MPAs) eine bessere SEO-Unterstützung, was einen der Hauptvorteile darstellt. Diese Architektur betrachtet jede Anfrage als separate HTML-Seite, wodurch der Server für jede Benutzeranfrage eine separate HTML-Seite sendet und im Browser lädt. Die Vielzahl von Seiten ermöglicht eine bessere Kontrolle über die SEO. Darüber hinaus können Entwickler effektiv Meta-Tags mit Schlüsselwörtern auf jeder Seite hinzufügen und erstellen. Diese Maßnahmen verbessern die Website-Optimierung und steigern gleichzeitig ihre Position in den Suchergebnissen [58, 59] (vgl. Ly 2022: o. S.; Ghosh 2023: o. S.).

Ein weiterer Vorteil von MPA ist die Skalierbarkeit. MPA ermöglicht es, neue Seiten basierend auf den Inhalten und den Benutzeranforderungen ohne Einschränkungen zu erstellen. Diese Freiheit bei der Erstellung neuer Seiten ermöglicht es Entwicklern, diese unabhängig und separat zu erweitern, was zu einer besseren Skalierbarkeit und Flexibilität des Projekts beiträgt [58, 59] (vgl. Ly 2022: o. S.; Ghosh 2023: o. S.).

Darüber hinaus bietet MPA im Bereich der Datenanalyse Vorteile. Durch die Verwaltung der Seiten auf dem Server verwenden diese Programme zahlreiche Analysetools, die sich auf den Server stützen und nicht auf den Browser. Diese Tools liefern nützliche Informationen über die Website, wie z. B. funktionierende Funktionen, Nutzerverhalten auf den einzelnen Seiten usw. Dadurch haben MPA-Analysewerkzeuge eine bessere Leistung im Vergleich zu SPAs, bei denen nur begrenzte Daten wie Besucherzahlen und Aufenthaltsdauer auf der Website verfügbar sind [58, 59] (vgl. Ly 2022: o. S.; Ghosh 2023: o. S.).

Ein potenzieller Nachteil von Multi-Page Applications liegt in ihrer Leistungsfähigkeit. Obwohl sie aufgrund der Möglichkeit einer großen Anzahl von Seiten eine bessere Skalierbarkeit bieten können, kann diese Vielzahl von Seiten die Gesamtleistung beeinträchtigen. Bei jeder HTTP-Anfrage an den Server muss der Browser die gesamte Seite und alle zugehörigen Ressourcen herunterladen, einschließlich häufig verwendeter Elemente wie Header und Footer. Dieser Prozess kann zu einer Verringerung der Leistung und einer Verlangsamung der Website führen, insbesondere wenn die Internetverbindung des Benutzers langsamer als gewöhnlich ist, was dazu führt, dass das Laden dieser Programme mehr Zeit in Anspruch nimmt [58, 59] (vgl. Ly 2022: o. S.; Ghosh 2023: o. S.).

Ein weiterer Aspekt, der berücksichtigt werden muss, ist der Zeit- und Kostenfaktor bei der Entwicklung von MPAs im Vergleich zu Single-Page Applications. Da in MPAs jede Seite

unabhängig erstellt wird, erfordert dies mehr Entwicklungszeit. Es besteht die Möglichkeit, dass Teile des Codes in einigen Projekten erneut geschrieben werden müssen. Im Gegensatz zu SPAs, die hauptsächlich auf Client-Ressourcen wie Speicher und Prozessor angewiesen sind, sind MPAs stärker auf Server-Ressourcen angewiesen. Dies kann zu höheren Kosten für Hosting und Wartung von Servern führen, was in einigen Projekten möglicherweise nicht kosteneffizient ist [58, 59] (vgl. Ly 2022: o. S.; Ghosh 2023: o. S.).

2.4.2 Single Page Application

Ein weiterer Ansatz im Bereich der Frontend Architektur ist die Single Page Application (SPA), die in diesem Abschnitt behandelt wird. Im Gegensatz zu traditionellen MPAs, die für jede Ansicht oder Seite eine separate HTML-Seite laden, beschreibt die offizielle Website von MDN die SPA als eine Webanwendung, die lediglich eine HTML-Datei im Browser lädt [60] (vgl. MDN Web Docs o. J.: o. S.). Um neue Informationen anzuzeigen, ohne die gesamte Seite neu zu laden, sendet der Server nach dem Laden der Hauptseite nur die angeforderten Daten des Kunden. Diese Aktualisierungen werden normalerweise im Voraus heruntergeladen und erfolgen dynamisch [6, 11] (vgl. Mezzalira 2021: 3; Peltonen et al. 2021: 2). Durch das Nichtladen der gesamten Seiten wird die Leistung verbessert und eine dynamische Benutzererfahrung ermöglicht [61] (vgl. SPA o. J.: o. S.).

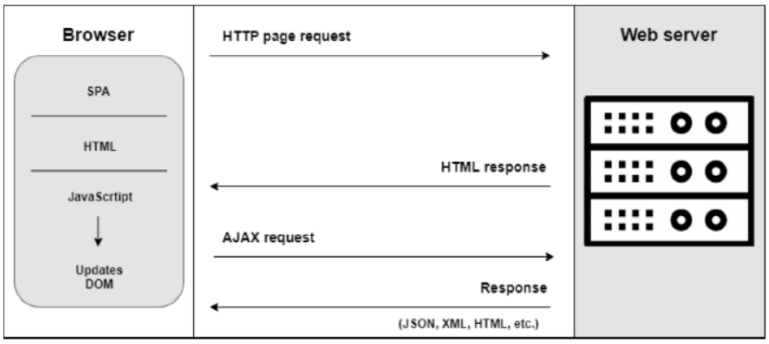

Abbildung 8 Kommunikationszyklus zwischen Client und Server in einer SPA
(Quelle: [11] Peltonen et al. 2021: 3)

Ajax ist die Grundlage eines SPA [58] (vgl. Ly 2022: o. S.) und bezeichnet eine Webprogrammiertechnologie [62] (vgl. Ryte o. J., o. S.), die es ermöglicht, die Seite asynchron zu aktualisieren, ohne sie neu zu laden. Dadurch ist es möglich, Daten asynchron zwischen dem Server und dem Client zu übertragen und zu empfangen [63] (vgl. Akil 2021: o. S.).

Wenn es notwendig ist, den Benutzern zusätzliche Informationen bereitzustellen und den Inhalt der Seite zu ändern, zum Beispiel wenn ein Besucher die Website öffnet, die zwar grundlegende Inhalte enthält, jedoch möglicherweise nicht alle Informationen, die die Anwendung benötigt. Der Benutzer interagiert dann mit der Anwendung, indem er ein Formular ausfüllt oder auf einen Link klickt. Der Browser stellt eine Anfrage an den Server und erhält als Antwort eine HTML-Datei. Bei Verwendung einer SPA-Anwendung übermittelt der Server lediglich beim ersten Aufruf eine HTML-Datei. Daraufhin sendet die

Anwendung nach jeder Verbindung eine AJAX-Anfrage an den Server. Der Server bearbeitet die Anfrage und sendet JSON-Daten als Antwort [57] (vgl. Kaur 2021: o. S.). Der JavaScript-Code auf der Client-Seite verwendet die erhaltenen JSON-Daten, um das Document-Object-Model (DOM) dynamisch zu aktualisieren. Das DOM muss durch die Verwendung von JavaScript und HTTP-Anfragen dynamisch aktualisiert werden [11] (vgl. Peltonen et al. 2021: 2).

Heutzutage erstellen Entwickler Single Page Applications (SPA) häufig mit Javascript-Frameworks wie Angular.js, Vue.js und Bibliotheken wie React.js [16, 17, 18] (vgl. React o. J.: o. S.; Angular o. J.: o. S.; vuejs.org o. J.: o. S.). Durch die Verwendung dieser Tools wird nicht nur der Entwicklungsprozess vereinfacht, sondern auch die Geschwindigkeit der Erstellung und Entwicklung dieser Anwendungen erhöht [64] (vgl. Digital Vitarag 2023: o. S.). Frameworks bieten eine Vielzahl von Werkzeugen und Komponenten, die für die Anwendungsentwicklung bereitgestellt werden können [65] (vgl. BasuMallick 2023: o. S.). Darüber hinaus bieten diese Frameworks eine Abstraktionsschicht für die DOM-Manipulation bereit [11] (vgl. Peltonen et al. 2021: 2).

Anwendungsfälle

Nach Erörterung dieser Architektur werden nun einige Beispiele für ihre Anwendung untersucht. Der Fokus dieses Abschnitts liegt darin, eine Zusammenfassung der Situationen zu bieten, in denen diese Architektur für Projekte eingesetzt werden könnte. Wie in den Veröffentlichungen von Kaur (2021: o. S.) und Ly (2022: o. S.) festgehalten, haben bedeutende Organisationen, darunter Trello, Gmail, Twitter, Google und Facebook, diese Architektur in ihren Websites implementiert [57, 58] (vgl. Kaur 2021: o. S.; Ly 2022: o. S.).

Die Projektspezifikationen bestimmen letztendlich die Anwendbarkeit dieser Architektur. Insbesondere erweist sich diese Architektur als ideal für Software-as-a-Service (SaaS)-Plattformen und soziale Netzwerke, bei denen die Suchmaschinenoptimierung keine ausschlaggebende Rolle spielt. Beispiele für gelungene Implementierungen dieser Architektur sind Google Maps und Gmail. Es wird jedoch davon abgeraten, diese Architektur zu wählen, wenn ein hohes Sicherheitsniveau und eine entscheidende Bedeutung der Suchmaschinenoptimierung im Projekt vorliegen [57, 58] (vgl. Kaur 2021: o. S.; Ly 2022: o. S.).

Vor- und Nachteile

Die Verwendung von Single Page Applications (SPA) bietet zahlreiche Vorteile, aber wie bei anderen Architekturen birgt sie auch Nachteile. Im Folgenden werden einige der Vor- und Nachteile dieser Architektur erläutert.

Einer der prominentesten Vorteile betrifft die Leistung. Wie bereits erwähnt, lädt eine SPA nur den Teil der Seite herunter, der vom Benutzer angefordert wird, im Gegensatz zu Multi Page Applications (MPA), die bei jeder Anfrage die gesamte Seite laden müssen. Dies führt dazu, dass SPAs eine höhere Geschwindigkeit und Leistung aufweisen und schneller auf Benutzeranfragen reagieren können [57, 58] (vgl. Kaur 2021: o. S.; Ly 2022: o. S.).

Ein weiterer Vorteil liegt in der Verbesserung der Benutzererfahrung. Da SPAs nur die

Kundenanfragen über den Server senden und nicht die gesamte Seite ändern, wird die Zeit, die benötigt wird, um die Seite zu aktualisieren, erheblich verkürzt. Die Benutzererfahrung wird verbessert, indem die Reaktionszeit auf Anfragen verkürzt wird, da Benutzer keinen neuen Seitenladeprozess beobachten müssen [57] (vgl. Kaur 2021: o. S.). Zusätzlich ermöglicht die Architektur von SPAs die Speicherung von Daten und die Nutzung geringerer Bandbreite. Da eine SPA nur eine Anfrage an den Server sendet und dann alle empfangenen lokalen Daten im Cache speichert, können Benutzer auch im Offline-Modus nahtlos weiterarbeiten [58] (vgl. Ly 2022: o. S.). Dies führt zu einer effizienteren Nutzung der Bandbreite, da SPAs Webseiten nur einmal laden und dann lokal speichern [57] (vgl. Kaur 2021: o. S.).

Ein weiterer Vorteil von SPAs ist die Leichtigkeit beim Debuggen. Da SPAs stark auf Frameworks und Bibliotheken basieren und häufig unter Verwendung dieser entwickelt werden, bieten die meisten dieser Frameworks ihre eigenen Entwicklertools an [66, 67, 68] (vgl. React Developer Tools o. J.: o. S., DevTools Overview o. J.: o. S., Vue DevTools o. J.: o. S.), die in Browsern wie Chrome, Firefox und Edge installiert werden können. Dadurch wird es einfach, Fehler in dieser Architektur zu beheben, da diese Tools den Zustand des Programms überwachen können, ohne große Schwierigkeiten zu verursachen [58] (vgl. Ly 2022: o. S.).

Nach der Darlegung einiger Vorzüge von Single Page Applications ist es angebracht, ebenso ihre Schwächen in Betracht zu ziehen. Zu den fundamentalen Einschränkungen dieser Architektur zählt die mangelnde Unterstützung für Suchmaschinenoptimierung (SEO). Da SPAs nur eine Seite dynamisch basierend auf Kundenanfragen laden, gestaltet sich die effektive Optimierung verschiedener Seiten für Suchmaschinen als herausfordernd, was die SEO-Möglichkeiten für Websites beeinträchtigen kann [57] (vgl. Kaur 2021: o. S.).

Ein weiterer Nachteil betrifft die Sicherheit. Laut Kaur (2021: o. S.) ist diese Architektur anfälliger für skriptbasierte Angriffe im Vergleich zu Multi Page Applications (MPA) [57] (vgl. Kaur 2021: o. S.). Ebenso unterstreicht der Autor Ly (2022: o. S.) in seinem Blog, dass unter Verwendung von XSS Hacker clientseitige Skripte in Webanwendungen einschleusen können [58] (vgl. Ly 2022: o. S.). Beide Quellen warnen davor, dass diese Architektur möglicherweise sensible Daten für alle Benutzer sichtbar machen kann [57, 58] (vgl. Kaur 2021: o. S.; Ly 2022: o. S.).

3 Microfrontends in Webanwendungen

In diesem vorliegenden Kapitel erfolgt eine Analyse der Motivation und des Ziels der Entwicklung der Microfrontend-Architektur. Dabei wird das Konzept der Microfrontend Architektur vorgestellt und Entscheidungen im Zusammenhang mit dieser Architektur vor Beginn des Projekts erläutert. Zudem wird ein kurzer Überblick über hilfreiche Frameworks für Microfrontend Entwickler gegeben. Es werden verschiedene Szenarien untersucht, in denen die Verwendung von Microfrontend angemessen oder ungeeignet erscheint. Abschließend werden sowohl die Vor- als auch die Nachteile dieser Architektur erörtert.

3.1 Motivation für den Einsatz von Microfrontends

Gemäß Geers (2020: 1) erscheint das Erstellen einer qualitativ hochwertigen Webanwendung in kleinen Projekten, insbesondere wenn die Entwicklung von nur wenigen Entwicklern durchgeführt wird, nicht als eine herausfordernde oder komplizierte Aufgabe [1] (vgl. Geers 2020:1). Mezzalira (2021: 15) schlägt vor, einen monolithischen Ansatz für den Frontend-Bereich in Betracht zu ziehen [6] (vgl. Mezzalira 2021: 15). Dies liegt daran, dass nach Kalske et al. (2018: 35) eine monolithische Anwendung aus drei Schichten besteht: dem „UI-Layer“, der „business logic layer“ und der „data access layer“ [37] (Kalske et al. 2018: 35). Diese drei Schichten, wie in der Abbildung dargestellt, stellen eine Verbindung zu einer Datenbank her [37] (vgl. Kalske et al. 2018: 35).

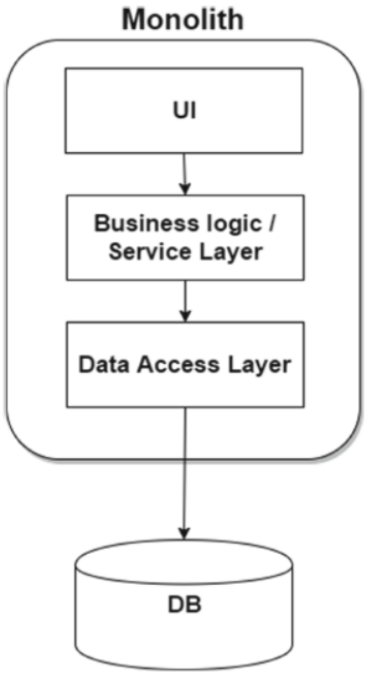

Abbildung 9 Monolithische Anwendung mit drei Schichten
(Quelle: [37] Kalske et al.2018: 36)

Mezzalira (2021: 15) betont, dass ein monolithischer Ansatz für den Frontend-Bereich in Betracht gezogen werden sollte [6] (vgl. Mezzalira 2021: 15), da Architekten und Softwareentwickler beim Erstellen des Frontends Zugriff auf eine Vielzahl von Architekturoptionen

haben, darunter Single Page Applications (SPAs), serverseitig gerenderte Anwendungen oder Anwendungen, die durch statische HTML-Dateien erstellt werden. Sie betonen jedoch, dass diese Architekturen sich im Laufe der Zeit und mit zunehmenden Anforderungen des Projekts in Richtung einer monolithischen Architektur entwickeln können [11] (vgl. Peltonen et al. 2021: 3).

Angenommen, das Unternehmen wächst im Laufe der Zeit, und die einfache Webanwendung erfüllt nicht mehr die Anforderungen. Es besteht die Möglichkeit, dass neue Funktionen hinzugefügt oder Verbesserungen an der Webanwendung vorgenommen werden möchten [1] (vgl. Geers 2020: 1). Es ist offensichtlich, dass der monolithische Ansatz es nicht ermöglicht, die Webanwendung effektiv zu verbessern, insbesondere wenn auf Plattformen gearbeitet wird, die voraussichtlich über Jahre hinweg für Benutzer verfügbar sein sollen [6] (vgl. Mezzalira 2021: 15).

Mit dem Wachstum des Projekts stellen sich auch verschiedene Herausforderungen ein. Die Projektgröße und das Entwicklungsteam nehmen zu, was es schwierig macht, alle Aspekte des Systems zu identifizieren. Dies führt zu einer erhöhten Komplexität der Anwendung und erschwert interne Teamdiskussionen. Darüber hinaus können Änderungen in einem Teil des Systems unerwartete Auswirkungen auf andere Teile haben [1] (vgl. Geers 2020: 4), was die Koordination zwischen Entwicklungsteams beeinträchtigt. Mit der Zunahme des Code-Umfangs nimmt auch die Abhängigkeit von der Anwendung zu, was die Verwaltung dieser Verbindungen und Abhängigkeiten zu einer komplexen Aufgabe macht [11] (vgl. Peltonen et al. 2021: 3).

Nach Mezzalira (2021: 14) wurde in den letzten Jahren im Bereich Frontend festgestellt, dass die Notwendigkeit der Skalierbarkeit von Webanwendungen aufgrund verschiedener Gründe nicht besonders stark betont wurde und nur begrenzte Optionen in diesem Bereich verfügbar waren. Jedoch suchen Benutzer heutzutage aufgrund verschiedener Veränderungen und neuer Anforderungen bei der Nutzung von Webplattformen nach einer verbesserten Benutzererfahrung, die mehr Interaktivität und bessere Interaktionen bietet. Daher benötigen Webanwendungen Maßnahmen, um die Benutzererfahrung zu verbessern und qualitativ hochwertige Dienstleistungen anzubieten [6] (vgl. Mezzalira 2021: 14).

Jackson (2019: o. S.) betont in seinem Blog, dass es äußerst schwierig ist, ein robustes Frontend zu entwickeln, insbesondere wenn mehrere Teams gleichzeitig an einem großen, komplexen Produkt arbeiten müssen. Dies erhöht die Komplexität der Frontend-Entwicklung [4] (vgl. Jackson 2019: o. S.). Geers (2017: o. S.) stellt fest, dass die Wartung und Verwaltung schwieriger werden, wenn die Frontend-Schicht von verschiedenen Teams entwickelt wird, was die Komplexität und die Herausforderungen der Frontend-Entwicklung, insbesondere bei der Zusammenarbeit größerer Teams, verdeutlicht [5] (vgl. Geers 2017: o. S.).

Wie bereits im Abschnitt zu Microservices erwähnt, wurde der Architekturstil der Microservices im Jahr 2014 von Lewis und Fowler eingeführt [50] (vgl. Lewis und Fowler 2014: o. S.). Um die Einschränkungen großer und monolithischer Backendsysteme zu überwinden, haben Organisationen Microservices in ihre Systemarchitekturen integriert. Heutzutage scheint die Entwicklung komplexer Webanwendungen eine umfassende Architektur und

Organisationsstruktur zu erfordern [4] (vgl. Jackson 2019: o. S.).

Gemäß Mezzalira (2021: 15) stellt sich nun die Frage:

„Do we have the opportunity to use a well known pattern or architecture that offers the possibility of adding new features quickly, evolving with the business and delivering part of the application autonomously without big-bang-releases?“[6] (Mezzalira 2021: 15)

Wie von Geers (2020), Mezzalira (2021) und Jackson (2019) festgestellt wurde, kann diese Frage bejaht werden, da diese Anforderung mit Microfrontend definitiv möglich [1, 4, 6] (vgl. Geers 2020: o. S., Jackson 2019: o. S., Mezzalira 2021: o. S.).

3.2 Definition und Konzept von Microfrontends

Peltonen et al. (2021: 3) betonen, dass Entwicklung, schnelle Bereitstellung und einfache Wartung entscheidende Aspekte für die Qualität von Webanwendungen sind. Daher suchen Forscher und Softwareexperten nach Methoden, die die Entwicklung von Webanwendungen unter Berücksichtigung dieser Faktoren ermöglichen. Unternehmen sind ebenfalls gezwungen, neue agile Methoden in ihre Organisationsstrukturen zu integrieren, um schnell auf die Bedürfnisse ihrer Kunden zu reagieren und sich von ihren Konkurrenten abzuheben [11] (vgl. Peltonen et al. 2021: 3).

Taibi und Mezzalira (2022: 25) stellen in ihrem Artikel fest: „modern applications are often built with feature-rich and powerful frontend applications.“[7] (Taibi und Mezzalira 2022: 25). Diese Anwendungen sind auch als single page apps bekannt und befinden sich im oberen Bereich der Microservices Architektur. Geers (2017: o. S.) bezeichnet diese Art von Anwendungen auch als Frontendmonolith [5] (vgl. Geers 2017: o. S.). Peltonen et al. (2021: 4) behaupten, wie in Abbildung 10 gezeigt, dass Frontend-Monolithen horizontale Schichten in den Frontend-Bereich der Anwendung einführen, während Microfrontends darauf abzielen, die Anwendung vertical zu gliedern [11] (vgl. Peltonen et al. 2021: 4).

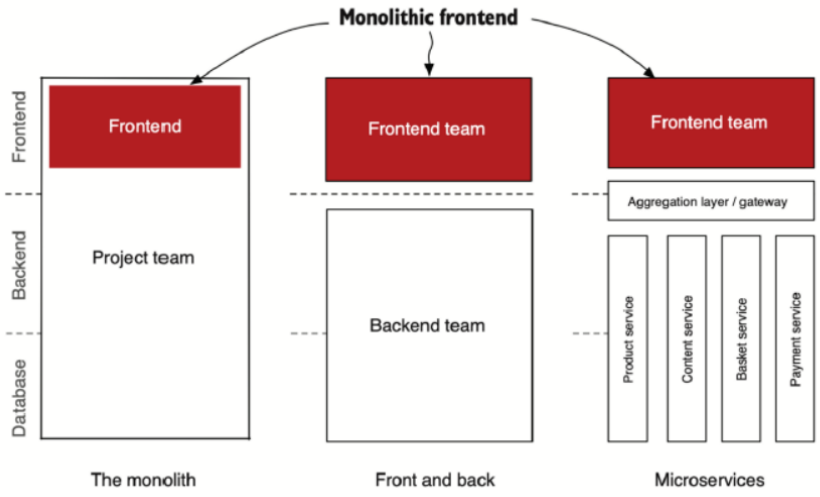

Abbildung 10 Frontend Monolith
(Quelle: [1] Geers 2020: 13)

Der Terminus Micro Frontend wurde Ende des Jahres 2016 von Thoughtworks Technology Radar eingeführt [8]. Yang et al. (2019: 2) stellen in ihrem Artikel fest, dass Microfrontends einen neuen Ansatz für die Entwicklung des Frontends darstellen und die Konzepte von Microservices auf die Frontend-Seite der Anwendung ausdehnen [13] (vgl. Yang et al. 2019: 2).

Geers (2020: 4) hebt hervor, dass die Verwendung von Microfrontends einen alternativen Ansatz für die Frontend-Seite darstellt, indem die Anwendung in verticale Splits unterteilt wird [1] (vgl. Geers 2022: 4). Unter Einsatz von Microfrontends wird jeder Teil der Anwendung von einem spezialisierten Team geleitet und durchgeführt, beginnend von der Datenbank bis zur Benutzeroberfläche [5, 7] (vgl. Geers 2017: o. S.; Taibi und Mezzalira 2022: 25). Des Weiteren beschreibt Geers (2020: 4), dass die verschiedenen Teile der Anwendung von verschiedenen Teams vorbereitet und anschließend im Browser des Kunden integriert werden, um die endgültige Seite zu erstellen und dem Benutzer anzuzeigen [1] (vgl. Geers 2022: 4). Microfrontends und Microservices teilen in der Praxis eine Anwendung in kleine, unabhängige Dienste auf. Der Hauptunterschied besteht darin, dass ein Dienst eine eigene Benutzeroberfläche hat und kein zentrales Team mehr benötigt [1] (vgl. Geers 2022: 4).

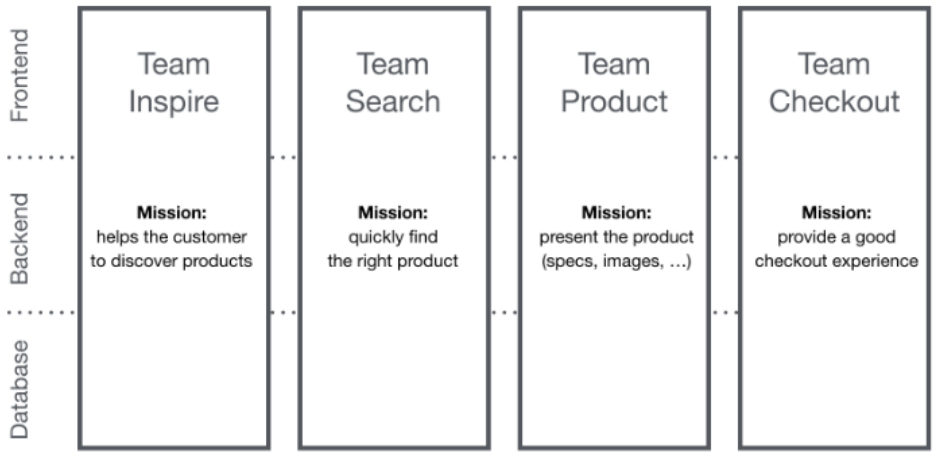

Abbildung 11 Organisation in Verticals End-to-End frontend Teams mit Micro-Frontends Architektur
(Quelle: [5] Geers 2017: o. S.)

Gemäß den Untersuchungen von Peltonen et al. (2021: 4) und Petcu et al. (2023: 323) gliedern Microfrontends das monolithische Design einer Anwendung in verschiedene eigenständige Module oder Komponenten [11, 10] (vgl. Peltonen et al. 2021: 4, Petcu et al. 2023: 323). Diese Module können, wie von Jackson (2019: o. S.) beschrieben, separat entwickelt, getestet und bereitgestellt werden, jedoch erscheinen sie für die Benutzer immer noch als ein einziges Produkt. Jedes Modul übernimmt dabei die Verantwortung für einen spezifischen Bereich der Benutzeroberfläche [4] (vgl. Jackson 2019: o. S.). Jackson (2019: o. S.) definiert Microfrontends auch als „An architectural style where independently deliverable

frontend applications are composed into a greater whole.“[4] (Jackson 2019: o. S.).

Geers (2017: o. S.) stellt auf seiner Website die Behauptung auf, dass es sich bei dieser Idee um keine Innovation im eigentlichen Sinne handele, da sie Parallelen zu dem Konzept der Self-contained-Systems aufweise. Zuvor seien Ansätze wie dieser unter dem Namen Frontend-Integration für vertikalisierte Systeme bekannt [5] (vgl. Geers 2017: o. S.).

3.3 Microfrontend Decision

Dieser Abschnitt erläutert die wesentlichen Entscheidungen zu Beginn eines Projekts mit einer Microfrontend Architektur. Um in Microfrontend Projekten erfolgreich zu sein, ist eine sorgfältige Berücksichtigung der Projektanforderungen, Organisationsstruktur und Entwicklererfahrung entscheidend. Vor dem Projektbeginn können diverse Fragen auftreten, die Probleme bereiten können. Zum Beispiel, wie die Kommunikation zwischen Microfrontends gestaltet werden und wie die Navigation des Benutzers von einer Ansicht zur anderen erfolgen soll [6] (vgl. Mezzalira 2021: 23).

Es gibt verschiedene Ansätze zur Implementierung einer Microfrontend Architektur. Um den besten Ansatz für ein bestimmtes Projekt zu wählen, ist es entscheidend, den Kontext, in dem das Projekt tätig sein wird, genau zu verstehen. Bevor mit einem Microfrontend-Projekt begonnen wird, müssen verschiedene Entscheidungen getroffen werden, die einen bedeutenden Einfluss auf den Projektfortschritt haben können [11] (vgl. Peltonen et al. 2021: 4).

Nach Mezzalira (2021: 24) werden diese Entscheidungen als das „Micro Frontend Decision Framework“[6] (Mezzalira 2021: 24) bezeichnet und setzen sich aus vier grundlegenden Teilen zusammen [6] (vgl. Mezzalira 2021: 24).

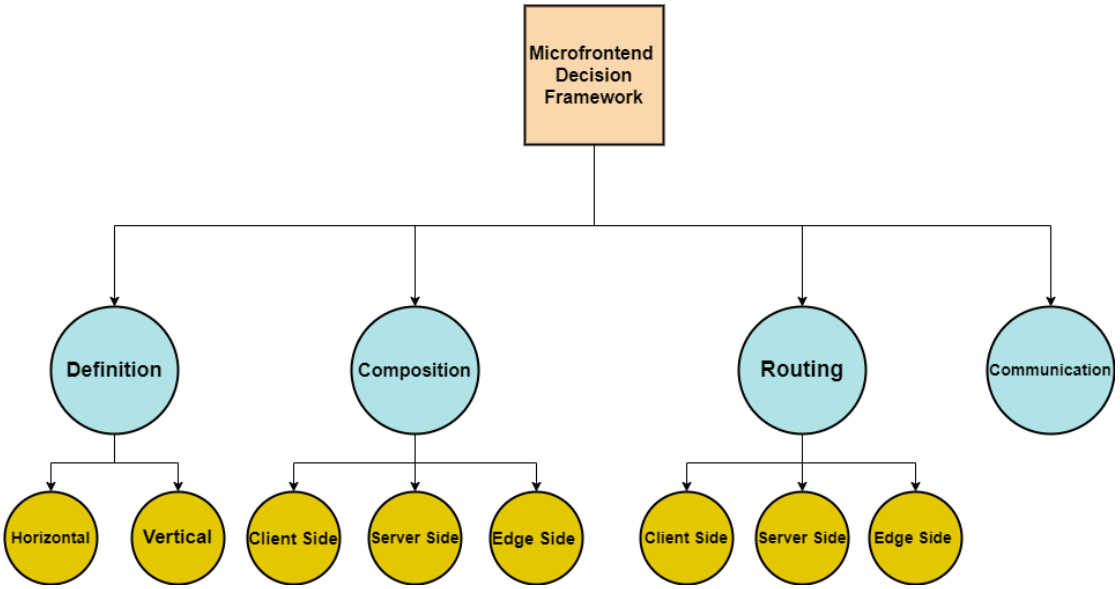

Abbildung 12 Microfrontend Decision Framework

3.3.1 Horizontal oder Vertical split

Tatsächlich kann die erste Entscheidung den Arbeitsablauf erheblich beeinflussen [6] (vgl. Mezzalira 2021: 24). Die wichtigste Entscheidung bei der Definition von Microfrontends besteht darin, die Notwendigkeit zu identifizieren, ein Microfrontend aus technischer Sicht zu betrachten [11] (vgl. Peltonen et al. 2021: 4). Laut einer Definition von Mezzalira (2021: 24) kann die Microfrontend Architektur in zwei Abschnitte unterteilt werden: „Horizontal“

und „Vertical“ Split [6] (vgl. Mezzalira 2021: 24).

Die Definitionen von Horizontal und Vertical Split werden in einem Artikel von Taibi und Mezzalira (2022: 25) wie folgt beschrieben:

- Horizontal Split: „multiple Micro-Frontends in the same view“[7] (Taibi und Mezzalira 2022: 25)
- Vertical Split: „just on Micro-Frontend loaded per time“[7] (Taibi und Mezzalira 2022: 25)

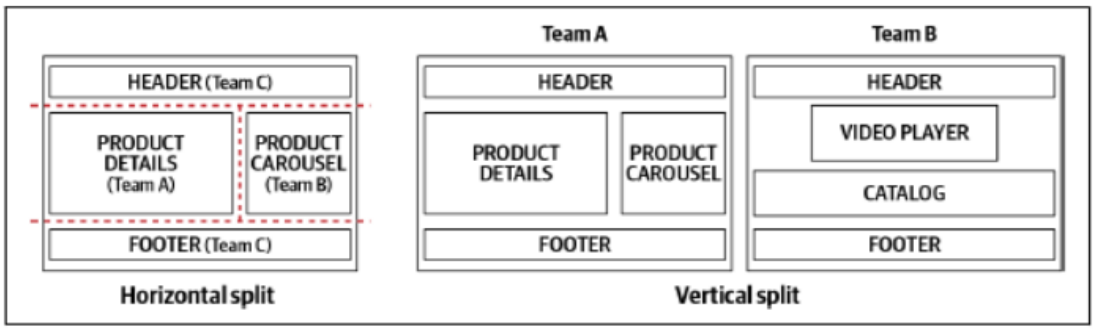

Abbildung 13 Horizontal VS Vertical Split
(Quelle: [6] Mezzalira 2021: 24)

Horizontal Split

Im Anschluss an die vorangegangenen Erklärungen wird zunächst der Horizontal Split behandelt. Hierbei wird deutlich, dass es möglich ist, mehrere Microfrontends auf einer Seite zu platzieren [6, 7, 11] (vgl. Mezzalira 2021: 24; Taibi und Mezzalira 2022: 25; Peltonen et al. 2021: 4). Zur besseren Veranschaulichung könnte gesagt werden, dass jedes Team für einen bestimmten Abschnitt der Seite verantwortlich ist und dass eine Koordination der Bemühungen zwischen den Teams erforderlich ist [6, 11] (vgl. Mezzalira 2021: 24; Peltonen et al. 2021: 4). Der Horizontal Split ermöglicht die Wiederverwendung einiger Microfrontends auf verschiedenen Seiten, was die Flexibilität des Projekts erhöht. Es ist jedoch zu beachten, dass der Horizontal Split eine präzise Organisation und Disziplin erfordert. Andernfalls besteht die Gefahr, dass die Hauptseite des Projekts von einer Vielzahl von Microfrontends beeinträchtigt wird [6] (vgl. Mezzalira 2021: 24).

Eine wichtige Überlegung ist, wann der Horizontal Split in Microfrontends angemessen ist und für welche Projekte er am besten geeignet ist. Mezzalira (2021: 41) stellt fest, dass der Horizontal Split eine praktikable Option ist, insbesondere wenn Business-Subdomains in verschiedenen Ansichten präsentiert werden müssen und die Wiederverwendbarkeit von Subdomains entscheidend ist, beispielsweise für die Suchmaschinenoptimierung (SEO). Der Horizontal Split erweist sich als besonders nützlich und effektiv in Situationen, in denen eine genaue Unterteilung der Subdomains erforderlich ist und ein Frontend-Projekt viele Entwickler involviert [6] (vgl. Mezzalira 2021: 41).

Basierend auf den vorherigen Erläuterungen von Mezzalira (2021: 41) zur Anwendung des Horizontal Split in Microfrontends lässt sich ableiten, dass diese Architektur eine Vielzahl

von Optionen bietet, um den unterschiedlichen Anforderungen von Microfrontend Anwendungen gerecht zu werden. Webarchitekten könnten sich fragen, für welche Projekte der Horizontal Split geeignet ist. Die Definition von Mezzalira (2021: 66) bietet hier eine plausible Lösung.

„Horizontal-split architectures are suggested not only to companies that already have a sizable engineering department but also to projects that have a high level of code reusability, such as a multitenant business-to-business (B2B) project in which one customer requests a customization“[6] (Mezzalira 2021: 66)

Es ist jedoch wichtig zu beachten, dass die Implementierung dieser Horizontal Split Architektur herausfordernd sein kann, da eine strenge Governance und regelmäßige Überprüfungen erforderlich sind, um die Grenzen der Microfrontends einzuhalten. Ebenso besteht das Risiko, dass diese Architektur die Organisationsstruktur herausfordert, wenn sie nicht sorgfältig durchdacht wurde, bevor das Projekt gestartet wurde. Daher ist eine gründliche Untersuchung der Teamstruktur und der Kommunikationsflüsse erforderlich, um den Entwicklern zu helfen, effektiv zu arbeiten und übermäßige externe Abhängigkeiten zwischen den Teams zu vermeiden [6] (vgl. Mezzalira 2021: 66).

Vertical Split

Jeder vertical Split beinhaltet nur ein Microfrontend, das zu einem bestimmten Zeitpunkt geladen wird [7] (vgl. Taibi und Mezzalira 2022: 25). Dies bedeutet, dass jeder vertical Split von einem eigenen Microfrontend-Modul für einen bestimmten Geschäftsbereich verantwortlich ist. Auf diese Weise können verschiedene Teams an verschiedenen Teilen der Anwendung zusammenarbeiten, ohne sich gegenseitig zu beeinträchtigen, beispielsweise bei der Authentifizierung oder Zahlung. Der vertical Split nutzt das Domain-Driven Design (DDD) [11] (vgl. Peltonen et al. 2021: 4). Gemäß Mezzalira (2021: 24) versteht man unter dem Domain-Driven Design (DDD) einen Ansatz zur Softwareentwicklung, welcher die Entwicklung auf die Programmierung eines Domänenmodells richtet, das ein tiefgreifendes Verständnis der Prozesse und Regeln einer Domäne aufweist [6] (vgl. Mezzalira 2022: 24).

Mezzalira (2021: 40) legt nahe, dass die Client Side Option die beste Wahl für Entwickler sei, wenn sie sich für einen Vertical Split entscheiden. Diese Entscheidung scheint aufgrund der vollständigen Konzentration der Frontend-Entwickler auf das Schreiben von Single Page Applications (SPAs) und ihres umfassenden Verständnisses dafür logisch zu sein. Daher kann geschlussfolgert werden, dass genau diese Client Side die beste Option für sie ist. Frontend-Entwickler könnten sich fragen, wie Sie von einem Vertical Split profitieren könnten? [6] (vgl. Mezzalira 2021: 40). Mezzalira (2021: 40) erklärt in seinem Buch „Building Micro-Frontends“ (2021), dass der Vertical Split nützlich sein kann,

„when your project requires a consistent user interface evolution and a fluid user experience across multiple views.“ [6] (Mezzalira 2021: 40).

Die Implementierung eines Microfrontend-Projekts mit einem Vertical Split wird einfacher, da den Entwicklern viele Entscheidungen erleichtert werden. Die Vertical Split Architek-

tur bietet Entwicklern die naheliegendste Erfahrung für eine SPA und ermöglicht es ihnen, Tools, bewährte Techniken und Muster für die Entwicklung eines Microfrontends zu nutzen. Dies ist wahrscheinlich der einfachste Weg für Frontend-Entwickler, mit einem Hintergrund im Frontend, in die Welt der Microfrontends einzutreten [6] (vgl. Mezzalira 2021: 45). Da jedes Team einen Verticalen Schnitt des Programms hat und nicht mehrere Teile über das gesamte Programm verteilt sind, sollte die Koordination zwischen den Teams erleichtert werden, was einer der wichtigsten Vorteile des Vertical Split ist [76] (vgl. Mezzalira 2019: o. S.). Der Vertical Split fragmentiert die Implementierung nicht stark und jedes Team hat einen bestimmten Teil des Programms [75] (vgl. Mezzalira 2020: o. S.).

Gemäß den Ausführungen von Taibi und Mezzalira (2022: 25) hängt die Entscheidung für die Auswahl eines Horizontal Split anstelle eines Vertical Splits von der Art des zu erstellenden Projekts ab. Ein Horizontal Split könnte für statische Seiten wie Kataloge oder E-Commerce besser geeignet sein, während ein Vertical Split für interaktivere Projekte wie eine Online-Streaming-Plattform oder interne Anwendungen, die eine Vertical Split erfordern, geeigneter sein könnte. Sie betonen auch, dass die Fähigkeiten des Teams bei der Entscheidung für einen Horizontal Split oder Vertical Split berücksichtigt werden sollten [7] (vgl. Taibi und Mezzalira 2022: 25).

Taibi und Mezzalira (2022: 25) sind der Ansicht, dass ein Vertical Split besser für traditionelle SPAs geeignet ist, während ein Horizontal Split eine anfängliche Investition erfordert, um eine starke und schnelle Entwicklungserfahrung zu schaffen und seine Rolle zu testen [7] (vgl. Taibi und Mezzalira 2022: 25).

Es wird behauptet, dass die Verwendung eines Vertical Splits dazu führt, dass jedes Team einen bestimmten Bereich des Gesamtprogramms hat. Daher sollte die Teamkoordination einfacher und die Implementierung sollte nicht zu stark fragmentiert sein [75, 76] (vgl. Mezzalira 2020: o. S.; Mezzalira 2019: o. S.). Es ist wichtig zu beachten, dass bei einem Vertical Split nur die Client Side die beste Option ist, während bei einem Horizontal Split alle drei verfügbaren Compositionen (Client, Server, Edge) genutzt werden können [6] (vgl. Mezzalira 2021: 40 ff.).

3.3.2 Composition

Im vorherigen Abschnitt wurden die Konzepte der Horizontal und Vertical Split erläutert. Die zweite Entscheidung betrifft den Ort der Erstellung der Microfrontends. Gemäß Mezzalira (2021: 29), Taibi und Mezzalira (2022: 26-28) und Peltonen et al. (2021: 4) stehen drei Auswahlmöglichkeiten zur Verfügung, um eine Microfrontend-Architektur zu entwickeln [6, 7, 11] (vgl. Mezzalira 2021: 29; Taibi und Mezzalira 2022: 26-28; Peltonen et al. 2021: 4): „Client-Side Composition“, „Edge-Side Composition“ und „Server-Side Composition“ [6, 7, 11] (Mezzalira 2021: 29; Taibi und Mezzalira 2022: 26-28; Peltonen et al. 2021: 4) (siehe Abbildung 14).

Client Side Composition

Die Implementierung von Methoden, bei denen eine Application-Shell einzelne Seitenanwendungen innerhalb ihrer eigenen Struktur lädt, wird als Client Side Composition be-

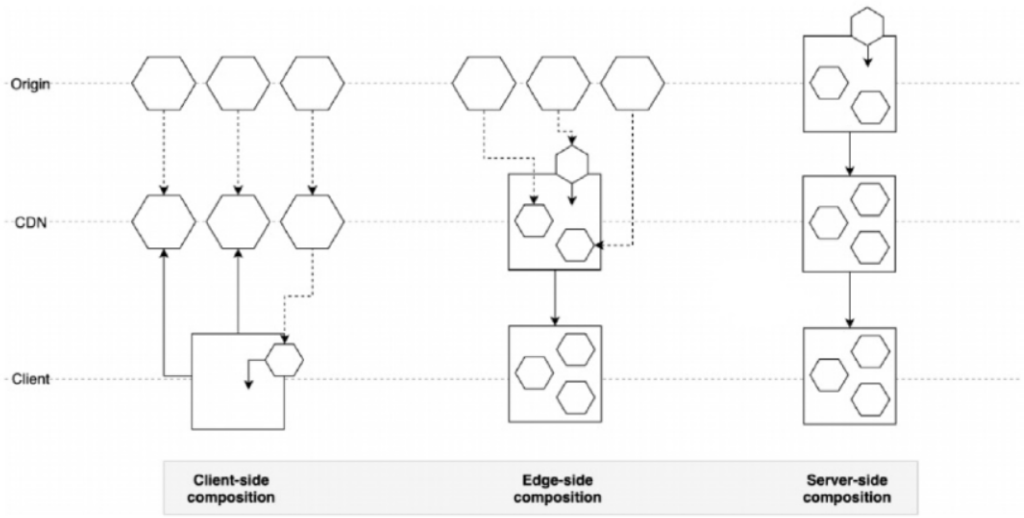

Abbildung 14 Microfrontend Composition
(Quelle: [11] Peltonen et al. 2021: 5)

zeichnet [11] (vgl. Peltonen et al. 2021: 4). Falls das Microfrontend noch nicht auf CDN-Ebene zwischengespeichert ist, kann die Application-Shell mehrere Microfrontends direkt von einem CDN oder vom Ursprungsladen. Damit die Application-Shell dynamisch an den DOM-Knoten angehängt wird, sollten die Microfrontends eine JavaScript- oder HTML-Datei als Einstiegspunkt haben [6] (vgl. Mezzalira 2021: 29).

Client Side Composition ist eine der am häufigsten verwendeten Arten von Microfrontend-Composition aufgrund der Verfügbarkeit von JavaScript-Frontend-Frameworks wie Angular, React, Vue und Svelte [1] (vgl. Geers 2020: 85).

Der wesentliche Vorteil der Client Side Composition liegt darin, dass sie sowohl horizontal als auch vertical gut funktioniert [75] (vgl. Mezzalira 2020: o. S.). Es stellt sich die Frage, wann diese Verwendung von Client Side Composition nützlich ist. Laut Mezzalira (2021: 41) kann dieser Ansatz hilfreich sein, insbesondere wenn das Unternehmen über ein Team verfügt, das mit der Frontend-Umgebung vertraut ist. Ein weiterer Vorteil besteht darin, dass die Client Side Composition in einem Projekt verwendet werden kann, wenn es einem hohen Datenverkehr mit erheblichen Spitzen ausgesetzt ist. Dadurch werden nicht nur Schwierigkeiten bei der Skalierbarkeit auf der Frontend-Ebene vermieden, sondern es können auch Microfrontends nahtlos zwischengespeichert und ein CDN verwendet werden [6] (vgl. Mezzalira 2021: 41).

Dieses Muster sollte in Umgebungen angewendet werden, in denen die Kontrolle über die endgültigen Ziele besteht, wie z. B. Desktop-Anwendungen, Unternehmensanwendungen oder Webplattformen mit einer starken client Side Component, wie von Mezzalira (2020: o. S.) in dem Artikel vorgeschlagen [75] (vgl. Mezzalira 2020: o. S.).

Edge Side Composition

Die Edge Side Composition wird auf der Ebene des Content Delivery Networks (CDN) durchgeführt. Hierbei wird die Benutzeroberfläche auf den Servern des CDNs erstellt, nicht

im Browser oder auf dem Server des Benutzers. Die Edge Side Composition kombiniert die Microfrontends vom Ursprung auf der CDN-Ebene und liefert das endgültige Ergebnis an den Client, also den Browser des Benutzers. Viele CDN-Anbieter ermöglichen die Verwendung von ESI, einer Markup-Sprache, die auf XML basiert. ESI ist keine neue Sprache, und Unternehmen wie Akamai und Oracle haben sie bereits im Jahr 2001 als Standard vorgeschlagen [11] (vgl. Peltonen et al. 2021: 5).

ESI unterstützt die Skalierung einer Webinfrastruktur, indem Inhalte effizient und schnell an Benutzer auf der ganzen Welt geliefert werden können, indem ein CDN-Netzwerk eine große Anzahl von Points of Presence weltweit nutzt [6] (vgl. Mezzalira 2021: 30 f.). Die Verwendung von ESI ist jedoch herausfordernd, da jeder CDN-Anbieter sie anders implementiert. Die Implementierung von ESI kann durch die Anwendung einer Multi-CDN-Strategie oder den einfachen Wechsel des Anbieters zu mehreren Überarbeitungen und der Einführung neuer Logik führen [6] (vgl. Mezzalira 2021: 30 f.).

Die Edge Side Composition ist ausschließlich mit dem Horizontal Split kompatibel, da die Transklusion auf der CDN-Ebene bei einem Vertical Split keinen Sinn ergibt. Wie bereits erwähnt, stellt die Skalierbarkeit bei Webanwendungen eine bedeutende Herausforderung dar. Durch die Edge Side Composition kann bei Projekten mit statischen Inhalten und hohem Datenverkehr die Skalierbarkeitsherausforderung an den CDN-Anbieter delegiert werden, anstatt sich damit in Ihrer eigenen Infrastruktur auseinandersetzen zu müssen. Dieser Ansatz erfordert nicht die vollständige Kontrolle über jedes Detail. Projekte wie beispielsweise ein Onlinekatalog, der keine individuellen Inhalte enthält, könnten für diesen Ansatz geeignet sein [75] (vgl. Mezzalira 2020: o. S.).

Server Side Composition

Microfrontends werden durch die Anwendung von Server Side Composition aggregiert, auf der CDN-Ebene zwischengespeichert und anschließend entweder zur Laufzeit oder zur Kompilierungszeit an den Client übertragen. Die Darstellung wird durch den Origin-Server erstellt, wobei die Microfrontends abgerufen und die finale Seite generiert werden [6] (vgl. Mezzalira 2021: 30).

Der primäre Vorteil der serverseitigen Integration liegt darin, dass die Seite vollständig zusammengesetzt ist, bevor sie den Browser des Endnutzers erreicht [1] (vgl. Geers 2020: 60). Falls die Seite umfangreich zwischengespeichert werden kann, erfolgt ihre Bereitstellung durch das CDN unter Anwendung einer langen Speicherdauer (TTL). Es ist essenziell, die Skalierbarkeit der Webseite zu berücksichtigen, insbesondere wenn eine Anpassung für jeden Benutzer erfolgt, da zahlreiche Anfragen von diversen Nutzern eingehen werden. Nach Festlegung der sogenannten Server Side Composition Strategie sollten die Anwendungsfälle in der Applikation eingehend analysiert werden. Bei Anwendung der Laufzeitkomposition bedarf das Projekt einer klaren Strategie zur Skalierung der Server, um Ausfallzeiten für die Benutzer zu minimieren [11] (vgl. Peltonen et al. 2021: 4).

Wie bereits erwähnt, besteht der wesentliche Unterschied zwischen Client Side und Server Side darin, dass die Verwendung von Client Side hilfreich ist, wenn die Nutzenden die

Kontrolle über die endgültigen Ziele haben möchten. Im Gegensatz dazu ermöglicht die Anwendung der Server Side Composition die größtmögliche Kontrolle über die Ausgabe. Besonders für stark indexierte Websites wie Nachrichtenportale oder E-Commerce-Websites ist dies von Vorteil [6] (vgl. Mezzalira 2021: 42).

Auf der Server Side gibt es sowohl Horizontal Split als auch Vertical Split. Gemäß Mezzalira (2020: o. S.) wäre es sinnvoll, die Server Side in Kombination mit dem Horizontal Split zu verwenden, um eine größere Flexibilität zu erzielen [75] (vgl. Mezzalira 2020: o. S.). Websites, die hohe Leistungsmetriken wie PayPal oder American Express benötigen, könnten diesen Ansatz als zielführend begutachten. Die Server Side ist für beide geeignet [6] (vgl. Mezzalira 2021: 42).

3.3.3 Routing

Nach Festlegung der Compositionsart, sei es Horizontal oder Vertical, steht die nächste Phase der Routenwahl bevor, die als Routin bezeichnet wird. Diese Entscheidung ist eng mit der Art der Composition verbunden. Bereits zu Beginn des Kapitels stellte sich die Frage, wie der Übergang von einer Ansicht zur nächsten Route gestaltet werden soll [7] (vgl. Taibi und Mezzalira 2022: 26).

Im vorherigen Abschnitt wurde erläutert, wie ein Microfrontend mit den verschiedenen Compositionsarten implementiert werden kann. Im Bereich des Routings haben Entwickler ähnliche Optionen wie bei der Composition. Es stehen ihnen drei Möglichkeiten zum Routing zwischen den Microfrontends zur Verfügung: Client, Server und Edge Side [75] (vgl. Mezzalira: 2020).

Die Entwickler müssen entscheiden, wo sie die Seitenanfragen routen möchten, sei es am Origin, am Edge oder auf der Client Side, wie in Abbildung 15 dargestellt [6] (vgl. Mezzalira 2021: 31). Wenn Entwickler das Routing auf der Client Side durchführen, haben sie die Möglichkeit, Logik am Edge hinzuzufügen oder die Routing-Logik ausschließlich auf dem Client oder Server zu implementieren [76] (vgl. Mezzalira: 2019).

In der Regel erfolgt das Routing auf der Client Side, wenn die Entscheidung auf die Client Side fällt [6] (vgl. Mezzalira 2021: 42). Zudem wird in einem Blogbeitrag von Mezzalira [76] (2019: o. S.) empfohlen, die App-Shell zur Abbildung der Routing-Logik zu nutzen. Dadurch lädt die App-Shell lediglich ein Microfrontend anstatt mehrere gleichzeitig, was besonders vorteilhaft ist, wenn komplexe Routen wie Authentifizierung, Geolokalisierung oder andere ausgefeilte Logiken tangiert sind. Beim Routing basierend auf den Seiten-URLs, welches bei der Entscheidung für die Edge Side Composition in der Architektur zum Einsatz kommt, wird die angeforderte Seite vom CDN durch die Zusammenstellung von Microfrontends auf der Edge-Ebene durch Transclusion bereitgestellt [75, 6] (vgl. Mezzalira 2020: o. S., Mezzalira 2021: 32). Die Orchestrierung und Rückgabe statischer Dateien wird dabei durch einen Webserver ermöglicht, um die Skalierbarkeitsprobleme zu vermeiden. Jedoch muss der auf dem CDN ausgeführte Code schnell ausführbar sein, was gelegentlich eine Einschränkung darstellen kann [75] (vgl. Mezzalira 2020: o. S.). Im Falle einer Entscheidung für die Server Side Composition, bei der sämtliche Anwendungslogik auf

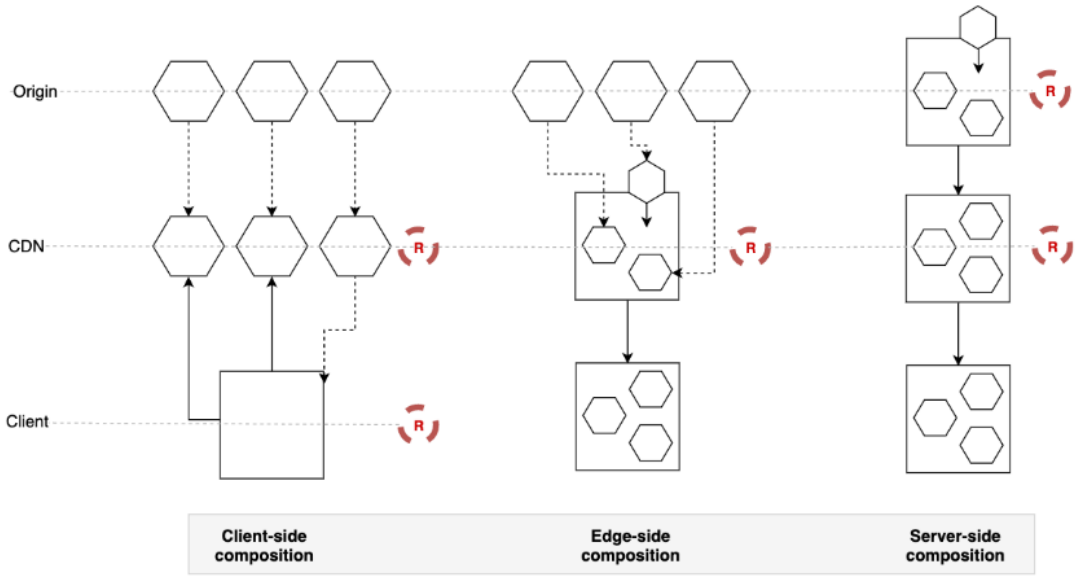

Abbildung 15 Microfrontend Routing
(Quelle: [76] Mezzalira 2019: o. S.)

den Anwendungsservern liegt, ist man angehalten, Anfragen am Ursprung zu routen. Das bedeutet, dass der Anwendungsserver das HTML-Template mit einer bestimmten Route verknüpft, während Routing und Komposition auf der Server Side erfolgen [6] (vgl. Mezzalira 2021: 32). Bei Auswahl der Server Side Composition ist es möglich, die Routing-Logik über den Webserver zu verwalten [6, 76] (vgl. Mezzalira 2021: 42, Mezzalira 2019: o. S.).

3.3.4 Communication

Abschließend ist es erforderlich, zu bestimmen, wie die Microfrontends untereinander kommunizieren sollen. Idealerweise sollten die Microfrontend Anwendungen vollständig unabhängig voneinander sein und keine direkte Kommunikation aufweisen. Jedoch ist dies nicht immer realisierbar, insbesondere wenn mit horizontal split Microfrontends gearbeitet wird, da andere Microfrontends über Benutzerinteraktionen und Aktivitäten informiert werden müssen [6] (vgl. Mezzalira 2021: 33). Bei Bedarf einer Kommunikation zwischen den Microfrontends verschiedener Teams ist sicherzustellen, dass jedes Microfrontend nichts von den anderen auf derselben Seite weiß, um das Prinzip der unabhängigen Bereitstellung nicht zu verletzen [6] (vgl. Mezzalira 2021: 33).

Bei einer horizontal split ist es entscheidend zu definieren, wie die Microfrontends untereinander kommunizieren können. Ein Event-Emitter kann in jedes Microfrontend integriert werden, wodurch jedes Microfrontend autonom arbeiten und völlig unabhängig von den anderen bleiben kann [11] (vgl. Peltonen et al. 2021: 4). Bei einer verticalen split ist es entscheidend zu verstehen, wie Informationen zwischen den Microfrontends geteilt werden können. Beide Arten der Aufteilung erfordern eine Methode zur Datenweitergabe, wobei die Kommunikationslösungen wie Custom events, web storage und query strings verwendet werden können [7, 6] (vgl. Taibi und Mezzalira 2022: 26; Mezzalira 2021: 33 ff.).

Um die Kommunikation zwischen den entkoppelten Komponenten zu ermöglichen, kann

ein Eventbus hinzugefügt werden. Dieser Mechanismus ermöglicht es, dass über einen Bus gesendete Ereignisse von den Microfrontends empfangen und an jedes Microfrontend gemeldet werden [6] (vgl. Mezzalira 2021: 33).

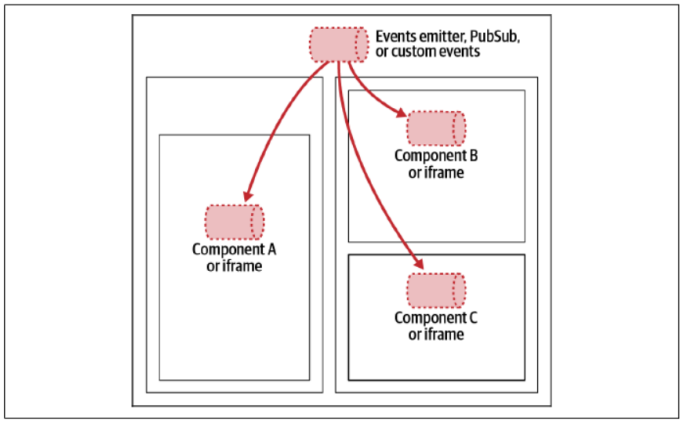

Abbildung 16 Event Emitter und Custom Events Diagramm
(Quelle: [6] Mezzalira 2021: 33)

Die Verwendung von Custom-Events bietet eine zusätzliche Option. Diese Events sind im Wesentlichen normale Events, jedoch mit einem individuellen Inhalt. Dies ermöglicht es, den String, der das Event identifiziert, sowie optional ein individuelles Objekt für das Event zu definieren [6] (vgl. Mezzalira 2021: 34). Es stehen auch andere Möglichkeiten zur Verfügung, um Daten zwischen Sitzungen zu übertragen, wie beispielsweise die Verwendung von Web Storage, Local Storage oder Cookies [6] (vgl. Mezzalira 2021: 34).

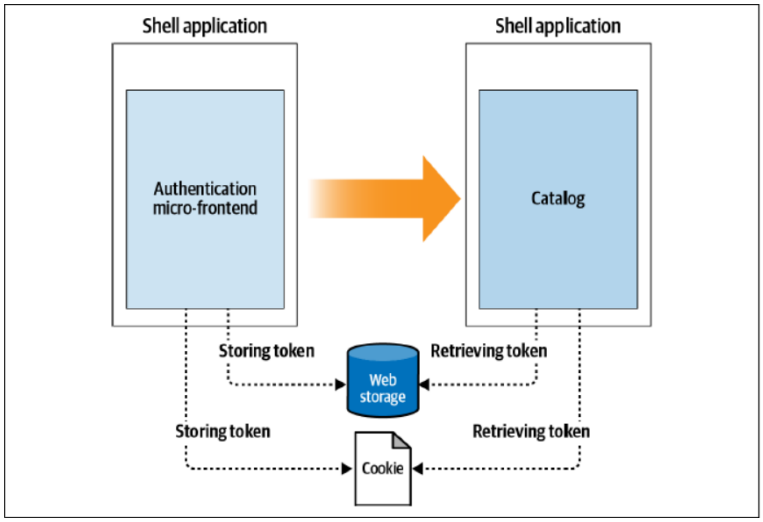

Abbildung 17 Das Teilen von Daten zwischen Microfrontends in verschiedenen Ansichten
(Quelle: [6] Mezzalira 2021: 34)

Eine zusätzliche Möglichkeit besteht darin, Daten mittels Query-Strings in der URL zu übertragen. Jedoch ist zu beachten, dass diese Option Sicherheitsrisiken birgt, insbesondere wenn es um die Übertragung sensibler Daten wie Passwörter, Tokens und Benutzer-IDs

geht [6] (vgl. Mezzalira 2021: 37).

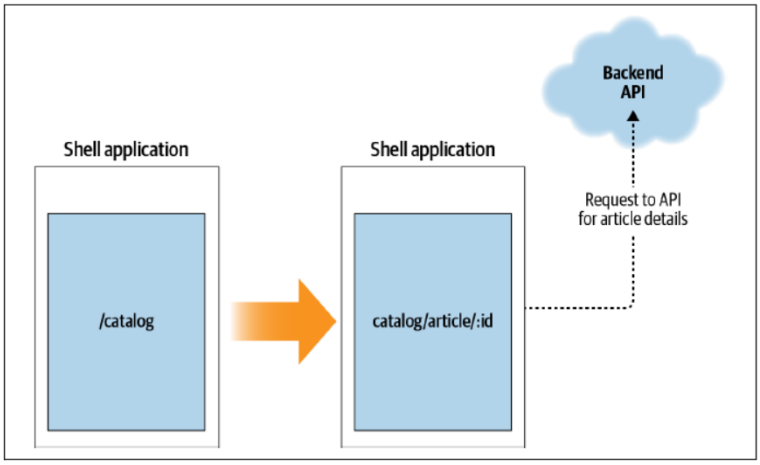

Abbildung 18 Microfrontends Kommunikation mit query strings oder URL (Quelle: [6] Mezzalira 2021: 35)

Definieren	Composing	Routing	kommonication
Horizontal Split	Client-Side Edge-Side Server-Side	Client-Side Edge-Side Server-Side	Event Emitter Custom Event Web Storage Query Strings
Vertical Split	Client-Side Server-Side	Client-Side Server-Side Edge-Side	Web Storage Query Strings

Tabelle 1 Zusammenfassung des Entscheidungsrahmens für Microfrontends

3.4 Microfrontend Frameworks

Ein Framework ist ein vorgefertigter, wiederverwendbarer Satz von Tools, Bibliotheken und Konventionen, der eine strukturierte Methode zur Erstellung und Wartung von Webanwendungen bietet. Diese Frameworks sollen den Entwicklungsprozess vereinfachen, indem sie eine Grundlage bereitstellen, auf der Entwickler aufbauen können, anstatt jedes Mal von Grund auf neu zu beginnen. Sie enthalten häufig gängige Funktionalitäten und Muster, was es den Entwicklern ermöglicht, sich auf die Entwicklung spezifischer Funktionen und auf die Lösung spezifischer Probleme zu konzentrieren, anstatt sich mit wiederholten Projektinitialisierungsaufgaben zu befassen [100] (vgl. FasterCapital: o.J.o.S).

Das Frontend- und Backend-Framework sind die beiden Hauptkomponenten des Website-Entwicklungsframeworks. Programmiersprachen wie JavaScript, HTML und CSS werden für die Erstellung des Frontends verwendet. Verschiedene Frontend-Frameworks wurden entwickelt, um die Arbeit der Entwickler zu erleichtern. Zu den bekanntesten gehören

Svelte.js, Angular.js, Vue.js und React.js.

Entwickler können eine Vielzahl von Programmiersprachen im Backend-Bereich einsetzen. Jede dieser Sprachen hat ihre eigenen Rahmenbedingungen, die Entwicklern bei der Webentwicklung und Implementierung helfen können, je nach den Zielen des Projekts. JavaScript (Node.js), Python (Django, Flask), PHP (Laravel) und Java sind einige dieser Frameworks.

Zusätzlich gibt es viele spezialisierte Frameworks für die Microfrontend Architektur, die jeweils für spezifische Zwecke entwickelt wurden. Einige dieser Frameworks sind für Entwickler auf der Server Side verfügbar, während andere für Entwickler auf der Client Side verfügbar sind. Im Folgenden werden einige der verfügbaren Frameworks kurz untersucht [72] (vgl. Saring 2020: o. S.).

Um Microfrontends zu implementieren, können verschiedene Open-Source-Frameworks wie Tailor.js, OpenComponents, Ara, Podium, Piral, Qiankun, Single-Spa, Bit, System.js, Frint.js und Puzzle.js verwendet werden [1, 6, 7, 72, 76] (vgl. Geers 2020: 73–81; Mezzalira 2021: 102; Taibi und Mezzalira 2022: 28; Saring 2020: o. S.; Mezzalira 2019: o. S.).

3.4.1 Piral

Piral ist ein Open-Source-Framework, das auf React basiert und als Microfrontend-Framework gilt. Es ermöglicht Entwicklern, schnell Webanwendungen zu entwickeln, die die Architekturregeln und Prinzipien von Microfrontends erfüllen [74] (vgl. Piral Official Website o. J.: o. S.).

Das Piral-Framework besteht aus zwei Hauptkomponenten. Die erste Komponente ist eine Piral-Instanz (Application Shell), die das Gesamtdesign der Anwendung umfasst, einschließlich Header, Footer und gemeinsamer Komponenten, die von Pilets verwendet werden können. Die Pilets (Feature-Module), die den Inhalt der Anwendung enthalten, sind im zweiten Abschnitt enthalten [74] (vgl. Piral Official Website o. J.: o. S.).

Das folgende Diagramm zeigt, dass Entwickler dieses Framework mit wenig Aufwand einrichten können. Dies bedeutet, dass sie es schnell mit Node.js, einer Entwicklungsumgebung, einem Terminal und einem Internetbrowser starten können [74] (vgl. Piral Official Website o. J.: o. S.).

„The Piral instance (application shell) and the pilets (feature modules) can be executed and debugged in the emulator on the local development machine.“[74] (Piral Official Website o. J.: o. S.).

Dieses Framework wird primär verwendet, wenn Microfrontends auf der Client Side implementiert werden müssen, da es auf React basiert. Im Artikel von Taibi und Mezzalira (2022: 28) wird jedoch erwähnt, dass dieses Framework auch mit der Server Side verwendet werden kann [7] (vgl. Taibi und Mezzalira 2022: 28).

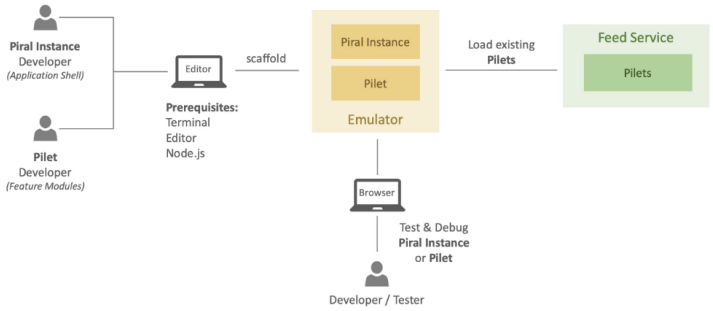

Abbildung 19 Aufbau und Entwicklungsprozess mit Piral
(Quelle: [74] (Piral Official Website o. J.: o. S.)

3.4.2 Luigi

Luigi ist ein Open-Source-Framework, das von SAP entwickelt wurde und speziell für die Entwicklung von Microfrontends konzipiert ist [6, 73] (vgl. Mezzalira 2021: 90; Luigi Official Website o. J.: o. S.). Das Luigi Framework besteht aus zwei Hauptkomponenten. Der erste Abschnitt umfasst das Luigi Core, das die Architektur der Microfrontends beinhaltet. Der Luigi Client bildet den zweiten Abschnitt, der sicher mit Microfrontend-Anwendungen und der Hauptansicht über den Mechanismus der PostMessage-API kommunizieren kann [6, 72, 73] (vgl. Mezzalira 2021: 90; Saring 2020: o. S.; Luigi Official Website o. J.: o. S.).

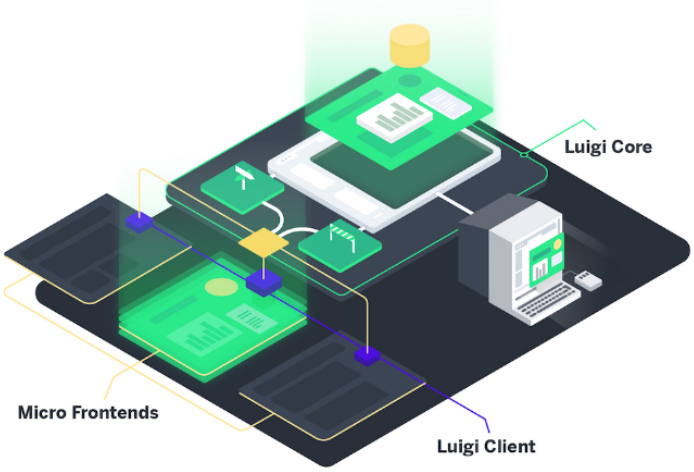

Abbildung 20 Diagramm des Luigi Frameworks
(Quelle: [73] Luigi Official Website o. J.: o. S.)

Das Hauptziel dieses Frameworks ist es, eine Verbindung zwischen einer Webanwendung und den enthaltenen Microfrontend-Anwendungen herzustellen. Wie im folgenden Diagramm dargestellt, ermöglicht dieses Framework über die Luigi.js-API eine sichere Kommunikation zwischen den Microfrontend-Anwendungen und dem Luigi Client [6, 72] (vgl. Mezzalira 2021: 90; Saring 2020: o. S.).

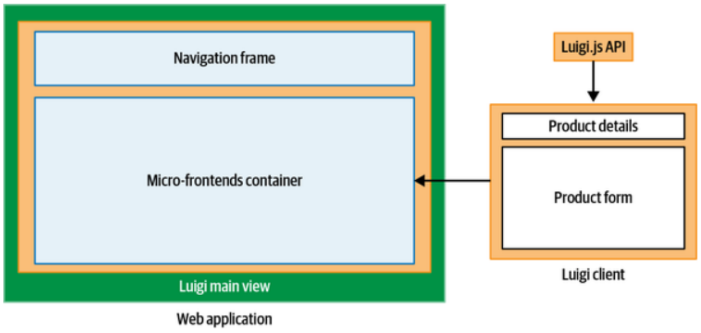

Abbildung 21 Microfrontend mit Luigi
(Quelle: [6] Mezzalira 2021: 91)

In Situationen, in denen die Entscheidung getroffen wurde, eine Microfrontend-Architektur auf der Client Side durch eine horizontale split mittels des IFrames-Ansatzes zu erstellen, erweist sich Luigi als wichtiges Framework. Luigi unterstützt beispielsweise Entwickler bei der Entwicklung von Webanwendungen [6] (vgl. Mezzalira 2021: 90).

Die Verwendung dieses Frameworks ermöglicht es Entwicklern, Funktionen und Fehler schneller bereitzustellen und zu beheben, die Wartungskosten zu senken und eine kleinere und leichter verwaltbare Codebasis zu schaffen. Darüber hinaus erlaubt es Entwicklern, Anwendungen mit verschiedenen Technologien zu erstellen [73] (vgl. Luigi Official Website o. J.: o. S.).

3.5 Anwendungsfälle von Microfrontend

Die erste Forschungsfrage dieser vorliegenden Masterarbeit lautet:

- Wann ist es sinnvoll, Microfrontends in Betracht zu ziehen, und was sind die Vor- und Nachteile?

Laut Geers (2020: 19) sollten Microfrontends nicht als universelle Lösung für alle auftretenden Probleme betrachtet werden [1] (vgl. Geers: 2020: 19). Zusätzlich betont Mezzalira (2021: 20), dass ein Projekt aufgrund falscher Architekturentscheidungen scheitern oder verzögert werden könne [6] (vgl. Mezzalira 2021: 20). Daher ist es entscheidend, ein grundlegendes Verständnis dieser Architektur zu haben und die Vor- und Nachteile zu berücksichtigen.

Basierend auf den Aussagen von Geers (2020: 12–19) ist das Hauptziel von Microfrontends die Reduzierung der Wartezeit zwischen Teams, die Beschleunigung der Entwicklungsgeschwindigkeit und die Verbesserung der Skalierbarkeit von Projekten [1] (vgl. Geers: 2020: 12–19). In dem Folgenden werden Szenarien untersucht, in denen die Verwendung von Microfrontends sinnvoll erscheinen könnte.

1. Mittlere bis große Projekte

Die Two-Pizza-Team-Regel von Amazon-CEO Jeff Bezos empfiehlt eine Teamgröße von

fünf bis zehn Personen. Wenn jedoch mehr als zehn Teammitglieder vorhanden sind, können Probleme wie steigende Kommunikationskosten und die Komplexität von Entscheidungsprozessen auftreten, was eine Aufteilung des Teams ratsam macht. Bei der Arbeit an einem Programm mit wenigen Teammitgliedern wird die Skalierbarkeit wahrscheinlich nicht als großes Problem angesehen [1] (vgl. Geers 2020: 19). Wie von Mezzalira (2021: 14) betont, ist Skalierbarkeit im Frontend-Bereich heutzutage entscheidend, obgleich sie früher nicht so wichtig war [6] (vgl. Mezzalira 2021: 14).

Die Anwendung eines Microfrontendansatzes ist ideal, wenn mit mehreren Teams gearbeitet wird, die an unterschiedlichen Merkmalen arbeiten. Dieser Ansatz eignet sich gut für komplexe Projekte und ermöglicht eine schnellere Bereitstellung von Funktionen, ohne die gesamte Website zu beeinträchtigen [70, 71] (vgl. Shpak 2023: o. S.; Brooks 2023: o. S.).

2. Web Projekt

Geers (2020: 20) und Brooks (2023: o. S.) betonen, dass Microfrontends in einer Vielzahl von Anwendungen verwendet werden können, ohne auf eine spezifische Plattform beschränkt zu sein, sondern vielmehr in verschiedenen Anwendungsbereichen Anwendung finden können. Ihre optimale Nutzung liegt jedoch in der Bereitstellung von Webanwendungen. Obwohl native Anwendungen für Android und iOS eine monolithische Architektur verwenden, ist es möglich, Microfrontends in diese Anwendungen zu integrieren. Es gestaltet sich jedoch schwierig, sie zu komponieren und Funktionalitäten auszutauschen [1, 71] (vgl. Geers 2020: 20; Brooks 2023: o. S.). Zusätzlich unterstreicht Geers (2020: 20), dass es sich jedoch als herausfordernd gestaltet, mehrere End-to-End-Teams gleichzeitig an der Implementierung nativer Benutzeroberflächen arbeiten zu lassen, ohne sich dabei gegenseitig zu beeinträchtigen [1] (vgl. Geers 2020: 20).

Wie bereits erwähnt, können Microfrontends dazu beitragen, die Skalierbarkeit von Projekten zu verbessern. Es ist jedoch wichtig zu bedenken, dass nicht jedes Projekt von der Implementierung von Microfrontends profitieren wird. In kleineren Projekten mit wenigen Entwicklern und geringem Bedarf an komplexer Kommunikation wird der Einsatz von Microfrontends wahrscheinlich nur begrenzten Nutzen bringen [1] (vgl. Geers 2020: 20).

Es ist entscheidend zu beachten, dass die Entscheidung, ob Microfrontends verwendet werden sollen, von den spezifischen Anforderungen des Projekts, der Projektgröße, den verfügbaren Ressourcen und anderen Faktoren abhängt. Bevor ein Projekt beginnt, sollten diese Anforderungen sorgfältig untersucht und analysiert werden, um festzustellen, ob die Implementierung von Microfrontends sinnvoll ist.

3.6 Vorteile und Einsatzszenarien

3.6.1 Technologievielfalt

Die Microfrontend-Architektur bietet eine Vielfalt bei der Auswahl der Technologie. Unterschiedliche Teams haben die volle Freiheit, ihre bevorzugte Technologie zu wählen, basierend auf den Anforderungen, Fähigkeiten und Zielen ihres spezifischen Bereichs. Diese Freiheit ermöglicht es den Teams, ihre bevorzugten Frameworks zu verwenden und entsprechend zu entwickeln [11, 69, 70] (vgl. Peltonen et al. 2021: 12; Ağaç 2023: o. S.; Shpak 2023: o. S.).

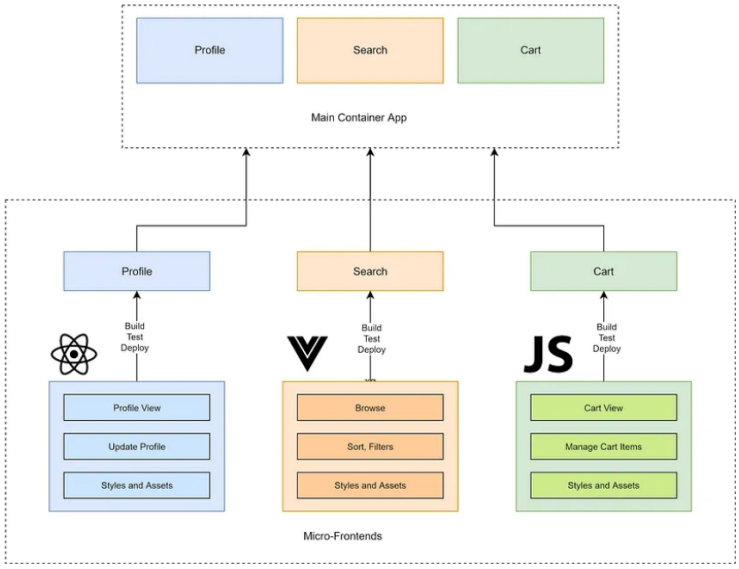

Abbildung 22 Vielfalt bei der Technologieauswahl
(Quelle: [4] Jackson 2019: o. S.)

3.6.2 Entwicklung und Bereitstellung

Die Hauptvorteile von Microfrontends, ähnlich wie bei Microservices, liegen in ihrer unabhängigen Entwicklung und Bereitstellung, die eine Schlüsselrolle spielen [1, 4, 6, 10, 11, 70] (vgl. Geers 2020: 16; Jackson 2019: o. S.; Mezzalira 2021: 19; Petcu et al. 2023: 327; Peltonen et al. 2021: 13; Shpak 2023: o. S.). So können Teams ihre unabhängigen Produkte mithilfe von Microfrontends in ihrem eigenen Tempo bereitstellen [6] (vgl. Mezzalira 2021: 19). Jedes Microfrontend sollte eine „continuous delivery pipeline“ haben, die es von Anfang bis Ende produziert, testet und ausführt [4] (vgl. Jackson 2019: o. S.). Der Vorteil besteht darin, dass keine Teile der Anwendung darauf warten müssen, dass externe Abhängigkeiten vor der Bereitstellung gelöst werden [6] (vgl. Mezzalira 2021: 19), da jedes Microfrontend seine eigenen Markierungen, Stile und Skripte mitbringt und keine gemeinsamen Laufzeitabhängigkeiten haben sollte [1] (vgl. Geers 2020: 16). Andererseits beeinflusst dieser Vorteil aufgrund der Unabhängigkeit jedes Microfrontend von anderen Teilen der Anwendung keine Änderungen an anderen Teilen der Anwendung, und Teams können neue Funktionen ohne Rücksprache mit anderen Teams hinzufügen und implementieren. Daher ermöglicht es eine schnellere Bereitstellung und Entwicklung von Anwendungen und unterstützt die „continuous integration, continuous deployment und continuous delivery“ [11] (Peltonen et

al. 2021: 13).

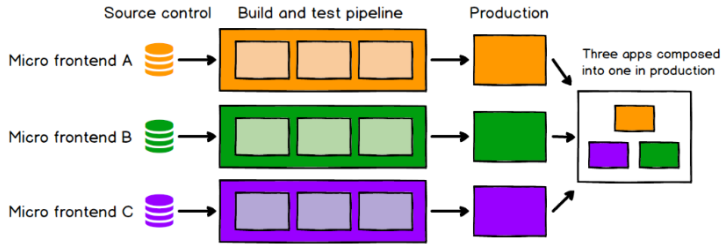

Abbildung 23 Unabhängige Entwicklung und Bereitstellung durch Microfronted (Quelle: [4] Jackson 2019: o. S.)

3.6.3 Isolation von Fehlern

Microfrontends haben auch den Vorteil, dass Fehler sich nicht auf das gesamte Programm auswirken, sondern isoliert auftreten. Die Microfrontend Architektur ermöglicht es, Probleme in einem Teil des Programms mithilfe des App-Shells zu identifizieren und den Benutzer darüber zu informieren. Fehler in einem Microfrontend haben nicht zwangsläufig Auswirkungen auf andere Teile des Programms, und das Beheben eines Fehlers in einem Modul kann keine Auswirkungen auf das gesamte Programm haben [11, 99] (vgl. Peltonen et al. 2021:13; Prajwal et al. 2021:125).

3.6.4 Skalierbarkeit

Durch die Aufteilung der Benutzeroberfläche in kleinere, eigenständige Teile können Microfrontends unabhängig voneinander entwickelt, bereitgestellt und skaliert werden. Dies ermöglicht eine bessere Skalierbarkeit der Anwendung, da bestimmte Teile bei Bedarf individuell skaliert werden können [1, 3, 4, 10, 11] (vgl. Geers 2020: 19; Pavlenko et al. 2020: 54; Jackson 2019: o. S.; Petcu et al. 2023: 323; Peltonen et al. 2021: 4).

3.6.5 Wiederverwendbarkeit

Microfrontend bietet die Wiederverwendung von Code an. Entwickler können Code zwischen verschiedenen Anwendungen und Teams wiederverwenden, da Microfrontends modular und unabhängig voneinander sind.

3.6.6 Autonome Teams

Ein weiterer Vorteil der Verwendung von Microfrontend sind Autonome Teams mit unterschiedlichen Verantwortlichkeiten. Aufgrund der unabhängigen Entwicklung und Bereitstellung von Microfrontends können Teams jedes Element der Anwendung, von der Benutzeroberfläche bis zur Datenbank und zum Backend, eigenständig erstellen, testen, bereitstellen und aktualisieren, wie in der Abbildung visualisier ist. Einer der wichtigsten Vorteile von Microservices und Microfrontends ist die Autonomie, da sie die Zusammenarbeit und die Motivation fördert. Teams müssen für eine verticale Geschäftsfunktionalität

gebildet werden, nicht um technische Fähigkeiten oder eine schnelle und effektive Teamarbeit zu gewährleisten [1, 4, 69, 70] (vgl. Geers 2020: 16; Jackson 2019: o. S.; Ağaç 2023: o. S.; Shpak 2023: o. S.).

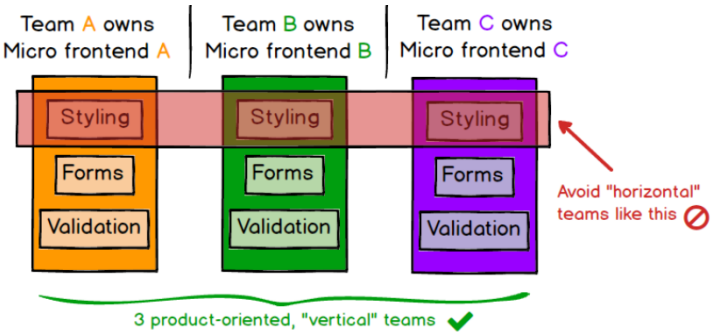

Abbildung 24 Verantwortlichkeit für Anwendungen
(Quelle: [4] Jackson 2019: o. S.)

3.7 Herausforderungen und Risiken

Wie bereits erörtert, eröffnet die Architektur der Microfrontends eine Vielzahl von Vorteilen für Entwickler. Dennoch stellt sich die Frage nach ihrer Eignung als optimale Lösung für Frontend Architekturen. Es ist evident, dass diese Architektur nicht sämtliche Herausforderungen und Probleme wie andere Architekturparadigmen adäquat adressieren kann. Daher ist es von zentraler Bedeutung, vor der Implementierung der Microfrontend Architektur in einem Projekt deren potenzielle Nachteile und Herausforderungen eingehend zu untersuchen und zu berücksichtigen. Im Folgenden werden einige dieser Aspekte näher erläutert.

3.7.1 Komplexität

Die Microfrontend Architektur ermöglicht durch die Unterteilung des Programms in kleinere Module unabhängige Bereitstellungen und Entwicklungen [1, 4, 6, 10, 11, 70] (vgl. Geers 2020: 16; Jackson 2019: o. S.; Mezzalira 2021: 19; Petcu et al. 2023: 327; Peltonen et al. 2021: 13; Shpak 2023: o. S.) Trotz dieser Vorteile haben die Autoren jedoch auf die Herausforderungen hingewiesen, die mit der Segmentierung des Programms in kleinere Module einhergehen und zu einer erhöhten Komplexität des Designs und der Implementierung von Microfrontends führen können [11, 69, 70, 71] (vgl. Peltonen et al. 2021: 14; Ağac 2023: o. S.; Shpak 2023: o. S., Brooks 2023: o. S.). Es ist daher von entscheidender Bedeutung, bei der Verwendung der Microfrontend Architektur sicherzustellen, dass der Code im Programm ordnungsgemäß funktioniert und leicht lesbar ist. Dies erfordert zusätzliche Unit-Tests, was die Entwicklungsschritte komplizierter macht [71] (vgl. Brooks 2023: o. S.).

Brooks (2023: o. S.) betont, dass die gleichzeitige Verwendung mehrerer Microfrontends in einem Programm problematisch sein kann, insbesondere wenn Aktualisierungen oder Änderungen am Inhalt erforderlich sind. Dies liegt daran, dass ein umfassendes Verständnis der grundlegenden Abhängigkeiten zwischen den Modulen erforderlich ist, um sicherzu-

stellen, dass sie ordnungsgemäß im Programm bereitgestellt werden können, was den Bereitstellungsprozess komplexer macht. Es kann erforderlich sein, mehrere Pipelines für die Bereitstellung zu koordinieren, um die Anforderungen eines Moduls zu erfüllen [71] (vgl. Brooks 2023: o. S.).

Die Annahme der Microfrontend Architektur erfordert eine eingehende Analyse, um zu verstehen, wie das Programm in kleinere Module aufgeteilt und diese entwickelt und bereitgestellt werden können. Eine höhere Ebene der Abstimmung und Zusammenarbeit ist erforderlich, um die Systemkohärenz aufrechtzuerhalten [11, 69, 70, 71] (vgl. Peltonen et al. 2021: 14; Ağaç 2023: o. S.; Shpak 2023: o. S.; Brooks 2023: o. S.). Peltonen et al. (2021: 14) weisen darauf hin, dass die Akzeptanz dieser Architektur neben der technischen auch zu einer Zunahme der organisatorischen Komplexität führen kann. Dies legt nahe, dass eine genaue Bewertung und Festlegung geeigneter Bedingungen erforderlich ist, da diese Architektur nicht für jede Anwendungssoftware geeignet ist [11] (Peltonen et al. 2021: 14).

3.7.2 Payload Size

Die Payload-Größe in der Microfrontend-Architektur stellt einen weiteren Nachteil dar. Durch die Wiederholung von Abhängigkeiten in Microfrontends wird die Größe der Datenübertragung erhöht [4, 70, 71] (vgl. Jackson 2019: o. S.; Shpak 2023: o. S.; Brooks 2023: o. S.). Wie bereits im Abschnitt über die Vorteile genannt, ermöglicht diese Architektur Teams die vollständige Freiheit bei der Auswahl von Technologien und Frameworks, sodass sie die besten Optionen gemäß ihren Zielen auswählen können. Diese Freiheit bei der Auswahl von Frameworks kann jedoch zu einer Herausforderung führen, die nicht ignoriert werden sollte [1] (vgl. Geers 2020: 18). Wenn ein Projekt mehrere Frameworks verwendet, müssen Browser viele Daten abrufen, was zu einer erhöhten Payload-Größe der Webanwendung führen kann [11, 70] (vgl. Peltonen et al. 2021: 14; Shpak 2023: o. S.).

In der Microfrontend Architektur wird dieser Aspekt als grundlegende Herausforderung betrachtet [4, 11, 70, 71] (vgl. Jackson 2019: o. S.; Peltonen et al. 2021: 14; Shpak 2023: o. S.; Brooks 2023: o. S.). Obwohl Shpak (2023: o. S.) und Jackson (2019: o. S.) in ihren Blogs behaupten, dass Webseiten in der Microfrontend-Architektur schneller heruntergeladen werden als in einer monolithischen Architektur, basiert diese Interpretation darauf, dass in der Microfrontend-Architektur nur die Ressourcen und Abhängigkeiten der betreffenden Seite heruntergeladen werden, was den Downloadvorgang beschleunigt [4, 70] (vgl. Jackson 2019: o. S., Shpak 2023: o. S.).

3.7.3 Communication

Ferner sollten Microfrontends normalerweise unabhängig voneinander sein und keine direkte Kommunikation untereinander haben [6] (vgl. Mezzalira 2021: 33). Dennoch können Probleme im Zusammenhang mit den Systemfunktionen auftreten, wenn eine gelegentliche Kommunikation zwischen den Microfrontends erforderlich ist [99] (vgl. Prajwal et al. 2021: 126).

3.7.4 Testen und Debuggen

Das Testen und Debuggen von Microfrontends können komplizierter sein als bei einer monolithischen Anwendung. Es erfordert möglicherweise zusätzliche Testfälle und Werkzeuge, um sicherzustellen, dass die einzelnen Microfrontends richtig funktionieren und gut zusammenarbeiten.

4 Ansätze zur Implementierung von Microfrontends

Das vorliegende Kapitel bietet eine eingehende Auseinandersetzung mit dem Konzept der Microfrontends. Das Hauptanliegen dieses Kapitels besteht darin, auf die formulierten Forschungsfragen adäquate Antworten zu liefern. Die erste Forschungsfrage befasst sich mit der Abgrenzung der vielfältigen Implementierungsansätze von Microfrontends sowie der Bestimmung derjenigen Ansätze, die als besonders wirksam erachtet werden. Die zweite Forschungsfrage zielt darauf ab, die potenziellen Vorteile zu erörtern, die sich aus der Anwendung der identifizierten Ansätze zur Implementierung von Microfrontends ergeben und wie diese die Qualität von Webanwendungen im Frontend verbessern können. Angesichts der Vielfalt verfügbarer Ansätze ist es erforderlich, jeden dieser Ansätze eingehend zu analysieren.

4.1 Client Side

4.1.1 iFrames

In diesem Unterkapitel wird ein Ansatz zur Implementierung von Microfrontends auf der Client Side durch horizontal split diskutiert. Es werden die Anwendungsmöglichkeiten sowie Vor- und Nachteile erörtert. Ein HTML-Element, das es ermöglicht, ein weiteres HTML-Dokument innerhalb des aktuellen Dokuments einzubetten, wird als IFrame bezeichnet [77] (vgl. w3schools o. J.: o. S.). Valdes (2023: o. S.) zufolge erlaubt ein IFrame das Einbetten von Inhalten von einer anderen Website auf eine eigene Seite [78] (vgl. Valde 2023: o. S.).

```
...
<iframe src="http://localhost:3002/recommendations/porsche"></iframe>
...
```

Abbildung 25 Iframes Defeine
(Quelle: [1] Geers 2020: 34)

Obwohl Iframes keine neue Technologie sind, gehören sie zu den ältesten Techniken, die für die Implementierung von Microfrontends in Betracht gezogen werden können. Abhängig von der Art des Projekts und den spezifischen Anforderungen kann die Verwendung von Iframes für die Implementierung von Microfrontends in Betracht gezogen werden [4, 6, 12, 15] (vgl. Jackson 2019: o. S.; Mezzalira 2021: 88; Stefanovska und Trajkovik 2022: 98; Wang et al. 2020: 1570). Iframes können insbesondere bei der Verwendung von Microfrontends mit horizontalem split eingesetzt werden [7] (vgl. Taibi und Mezzalira 2022: 27).

Gemäß Mezzalira (2021: 87), Jackson (2019: o. S.) und Wang et al. (2020: 1570) könnte die Verwendung von Iframes bei der Implementierung eines Microfrontends aufgrund ihrer Benutzerfreundlichkeit bei der Integration und Verbindung von Anwendungen auf Webseiten die erste Lösung sein, die Entwickler in Betracht ziehen [4, 6, 15] (vgl. Jackson 2019: o. S.; Mezzalira 2021: 87; Wang et al. 2020: 1570). Dies liegt daran, dass sie eine starke Sandbox-Isolierung zwischen den Microfrontends schaffen können, um Konflikte bei Bibliotheksversionen, Speicherlecks oder anderen Problemen zu vermeiden, die bei anderen Ansätzen auftreten können [1, 6, 7, 9] (vgl. Geers 2020: 34; Mezzalira 2021: 87; Taibi

und Mezzalira 2022: 27; Bühler et al. 2022: 7). Mezzalira (2021: 87) ist der Meinung, dass diese Funktion Iframes von anderen Implementierungsmethoden unterscheidet, da keine der anderen Methoden diese Eigenschaft bieten kann [6] (vgl. Mezzalira 2021: 87). Geers (2020: 34) betont auch, dass alle Änderungen in einem Iframe innerhalb desselben Iframes verbleiben. Es ist jedoch möglich, die lockeren Kopplungs- und Robustheitseigenschaften aufrechtzuerhalten, die die Verbindung durch Link-Integration bietet [1] (vgl. Geers 2020: 34).

Im weiteren Verlauf wird auch darauf eingegangen, wie Iframes verwendet werden können, um Microfrontends zu erstellen. Ein Beispiel, das in einschlägiger Literatur angeführt wird, wird kurz erläutert. Angenommen, es besteht eine Website, die einem Traktorgeschäft gehört, wie in der Abbildung 25 zu sehen ist. Auf der Startseite ist lediglich ein Produkt zu sehen, während auf andere Produkte über einen Link zugegriffen werden kann, der vom Entwicklerteam oben rechts platziert wurde. Obwohl die Verwendung eines Links eine der einfachsten Möglichkeiten zur Integration mehrerer Seiten darstellt, ist es wichtig zu beachten, dass eine Verbesserung des Benutzererlebnisses erforderlich ist [1] (vgl. Geers 2020: 34).

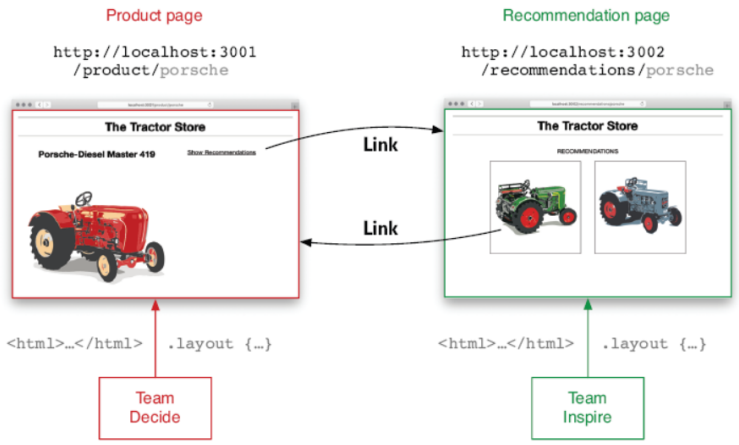

Abbildung 26 Verbindung Produkt und Recommendation Page mit Links
(Quelle: [1] Geers 2020: 29)

Auf den ersten Blick scheint alles wie gewünscht zu sein und die anderen Produkte können über einen Link auf der neuen Seite besucht werden. Es besteht jedoch die Möglichkeit, dass der Link für alle, die das Geschäft besuchen, unklar genug ist. Dies könnte ein großes Problem darstellen. Untersuchungen haben gezeigt, dass die Hälfte der Besucher der Unternehmenswebsite den Link nicht bemerkt und die Website verlässt, da sie glaubten, dass es nur dieses eine Produkt gibt.

Iframes ermöglichen die Integration mehrerer Microfrontends auf einer Seite, wie bereits in den vorherigen Abschnitten erklärt wurde. Die schnellste Methode zur Integration der Empfehlungsseite auf der Produktseite besteht darin, zwei Seiten durch die Verwendung

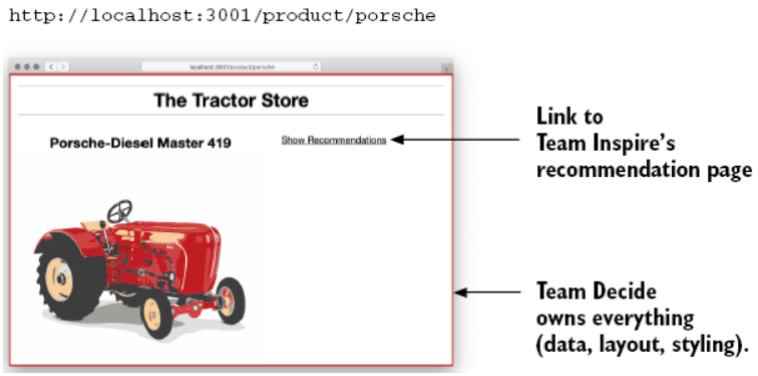

Abbildung 27 Produktseite
(Quelle: [1] Geers 2020: 31)

von Iframes zu kombinieren. Wie die Abbildung unten gezeigt, kann die Empfehlungsseite auf der rechten Seite der Hauptseite des Unternehmens integriert werden, indem Iframes verwendet werden. Jede dieser Seiten hat ein eigenes HTML-Dokument mit einem besonderen Stil, das von der anderen nicht beeinträchtigt wird. Auf diese Weise kann die Benutzererfahrung verbessert und das Interesse der Kunden an dieser Empfehlungsseite auf der Hauptseite geweckt werden.

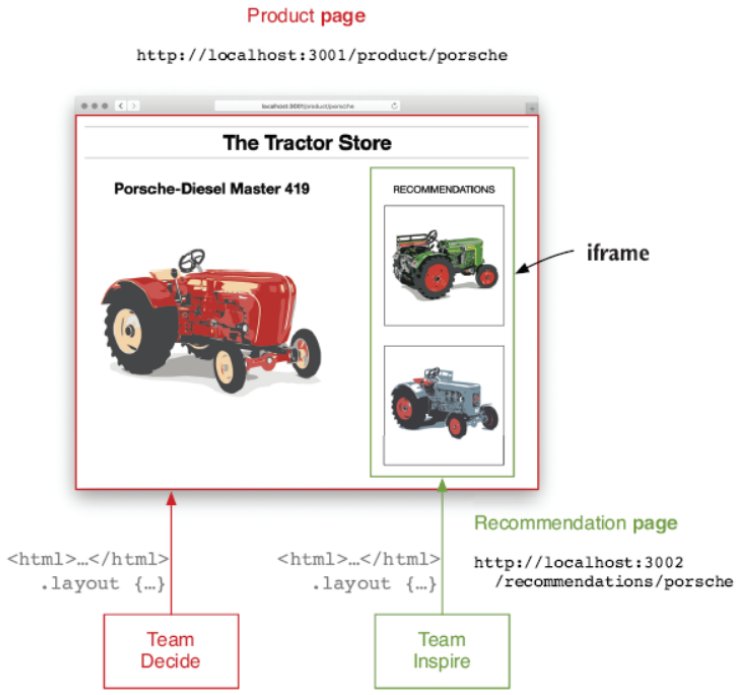

Abbildung 28 Integration von Recommend auf der Produktseite mit IFrames
(Quelle: [1] Geers 2020: 33)

Verfügbare Framework

Wie von Mezzalira (2021: 90) festgestellt, sind nur wenige Frameworks für die Implementierung von Microfrontends mit IFrames verfügbar. Entwickler könnten daher entweder eine interne Strategie entwickeln oder das Luigi Framework [73] verwenden [6] (vgl. Mezzalira

2021: 90).

Anwendungsfälle von IFrames

Bevor im Folgenden die Vor- und Nachteile von Iframes in einem Microfrontend-Projekt erörtert werden, wird in diesem Abschnitt deren Verwendung in verschiedenen Szenarien untersucht, wobei die spezifischen Anforderungen und Ziele des Projekts berücksichtigt werden. Mezzalira (2021:91) betont, dass Iframes nicht für jedes Projekt geeignet sind und unter bestimmten Bedingungen nützlicher sein können als andere Ansätze [6] (vgl. Mezzalira 2021: 91). Ebenso weist Geers (2020: 36) darauf hin, dass die Anwendung von Iframes in einem Projekt stark von den spezifischen Anforderungen abhängt [1] (vgl. Geers 2020: 36).

Aufgrund dieser Argumente werden im Folgenden einige Szenarien untersucht, in denen die Verwendung von Iframes entweder geeignet oder ungeeignet sein könnte. Die Entscheidung für die Nutzung von Iframes sollte sorgfältig abgewogen und an die spezifischen Anforderungen des Projekts angepasst werden.

Für Projekte, die eine Kontrolle über die Benutzerumgebung erfordern, wie Desktop-Anwendungen, B2B-Anwendungen (Business-to-Business) oder Intranet-Anwendungen, sind Iframes eine geeignete Option [1, 6, 7] (vgl. Geers 2020: 36; Mezzalira 2021: 91; Taibi und Mezzalira 2022: 27). Mezzalira (2021: 91 f.). Dies wird anhand eines Beispiels den Einsatz von Iframes in Intranet-Anwendungen veranschaulicht, wo sie als Sicherheitsgrenzen dienen können und eine stärkere Unabhängigkeit zwischen Teams fördern [6] (vgl. Mezzalira 2021: 91). Ein weiteres Szenario ist die Integration von Legacy-Systemen, bei denen Iframes eine schnelle Methode zur schrittweisen Modernisierung des vorhandenen Inhalts ermöglichen, ohne die gesamte Anwendung neu entwickeln zu müssen [1] (vgl. Geers 2020: 34).

In Projekten, in denen umfangreiche Kommunikation zwischen Microfrontends erforderlich ist, wird die Verwendung von Iframes hingegen nicht empfohlen, da der Informationsaustausch zwischen Iframes komplexer und weniger effizient sein kann [6] (vgl. Mezzalira 2021: 91). Ebenso sollte die Verwendung von Iframes vermieden werden, wenn eine schnelle Ladezeit, Barrierefreiheit und gute Indexierung für Suchmaschinen (SEO) erforderlich sind, da dies zu langsameren Ladezeiten führen kann [1] (vgl. Geers 2020: 35 f.).

Vor- und Nachteile

Wie bereits erwähnt, scheinen Iframes ein einfacher und effektiver Ansatz zur Umsetzung von Microfrontends zu sein. Die Verwendung von Iframes bietet eine Reihe von Vorteilen für die Entwicklung von Microfrontends. Erstens schaffen Iframes eine starke Isolation, die es ermöglicht, Komponenten, Programme und die Laufzeitumgebung voneinander zu trennen. Dadurch werden Stile oder Skripte nicht mit anderen Teilen geteilt [9, 15] (vgl. Bühler et al. 2022: 7; Wang et al. 2020: 1570). Dies trägt zur sauberen Strukturierung und Organisation des Codes bei. Ein weiterer Vorteil ist die Browserkompatibilität, da Iframes in allen Browsern funktionieren [1] (vgl. Geers 2020: 35).

Ein wichtiger Aspekt ist auch die Bereitstellung, bei der die Implementierung von Micro-

frontends mit der IFrame-Methode keine besonderen Herausforderungen darstellt [6] (vgl. Mezzalira 2021: 92). Diese Methode ermöglicht eine einfache Integration und Bereitstellung von Microfrontends auf einer Seite. Darüber hinaus gibt es keine Skalierbarkeitsprobleme, da der von Iframes erzeugte Inhalt auf CDN-Ebene zwischengespeichert wird, was es dem CDN ermöglicht, Skalierbarkeitsprobleme zu überwinden [6] (vgl. Mezzalira 2021: 93).

Entwickler, die Microfrontends mit Iframes entwickeln, profitieren von einer ähnlichen Erfahrung wie bei der Entwicklung von Single Page Applications (SPA) [6] (vgl. Mezzalira 2021: 90 ff.). Die Ergebnisse sind statische Seiten, die sich ähnlich wie SPA verhalten. Dies erleichtert die Entwicklung und gewährleistet eine konsistente Benutzererfahrung. Allerdings stehen Entwickler vor Herausforderungen bei End-to-End-Tests, die die Verwendung dieses Ansatzes für sie erschweren können. Dennoch bietet die Verwendung von Iframes eine effektive Möglichkeit zur Entwicklung von Microfrontends und trägt zur Modularität und Skalierbarkeit von Webanwendungen bei.

Ein großer Nachteil von Iframes liegt in ihrer starken Beeinträchtigung der Performance, insbesondere wenn mehrere Iframes auf einer Seite verwendet werden. Dies kann zu unerwünschten Erhöhungen des Hauptspeicherverbrauchs und der CPU führen, was die Gesamtperformance der Anwendung negativ beeinflusst [1, 6] (vgl. Geers 2020: 35; Mezzalira 2021: 88). Daher ist es vor der Verwendung von Iframes in einem Projekt wichtig, die Auswirkungen auf die Leistung sorgfältig zu prüfen.

Ein weiterer Nachteil ist das schlechte Responsive Design, das mit der Verwendung von Iframes einhergehen kann. Da Iframes ältere Technologien sind, können sie in einigen Projekten nicht die bestmögliche Erfahrung für Benutzer und Entwickler bieten, insbesondere im Hinblick auf das Responsive Design der Website. Die Verwendung von Iframes kann die Responsivität der Website beeinträchtigen und somit die Benutzererfahrung negativ beeinflussen [78] (vgl. Valdes 2023: o. S.). Die Flexibilität von Iframes ist im Vergleich zu anderen Methoden eingeschränkt, was sich auf ihre Integration von Microfrontends in großen Projekten auswirkt. Obwohl Iframes eine einfache Isolierung ermöglichen, kann dies ihre Flexibilität beeinträchtigen und die Integration von Microfrontends untereinander erschweren [4, 78] (vgl. Jackson 2019: o. S.; Valdes 2023: o. S.).

Obwohl Iframes aufgrund ihrer Einfachheit bevorzugt werden, gibt es bestimmte Schwierigkeiten, die Kommunikation zwischen Iframes zu ermöglichen und die Größen der Iframes zu orchestrieren, insbesondere wenn die Seite verkleinert wird, ohne das Layout zu ändern. Dies erhöht die Komplexität des Ansatzes und stellt eine zusätzliche Herausforderung dar [6] (vgl. Mezzalira 2021: 92). Die Testbarkeit von Iframes ist ebenfalls eine Schwäche dieses Ansatzes. Aufgrund der DOM-Baumstruktur von Iframes innerhalb einer Ansicht kann die Testbarkeit komplex und anspruchsvoll sein, insbesondere bei End-to-End-Tests [6] (vgl. Mezzalira 2021: 92). Dies erschwert die Durchführung von umfassenden Tests und kann zu Testproblemen führen, die die Qualität der Anwendung beeinträchtigen.

4.1.2 Module Federation

Mezzalira (2021: 81) berichtet in seiner Publikation „Building Micro-Frontends“, dass die Microfrontend Architektur mit der Veröffentlichung von Webpack 5 eine bedeutungsvolle neue Technologieinnovation mit der Bezeichnung „Module Federation“ erhalten hat [6] (vgl. Mezzalira 2021: 81). Gemäß Jaeyow (2022: o. S.) wurde Module Federation im Jahr 2020 unter der Leitung von Zack Jackson, einem Webpack Core-Maintainer, erfunden [79] (vgl. Jaeyow 2022: o. S.). Laut Jackson (2020: o. S.) handelt es sich hierbei nicht um ein Framework, sondern um eine JavaScript-Architektur und ein Plugin, die zu Webpack hinzugefügt wurden [80] (vgl. Jackson 2020: o. S.).

„Module federation allows a JavaScript application to dynamically load code from another application.“[80] (Jackson 2020: o. S.)

Die Hauptmotivation für die Entwicklung von Module Federation durch Jackson besteht darin, die gemeinsame Nutzung von Code unkomplizierter und unabhängiger zu gestalten. Laut El Housieny (2021: o. S.) bietet die Module Federation eine neue Methode, die den Austausch von Komponenten und Informationen zwischen verschiedenen Frontend-Anwendungen erleichtert [81] (vgl. El Housieny 2021: o. S.).

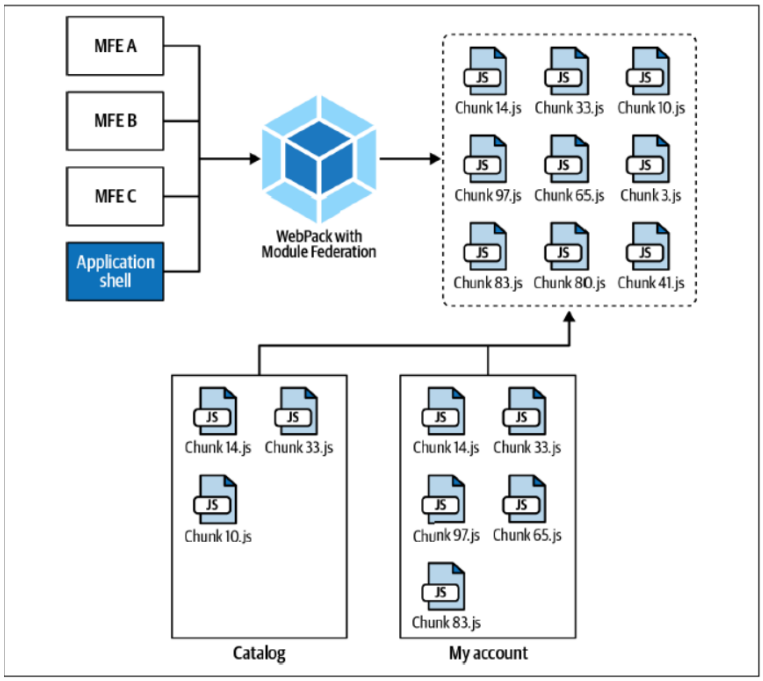

Abbildung 29 Module Federation: Asynchrone Integration von Micro-Frontends für nahtlose Benutzererfahrung
(Quelle: [6] Mezzalira 2021: 82)

Die offizielle Webseite von Webpack unterscheidet zwischen lokalen und Remote-Modulen innerhalb des Module Federation systems. Diese Unterscheidung bezieht sich auf den Speicherort der Module und wie sie in eine Anwendung integriert werden. Lokale Module sind solche, die sich im gleichen Build- oder Entwicklungsprozess wie die Hauptanwendung befinden und während des Buildvorgangs in die Hauptanwendung eingebunden werden. Im Gegensatz dazu sind Remote-Module in einem separaten Build-Prozess erstellte Module,

die unabhängig von der Hauptanwendung entwickelt und gebündelt werden. Sie können in einem anderen Repository oder einer anderen Codebasis existieren und werden zur Laufzeit aus einem Remote-Container geladen [82] (vgl. Webpack Module Federation o. J.: o. S.).

Module Federation ermöglicht das Laden von JavaScript-Codeabschnitten entweder synchronously oder asynchronously zur Laufzeit. Durch die Integration von Remote Modulen ermöglicht Module Federation die unabhängige Entwicklung verschiedener Teile einer Anwendung. Die Option, Module isoliert zu entwickeln und erst zur Laufzeit in die Anwendung zu integrieren, ermöglicht es Entwicklern und Teams, sich auf spezifische Funktionen oder Teile der Anwendung zu konzentrieren, ohne sich um die gesamte Anwendungsstruktur kümmern zu müssen. Dies bedeutet, dass die Anwendung nicht darauf warten muss, dass alle Module sofort verfügbar sind, sondern verschiedene JavaScript-Abschnitte können im Hintergrund zur Laufzeit verzögert geladen werden [6] (vgl. Mezzalira: 2021, S. 81).

Anwendungsfälle von Module Federation

Nachdem der Ansatz der Module Federation eingeführt wurde, werden im Folgenden potenzielle Anwendungsfelder für dessen Nutzung betrachtet. Dieser Ansatz ermöglicht nicht nur die Ausführung von Programmen auf der Client Side, sondern auch das Rendern auf der Server Side [6] (vgl. Mezzalira 2021: 83).

In Betracht gezogen wird eine große Organisation, die den Webpack-Standard einhält und lediglich einen Teil ihres Programms veröffentlichen möchte, anstatt das gesamte Programm. Dieser Ansatz erweist sich insbesondere für Projekte, die auf Webpack basieren, als geeignet. Zudem könnte er in Projekten mit hoher Komplexität aufgrund der Verwendung unterschiedlicher Methoden wie Web Component oder Client-side-Transclusions eingesetzt werden [7] (vgl. Taibi und Mezzalira 2022: 28).

Ein bedeutender Vorteil dieses Ansatzes liegt in der Möglichkeit des Code-Sharings. Dies ermöglicht Entwicklern, Module für mehrere Anwendungen zu teilen. Die Zusammenarbeit erleichtert nicht nur eine nahtlose Integration, sondern ermöglicht auch eine optimale Datenübertragung für jedes Modul. Zusammengefasst erleichtert dieser Ansatz die nahtlose Integration mehrerer Anwendungen mit gemeinsamer Funktionalität [83] (vgl. Module Federation o. J.: o. S.). Als Beispiel könnten eine E-Commerce-Website und eine Blog-Plattform dienen, die beide eine Benutzerauthentifizierung erfordern. Ein gemeinsames Authentifizierungsmodul könnte für beide Anwendungen erstellt und verwendet werden, was nicht nur eine effizientere Entwicklung ermöglicht, sondern auch die nahtlose Integration von Funktionen zwischen verschiedenen Anwendungen erleichtert [83] (vgl. Module Federation o. J.: o. S.).

Ein weiterer Aspekt dieses Ansatzes ist die sogenannte Multi-Application Integration, wie im vorherigen Beispiel dargelegt. Zudem kann er für Third-Party Integrations genutzt werden [89] (vgl. Module Federation o. J.: o. S.). Dies ermöglicht Entwicklern die Integration verschiedener Module aus unterschiedlichen Quellen in ihre Anwendung. Ein weiteres Szenario wäre beispielsweise eine Plattform für den Kauf und Verkauf von Gold, die eine neue Funktion wie den Online-Goldpreis hinzufügen möchte. Diese Funktionalität wird

von einem Third-Party bereitgestellt. Anstatt diese Funktionalität von Grund auf neu zu entwickeln, kann von der Modul Federation profitiert werden, um die Integration von Drittanbieter-Funktionalitäten zu erleichtern.

Vor- und Nachteile

Die Module Federation bietet für die Entwicklung von Webanwendungen zahlreiche Vorteile, gleichwohl es ebenso einige mögliche Nachteile gibt. Diese werden im Folgenden dargelegt.

Die Möglichkeit, Codes zu teilen, ist ein großer Vorteil von Module Federation, was erheblich zur Reduzierung der Code-Wiederholung zwischen Programmen beiträgt. Diese Funktion ermöglicht es, Module zwischen Anwendungen zu teilen, was nicht nur die Fehlererkennung schneller macht, sondern auch das Size und die Komplexität des Programms reduziert [6, 83] (vgl. Mezzalira 2021: 84; Module Federation o. J.: o. S.). Die Module Federation können in bestimmten Situationen die Leistung der Anwendung verbessern, indem sie die Menge des Codes reduzieren, der von jeder Anwendung geladen und ausgeführt werden muss. Die Ladezeiten der Anwendung werden verkürzt, indem der Code verringert wird [83] (vgl. Module Federation o. J.: o. S.). Des Weiteren wird die Skalierbarkeit der Anwendung verbessert, indem nur die Module geladen werden, die zur Laufzeit erforderlich sind. Diese Methode kann helfen, die Größe und Komplexität einzelner Anwendungen zu verringern, was sie wartbar und skalierbar macht [83] (vgl. Module Federation o. J.: o. S.).

Es ist unbestritten, dass die Verwendung des Module Federations die Entwicklung von Microfrontends erheblich erleichtert, insbesondere für Entwickler, die mit ihren Tools vertraut sind. Dieser Ansatz ermöglicht es Entwicklern, den Technologiestack frei auszuwählen, ohne an ein bestimmtes Framework gebunden zu sein. Andererseits ermöglicht die Module Federation mit seinen Funktionen den Entwicklern, Abhängigkeitsprobleme schnell zu lösen. Darüber hinaus haben Nutzende die Möglichkeit, ihr Wissen und ihre Fachkenntnisse durch das Teilen von Codes miteinander zu teilen und gleichzeitig oder asynchron gemeinsam genutzte Codes in Form von Bibliotheken zu laden [6, 78, 83, 84] (vgl. Mezzalira 2021: 85 ff.; Valdes 2023: o. S.; Module Federation o. J.: o. S.; Lakshman 2022: o. S.).

Die Module Federation bietet die Möglichkeit, das Programm in kleinere und einfache zu verwaltende Module aufzuteilen. Dies ermöglicht es den Entwicklern, bei Bedarf Module einfach zu ändern oder auszutauschen, ohne das gesamte Programm neu zu schreiben. Diese Eigenschaft macht diesen Ansatz flexibler [83] (vgl. Module Federation o. J.: o. S.). Die Testbarkeit dieses Ansatzes wird zweifellos als einer ihrer wichtigsten Vorteile betrachtet. Nicht nur Unit- und End-to-End-Tests können durchgeführt werden, sondern auch die Möglichkeit zur Durchführung von integrierten Tests wird durch die Verwendung von Jest eröffnet [6] (vgl. Mezzalira 2021: 86). Wie bereits genannt, wird das Programm durch die Verwendung der Module Federation in kleinere Module aufgeteilt, die statische Dateien und eine hohe Speicherkapazität haben. Daher können sie an verschiedenen Orten wie Cloud-Services oder automatisierten Pipeline-Systemen einfach bereitgestellt werden

[6] (vgl. Mezzalira 2021: 86).

Trotz der Vorteile, die sich aus dem Teilen von Code ergeben, kann die Verwendung der Module Federation auch zu einer erhöhten Komplexität in der Programmierung führen. Insbesondere in Teams ohne klare Struktur oder Disziplin kann das Teilen von Code und Modulen zu einer erhöhten Komplexität führen, die sich negativ auf den Implementierungs- und Wartungsprozess auswirken kann. Daher erfordert dies eine sorgfältige Untersuchung und Verwaltung [6, 83] (vgl. Mezzalira 2021: 83, Module Federation o. J.: o. S.). Die Leistung der Anwendung kann unter verschiedenen Bedingungen variieren und potenzielle Schwächen aufweisen, ähnlich wie beim Teilen von Code in der Module Federation. Insbesondere beim Laden von Modulen, die remote sind, kann die Ladezeit zunehmen, besonders wenn die Module groß und komplex sind [83] (vgl. Module Federation o. J.: o. S.). Es ist von entscheidender Bedeutung sicherzustellen, dass keine unsicheren oder schädlichen Module in die Anwendung geladen werden, da die gemeinsame Nutzung von Code zwischen Anwendungen zu Sicherheitsrisiken führen kann. Eine sorgfältige Handhabung der Sicherheitsaspekte ist daher unerlässlich [83] (vgl. Module Federation o. J.: o. S.).

4.1.3 Web Components

Im weiteren Verlauf dieses Kapitels wird ein weiterer Ansatz zur Implementierung von Microfrontends erläutert, der auf der Client side mithilfe von Web Component umgesetzt wird. Web Component sind eine Reihe von Programmierschnittstellen (APIs) auf Webplattformen, mit denen individuelle, wiederverwendbare HTML-Tags erstellt werden können, die in Webseiten und Webanwendungen eingesetzt werden können [85] (vgl. Webcomponents o. J.: o. S.).

Web Component sind nicht die einzige Lösung für die Implementierung von Microfrontends, allerdings bieten sie Entwicklern spezielle Fähigkeiten, um die Webstandards bei der Umsetzung der Details jedes Microfrontends zu verbergen. Es ist jedoch zu beachten, dass praktisch alle gängigen Frameworks von heute auch die Fähigkeit zur Erzeugung von Web Component besitzen. Web Component ermöglichen es Entwicklern, eine gemeinsame Bibliothek zu erstellen, auch wenn verschiedene Frameworks in Microfrontend-Projekten verwendet werden müssen. Laut Mezzalira (2021: 94 f.) sind Web Component die beste Lösung für die Entwicklung und Implementierung von Microfrontends. Web Component bestehen aus drei Hauptelementen: Custom Elements, Shadow DOM und HTML-Templates. Durch die Verwendung von Custom Elements, Shadow DOM und Web Component können Microfrontends Architekturen effizienter umgesetzt werden [6] (vgl. Mezzalira 2021: 94 f.).

Wie bereits erwähnt, spielt die Flexibilität von Websites heutzutage eine wichtige Rolle im Entwicklungsprozess von Anwendungen. Aus diesem Grund stellen Taibi und Mezzalira (2022: 28) in ihrer Arbeit fest, dass Stencil, eine Bibliothek zur Erstellung von wiederverwendbaren, skalierbaren Komponentenbibliotheken, ein zusätzliches Werkzeug sei, das verwendet werden kann, um die Entwicklung von Web Component zu unterstützen [7] (vgl. Taibi und Mezzalira 2022: 28). Stencil erstellen kleine, sehr schnelle Web Component, die überall ausgeführt werden können [86] (vgl. Stencil o. J.: o. S.). Das Ionic Framework-Team entwickelte Stencil als sogenannten [87] (vgl. Ionic Framewok o. J.: o. S.) „Web Component

Compiler“[86] (vgl. Stencil o. J.: o. S.).

In Wirklichkeit ist dies kein unabhängiges Framework, sondern ein Werkzeug, das für bestimmte Zwecke entwickelt wurde. Zu den wichtigsten Zielen gehören unter anderem die folgenden Punkte.

- „allows developers to scale framework-agnostic components across many projects, teams and large organizations.“[88] (Stencil o. J.: o. S.).
- „provides a compiler that generates highly optimized Web Components, and combines the best concepts of the most popular frameworks into a simple build-time too“[88] (Stencil o. J.: o. S.).

Außerdem ermöglicht dieses Werkzeug, durch die Bereitstellung geeigneter Funktionen, flexiblere Ausgaben zu erzielen, die in Zukunft nützlich und unterstützend sein können [7] (vgl. Taibi und Mezzalira 2022: 28).

Anwendungsfälle von Web Component

Web Component lassen sich problemlos in jede JavaScript-basierte Bibliothek oder jedes Framework integrieren, insbesondere in Bezug auf die Hypertext Markup Language (HTML) [85] (vgl. Webcomponent o. J.: o. S.). Allerdings sollten Web Component nicht für Microfrontend-Projekte verwendet werden, die eine gute Suchmaschinenoptimierung (SEO) erfordern, ähnlich wie iFrames. Mezzalira (2021: 96) schlägt vor, dass dynamisches Rendern eine geeignete Alternative sein kann, wenn SEO für das Projekt von entscheidender Bedeutung ist [6] (vgl. Mezzalira 2021: 96).

Vor- und Nachteile:

Web Component bieten zahlreiche Vorteile für die Entwicklung von Microfrontends. Sie sind vollständig kompatibel mit modernen Browsern wie Chrome, Firefox und Edge und arbeiten äußerst effizient [85] (vgl. Web component o. J.: o. S.). Zudem bieten sie eine hohe Flexibilität, da Entwicklungsteams die besten Tools und Frameworks entsprechend den Anforderungen des Projekts auswählen können [88] (vgl. stenciljs o. J.: o. S.). Die Bereitstellung von Web Component gestaltet sich einfach und unkompliziert. Mit ausreichenden Kenntnissen in Frontend-Technologien stellen Web Component eine zugängliche Möglichkeit dar, Microfrontends zu implementieren [6] (vgl. Mezzalira 2021: 97). Ferner sind Web Component skalierbar und ermöglichen eine effiziente Entwicklung von Microfrontends in Form von statischen Dateien zur Kompilierungs- und Laufzeit. Außerdem sind sie leicht testbar und ermöglichen verschiedene Arten von Tests ohne spezifische Probleme. Darüber hinaus bieten Web Component eine verbesserte Leistung, da sie nur die HTML-Komponenten erweitern und keine externen Bibliotheken laden müssen und sich dadurch positiv beeinflussen lassen [6] (vgl. Mezzalira 2021: 97).

Die Testbarkeit von Web Component stellt eine Herausforderung dar, die jedoch gemeistert werden kann. Ein umfassendes Verständnis der APIs ist hierbei unerlässlich. Während der Testphase ist es von großer Bedeutung, dass das Entwicklungsteam mit den APIs von Web Component vertraut ist, um mögliche Schwierigkeiten zu vermeiden [6] (vgl. Mezza-

lira 2021: 97).

Web Component sind mit den meisten aktuellen Browsern kompatibel. Allerdings gibt es eine grundlegende Herausforderung: „Web Components are supported only in the latest browsers versions“[7] (Taibi und Mezzalira 2022: 28), was zu Problemen führen kann, wenn Besucher ältere Browserversionen verwenden. Um dieses Problem zu lösen, empfehlen Taibi und Mezzalira (2022: 28) die Verwendung von Polyfills. Polyfills sind Code-Stücke, die älteren Browsern moderne Funktionen zur Verfügung stellen, die diese nicht nativ unterstützen [60] (vgl. MDN Web Docs o. J.: o. S.). Durch die Verwendung von Polyfills wird sichergestellt, dass das Projekt auf allen Browsern zugänglich ist und Besucher die Anwendung problemlos anzeigen können [7] (vgl. Taibi und Mezzalira 2022: 28). Vor Projektbeginn ist es empfehlenswert, die Website „caniuse“ [89] zu besuchen, um weitere Informationen zur Kompatibilität von Web Component mit den neuesten Browserversionen zu erhalten [90] (vgl. canuise o. J.: o. S.).

4.2 Server Side

Wie in vorherigen Abschnitten erläutert, zählen Web Component, iFrames und Module Federation zu den beliebtesten Ansätzen für die Implementierung von Microfrontends auf der Client Side. In diesem Abschnitt liegt der Fokus auf der Server Side und es werden weitere Aspekte der Implementierung von Microfrontends durch Server Side Composition untersucht.

Mezzalira (2020: o. S.) betont, dass die Server Side Composition sowohl für den Horizontal Split als auch für den Vertical Split eingesetzt werden kann [75] (vgl. 2020: o. S.). Außerdem stellt Mezzalira (2021: 98) fest, dass die Server Side Composition mit einer Horizontal Split Architektur eine der besten Lösungen für Microfrontends ist [6] (vgl. Mezzalira 2021: 98) . Damit verbunden, wird darauf hingewiesen, dass die Server Side Composition möglicherweise flexibler als andere Ansätze in der Horizontal Split Architektur zur Erstellung von Microfrontends sei [7, 75] (vgl. Taibi und Mezzalira 2022: 28; Mezzalira 2020: o. S.). Bei der Server Side Composition werden alle Microfrontends zu einer Seite aggregiert und auf der Ebene des Content Delivery Networks (CDN) gespeichert. Anschließend werden sie entweder zur Laufzeit oder zur Kompilierungszeit an den Client übertragen. Die Ansicht wird in diesem Szenario vom Ursprungsserver erstellt, der alle einzelnen Microfrontends abruft und schließlich die endgültige Seite zusammenstellt [6, 11, 75] (vgl. Mezzalira 2021: 31; Peltonen et al. 2021: 5; Mezzalira 2020: o. S.).

Die Abbildung 30 zeigt, dass zwischen dem Browser und den Anwendungsservern ein Dienst existiert, der die Operationen dieser Art der Composition selbst durchführt [1] (vgl. Geers 2020: 60).

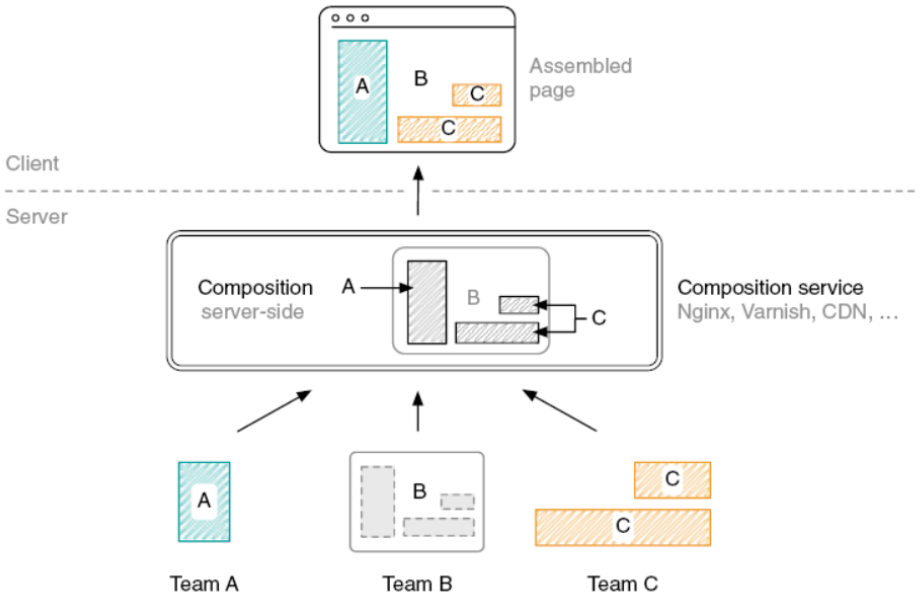

Abbildung 30 Serverseitige Fragmentzusammenstellung für den Client
(Quelle [1] Geers 2020: 60)

Der Hauptvorteil der Sever Side Composition besteht darin, dass die Seite vollständig komponiert ist, bevor sie den Browser des Endnutzers erreicht [1] (vgl. Geers 2020: 60). Wenn

die Seite in großem Umfang zwischengespeichert werden kann, wird sie vom CDN mit einer langen Speicherungszeit (TTL) bereitgestellt. Es ist wichtig, die Skalierbarkeit der Webseite zu berücksichtigen, insbesondere wenn eine Anpassung für jeden Nutzer erfolgt, da viele Anfragen von verschiedenen Nutzern eingehen werden [6, 11, 75] (vgl. Mezzalira 2021: 31, Peltonen et al. 2021: 5, Mezzalira 2020: o. S.). Für Entwickler, die Laufzeitkomposition für Server Side Microfrontends einsetzen wollen, ist es wichtig, die Anwendungsfälle gründlich zu analysieren und zu untersuchen. Dies erfordert die Implementierung einer spezifischen serverseitigen Skalierungsstrategie, um Ausfallzeiten für die Nutzer zu minimieren [6, 11] (vgl. Mezzalira 2021: 31, Peltonen et al. 2021: 5).

Für die Umsetzung der Sever Side Composition stehen spezielle Open-Source-Frameworks wie Tailor.js, Opencomponents, Ara, Podium und Puzzel.js zur Verfügung [1, 6, 7, 46] (vgl. Geers 2020: 73-81; Mezzalira 2021: 102; Taibi und Mezzalira 2022: 28; Mezzalira 2019: o. S.).

Anwendungsfälle von Server Side

Wie bereits erwähnt, betont Mezzalira (2020: o. S.) den entscheidenden Unterschied zwischen Client- und Server-Side Composition, indem er feststellt, dass die Verwendung von Client Side Composition sinnvoll ist, wenn man die volle Kontrolle über die Endziele haben möchte. Im Gegensatz dazu bietet die Server Side Composition die maximale Kontrolle über die Ausgabe [75] (vgl. Mezzalira 2020: o. S.).

Websites wie Nachrichtenportale oder E-Commerce-Plattformen, die eine stärkere Indizierung benötigen, sowie Unternehmen wie PayPal oder American Express, die eine hohe Performance benötigen, können von der Server Side Composition profitieren und sie als eine sehr gute Option betrachten [6] (vgl. Mezzalira 2021: 42). Auch Unternehmen wie Amazon, Zalando und IKEA haben die Server Side Composition genutzt, um von monolithischen zu Microfrontend Architekturen zu migrieren [1] (vgl. Geers 2020: 59). Die Entscheidung, Server Side Composition in Projekten einzusetzen, wird von verschiedenen Faktoren beeinflusst. Im Folgenden werden die wichtigsten Faktoren diskutiert, die den Einsatz dieses Ansatzes beeinflussen [1, 6, 7] (vgl. Geers 2020: 83; Mezzalira 2021: 106; Taibi und Mezzalira 2022: 28).

Geers (2020: 83), Mezzalira (2021: 98) und Taibi und Mezzalira (2022: 28) sind der Ansicht, dass Server Side Composition nicht nur die Performance verbessern kann, sondern auch die beste Option ist, insbesondere wenn SEO-Anforderungen in einem Projekt von Bedeutung sind [1, 6, 7] (vgl. Geers 2020: 83; Mezzalira 2021: 98; Taibi und Mezzalira 2022: 28). Mezzalira (2021: 98) hebt hervor, dass diese Technik die Seitenladezeit beschleunigt und die Seite vollständig gerendert wird, ohne dass JavaScript-Logik benötigt wird [6] (vgl. Mezzalira 2021: 98). Darüber hinaus wird betont, dass SSR häufig auch zur Verbesserung der SEO eingesetzt wird, da die Ladegeschwindigkeit einer Seite für Suchmaschinen von großer Bedeutung ist [6] (vgl. Mezzalira 2021: 98). Um festzustellen, in welchen Situationen die Verwendung der Server Side Composition geeignet sein könnte, werden einige Beispiele

genauer betrachtet.

Im Zusammenhang mit Suchmaschinenoptimierung (SEO) ist es insbesondere bei Business-to-Consumer (B2C)-Websites wichtig, dass die Inhalte für Suchmaschinen leicht zugänglich sind [6] (vgl. Mezzalira 2021: 106). Um dies zu veranschaulichen, wird angenommen, dass eine B2C-Website betrieben wird, deren Hauptzweck der direkte Verkauf von Produkten an Endkunden ist. Aufgrund der umfangreichen Produktpalette auf der Website ist es entscheidend, dass Suchmaschinen die Produkte schnell und effizient identifizieren können, um den Kunden präzise Suchergebnisse zu liefern. Da die Zugänglichkeit der Inhalte einen großen Einfluss auf die Nutzererfahrung und letztendlich auf die Verkaufsergebnisse hat, ist die Verwendung von Server Side Composition hier äußerst effektiv und wird für diese Art von Websites dringend empfohlen.

Wie von Geers (2020: 83) erwähnt, könnte dieser Ansatz auch bei der Entwicklung interner Anwendungen nützlich sein, bei denen die Benutzerinteraktion nicht im Vordergrund steht. Für solche Anwendungen ist die Server Side Composition die logischere Wahl [1] (Geers 2020: 83).

Vor- und Nachteile

Die Server Side Composition bietet eine hervorragende Testbarkeit im Vergleich zu anderen Ansätzen, da die Anwendungen mit SSR eine hohe Ähnlichkeit aufweisen, was die Testmethodik erleichtert [6] (vgl. Mezzalira 2021: 107). In Bezug auf die Performance ermöglicht dieser Ansatz eine außergewöhnliche Qualität im Vergleich zu anderen Ansätzen, da das Entwicklungsteam die volle Kontrolle über das Endergebnis hat und das Programm optimal gestalten und optimieren kann [6] (vgl. Mezzalira 2021: 107).

Hinsichtlich der Skalierbarkeit kann Server Side Composition in Projekten mit großen Aufgaben und hohen Anforderungen jedoch problematisch sein, da eine Skalierung des Backends, das die endgültige Seite erstellt, erforderlich ist [6] (vgl. Mezzalira 2021: 107). Die Implementierung dieses Ansatzes ist sehr anspruchsvoll, da er viele bewegliche Komponenten im Frontend und Backend erfordert, was zu einer hohen Komplexität führt, obwohl er eine enorme Performance bietet [6] (vgl. Mezzalira 2021: 107).

4.3 Edge Side

Ein weiterer Ansatz zur Umsetzung der Microfrontend Architektur ist die Edge Side Composition (ESC), die ausschließlich im Horizontal Split realisiert werden kann. Die Edge Side Composition wird auf der Ebene des Content Delivery Network (CDN) realisiert [6, 11, 75] (vgl. Mezzalira: 2021: 30; Peltonen et al. 2021: 5; Mezzalira 2020: o. S.), wobei aufgrund der Architektur des CDN hervorragende Skalierungsmöglichkeiten gegeben sind [6] (vgl. Mezzalira 2021: 108).

Die Generierung der Benutzerschnittstelle erfolgt auf den Servern des CDNs und nicht im Browser oder auf dem Server des Nutzers. Die Edge Side Composition führt die Microfrontends auf CDN-Ebene zusammen und liefert das Endergebnis an den Client, also den Browser des Nutzers [6, 75] (vgl. Mezzalira 2021: 108; Mezzalira 2020: o. S.). Viele CDN-Anbieter ermöglichen die Verwendung von Edge Side Includes (ESI), einer auf XML basierenden Auszeichnungssprache.

ESI ist keine neue Sprache und wurde bereits 2001 von Unternehmen wie Akamai und Oracle als Standard vorgeschlagen [6, 7, 11, 75] (vgl. Mezzalira 2021: 30; Taibi und Mezzalira 2022: 28; Peltonen et al. 2021: 5; Mezzalira 2020: o. S.). „ESI uses a technique called transclusion for including existing content inside a new document without the need to duplicate it.“[6] (Mezzalira 2021: 109). ESI trägt zur Skalierbarkeit einer Web-Infrastruktur bei, indem Inhalte effizient und schnell an Nutzer weltweit geliefert werden, wobei ein CDN-Netzwerk eine Vielzahl von Points of Presence weltweit nutzt [6, 75] (vgl. Mezzalira 2021: 30. f.; Mezzalira 2020: o. S.).

Die Nutzung von ESI ist jedoch kompliziert, da jeder CDN-Anbieter ESI anders implementiert. Die Implementierung von ESI kann zu zahlreichen Überarbeitungen und der Einführung neuer Logik führen, sei es durch die Anwendung einer Multi-CDN-Strategie oder durch den einfachen Wechsel des Anbieters [6, 11, 75] (vgl. Mezzalira 2021: 30; Peltonen et al. 2021: 5; Mezzalira 2020: o. S.).

Anwendungsfälle der Edge Side

Die Edge Side Composition ist nur mit Horizontal Split kompatibel, da die Transklusion auf CDN-Ebene bei Vertical Split keinen Sinn ergibt. In diesem Abschnitt werden einige Fälle betrachtet, in denen die Verwendung der Edge Side Composition für die Implementierung von Microfrontends nützlich sein kann.

Die Edge Side Composition ermöglicht es, die Skalierbarkeitsprobleme bei Projekten mit statischen Inhalten und hohem Datenverkehr an den CDN-Anbieter zu delegieren, anstatt sich in der eigenen Infrastruktur damit auseinandersetzen zu müssen [6] (vgl. Mezzalira 2021: 42). Statische Websites, die skaliert werden müssen und aus verschiedenen Teams bestehen, die nicht jedes Detail kontrollieren müssen, können von der Anwendung dieser Methode profitieren. Dies ist besonders nützlich für Websites wie Nachrichtenportale und Online-Kataloge, die keinen Bedarf an dynamischen Inhalten haben. Ein bekanntes Unternehmen, das von dieser Strategie profitiert hat, ist IKEA, das eine Kombination aus Edge-Side-Includes und Client-Side-Includes verwendet [6, 75] (vgl. Mezzalira 2021: 108

ff.; Mezzalira: 2020: o. S.).

Vor- und Nachteil

Die Skalierbarkeit dieses Ansatzes liegt darin, dass durch die Kombination von CDN-Ebenen und das Vorhandensein statischer Inhalte die Skalierbarkeit dem CDN-Anbieter überlassen werden kann, was eine Skalierbarkeit für diesen Ansatz ermöglicht [6] (vgl. Mezzalira 2021: 112). Weiterhin ist die Leistung ein wichtiger Faktor und die Verwendung von ESI für Webseiten mit statischem Inhalt bietet eine herausragende Option. Durch den Cache-Speicher kann die Anwendung eine hohe Leistung für diese Seiten erzielen. Es ist jedoch zu beachten, dass gelegentliche Netzwerkprobleme die Leistung beeinträchtigen können, da Seiten hängen bleiben und keine der Seiten bedient wird, bis die Anfrage abgelaufen ist [6] (vgl. Mezzalira: 2021: 112).

Die Developer Experience spielt eine entscheidende Rolle für den Erfolg jedes Projekts. Die Implementierung von Microfrontends in ESI stellt eine große Herausforderung dar. Entwickler benötigen eine Umgebung, die es ihnen ermöglicht, mit den Herausforderungen der Entwicklung, neuen Funktionen und komplexen Algorithmen umzugehen. Die Umsetzung von ESI kann sehr anspruchsvoll sein, weshalb Entwickler oft lokale Tests durchführen müssen, um sicherzustellen, dass ihre Anwendung ordnungsgemäß funktioniert. Um einen Testserver wie Varnish, NGINX oder Akamai zu betreiben und seine Implementierung zu testen, benötigen Sie eine virtuelle Umgebung oder Docker-Container. [6] (vgl. Mezzalira 2021: 110 ff.).

Die Testbarkeit ist ein weiteres Problem bei der Verwendung dieses Ansatzes. Entwickler sind häufig gezwungen, zusätzliche Tools für lokale Tests zu verwenden, was eine sehr komplexe Infrastruktur erfordert und den Feedback-Zyklus eines Teams verlangsamen kann [6] (vgl. Mezzalira 2021: 112). Die Komplexität dieses Ansatzes wird als großer Nachteil angesehen, da die Implementierung von Microfrontends unter Verwendung dieser Methode äußerst schwierig und anspruchsvoll ist. Diese Methode kann aufgrund mangelnder Entwicklerkenntnisse und der Notwendigkeit zusätzlicher Tools für Code-Tests übermäßig kompliziert sein [6] (vgl. Mezzalira 2021: 112).

Ein weiterer Nachteil dieser Methode ist die Koordination zwischen den Teams, da Änderungen am Inhalt des Projekts möglicherweise alle Teams beeinflussen und daher eine erhöhte Koordination erforderlich ist [6] (vgl. Mezzalira 2021: 112).

5 Fallstudien

In diesem Kapitel wird die letzte Forschungsfrage dieser Abschlussarbeit behandelt: Welche Erfahrungen haben Unternehmen gemacht, die bereits Microfrontends implementiert haben, und welche Empfehlungen können daraus abgeleitet werden? Bevor dieses Kapitel beginnt, könnte die Frage aufkommen, warum Unternehmen die Entscheidung zur Migration zu einer Microfrontend Architektur für ihre Programme getroffen haben und welche Gründe hinter dieser Entscheidung liegen.

Es ist erkennbar, dass Unternehmen heutzutage nach Lösungen suchen, um ihre Benutzeroberflächen flexibler zu gestalten, die Aktualisierung zu erleichtern, die Frontend-Entwicklung zu vereinfachen, die Skalierbarkeit zu verbessern und die Wartbarkeit zu erhöhen. Aus diesem Grund hat die Implementierung von Microfrontends in den letzten Jahren erheblich an Bedeutung gewonnen [1, 91] (vgl. Geers 2020: 4; OpenSourcee 2023: o. S.). Unternehmen mit einer umfangreichen Online-Präsenz und hohem Traffic können von der Verwendung der Microfrontend Architektur profitieren. Microfrontends erfreuen sich in verschiedenen Branchen immer größerer Beliebtheit, darunter E-Commerce, Buchungsplattformen, soziale Netzwerke und komplexe Webanwendungen [1] (vgl. Geers 2020: 21).

Die Microfrontend Architektur wird von verschiedenen Unternehmen im E-Commerce-Bereich eingesetzt: Die Otto Group, ein deutsches Versandhandelsunternehmen und einer der größten E-Commerce-Anbieter weltweit, begann 2012 mit der Aufspaltung ihres Monolithen , Auch IKEA, ein schwedisches Möbelunternehmen, und Zalando, einer der größten Modehändler Europas, haben dieses Modell übernommen Die deutsche Buchhandelskette Thalia hat ihren E-Reader-Store in verticale Slices umgewandelt, um die Entwicklungsgeschwindigkeit zu erhöhen. Amazon hingegen gibt keine detaillierten Einblicke in seine interne Entwicklungsstrategie [1, 6, 7] (vgl. Geers 2020: 21; Mezzalira 2021: 36 ff.; Taibi und Mezzalira 2022: 25).

Einige große Unternehmen in anderen Branchen haben ebenfalls die Microfrontend Architektur eingeführt. Spotify und SAP organisieren sich in autonomen End-to-End-Teams . SAP hat ein Framework namens Luigi [73] veröffentlicht, um verschiedene Anwendungen zu integrieren. Der Sport-Streaming-Dienst DAZN hat sein monolithisches Frontend ebenfalls als Microfrontend-Architektur neu aufgebaut [1, 6, 7] (vgl. Geers 2020: 21; Mezzalira 2021: 36 ff.; Taibi und Mezzalira 2022: 25).

Des Weiteren haben Unternehmen wie HelloFresh, AllegroTech, OpenTable, Netflix, NewRelic, Starbucks, Skyscanner und American Express Microfrontends implementiert [6, 7, 91, 92] (vgl. Mezzalira 2021: 36ff.; Taibi und Mezzalira 2022: 25; OpenSourcee 2023: o. S.; Mezzalira 2019a: o. S.).

Der größte Vorteil dieser Implementierungen liegt in der Skalierbarkeit, da die Anwendungen in separate Module aufgeteilt werden können. Dies ermöglicht eine schnelle und eigenständige Umsetzung von Änderungen sowie eine verbesserte Ressourcenverwaltung. Obwohl diese Unternehmen als Pioniere der Microfrontend Architektur gelten, ist es möglich, dass auch andere Unternehmen diese Architektur weiterhin anwenden oder untersu-

chen werden.

5.1 Zalando

Zalando war das erste Unternehmen, das das Projekt Mosaic Server Side Composition zur Migration von monolithischer zu Microfrontend Architektur einsetzte. Mosaic wurde unter Verwendung von Tailor.js, einer Node.js-Bibliothek, erstellt [1, 6] (vgl. Geers 2020: 73.; Mezzalira 2021: 102).

Das Mosaic-Projekt umfasst Frameworks und Tools, die aufgrund ihrer Bekanntheit Aufmerksamkeit erregt haben. Es gibt viele Forks von Tailor.js in diesem Projekt, die für die Kommunikation und Integration verschiedener Teile von HTML-Fragments verwendet werden [1, 6] (vgl. Geers: 2020: 73; Mezzalira 2021: 102). Die Entwickler haben jedoch beschlossen, eine neue Version des Mosaic-Projekts zu veröffentlichen, in der das bekannte Tailor-Framework durch das Interface Framework ersetzt wird. Das Interface Framework ähnelt den Konzepten von Tailor, basiert jedoch auf React und GraphQL. Der Fokus liegt dabei mehr auf Komponenten als auf Fragmenten [6] (vgl. Mezzalira 2021: o. S.).

5.2 Netflix

Der Streaminganbieter Netflix [93] nutzt die Microfrontend Architektur. Laut einem Blogbeitrag des Netflix TechBlogs aus dem Jahr 2021 wurden gemeinsame Muster und Architekturen in verschiedenen Tools identifiziert. Um die Skalierbarkeit mit anderen Teams zu verbessern, wurden diese Werkzeuge integriert. Daher hat sich das Unternehmen dazu entschieden, die Microfrontend Architektur zu verwenden und das interne Framework namens Lattice [94] zu entwickeln und zu veröffentlichen [95] (vgl. Possumato et al. 2021: o. S.).

„Lattice is a tiny framework that provides an abstraction layer for React web applications to leverage.“[95] (Possumato et al.: 2021)

Lattice zielt darauf ab, interne Abhängigkeiten in einem Projekt zu vereinfachen. Netflix hingegen wollte alle externen Abhängigkeiten eliminieren, die durch die Anforderungen jeder Quelle entstehen könnten. In ihrer Client-Anwendung, die hauptsächlich aus React und Typescript besteht, erleichtert Lattice die Verwendung gemeinsamer Zustände mit Webpack. Ein weiteres Ziel bei der Entwicklung von Lattice war es, eine Plattform auf Basis des internen Technologiestacks des Unternehmens zu schaffen. Lattice ist ein React-basiertes Framework, das eine schnelle und einfache Implementierung dieser Technologie ermöglicht [95] (vgl. Possumato et al. 2021: o. S.).

Darüber hinaus ermöglicht dieses Framework Entwicklern, sich auf ihr Hauptprodukt zu konzentrieren, während sie Teile der Anwendung, die über externe anpassbare Plugins verfügbar sind, wrapped gestalten können. Der Kern dieses Frameworks bietet die Fähigkeit zur asynchronen Lastübertragung von Fernmodulen durch die Modul-Federation [95] (vgl. Possumato et al. 2021: o. S.).

5.3 Spotify

Wie bereits erklärt wurde, können iFrames in Desktop-Anwendungen eingesetzt werden. Spotify, ein Online-Media-Streaming-Dienst, hat für seine Desktop-Anwendungen Microfrontend verwendet und von iFrames profitiert, um verschiedene Aspekte seiner Anwendungen zu integrieren [1, 6, 92] (vgl. Geers 2020: 36; Mezzalira 2021: 91; Mezzalira 2019a: o. S.). Es ist zu beachten, dass die Verwendung von iFrames in Desktop-Anwendungen Vorteile bietet. Um zwischen den iFrames zu kommunizieren, wurde der Event Bus verwendet. Dieses Ereignis ermöglicht die Kommunikation zwischen verschiedenen Komponenten der Anwendung [92] (vgl. Mezzalira 2019a: o. S.). Durch diese Methode ist es der Desktop-Anwendung möglich, keine Abhängigkeiten für das Rendern eines Microfrontends herunterzuladen. Das bedeutet, dass alle Komponenten mit einem einzigen ausführbaren Programm verfügbar sind [6] (vgl. Mezzalira 2021: 91).

Der iFrame-Ansatz war für Spotify akzeptabel, da das Gesamtdesign der Anwendung statisch war und keine Suchmaschinen benötigte [1] (vgl. Geers 2020: 36). Außerdem hat das Unternehmen die Idee von Infrastructure Squads eingeführt, die in gewisser Weise mit A/B-Testing-Tools vergleichbar sind [1] (vgl. Geers 2020:19).

5.4 IKEA

Gemäß Thinksys (2023: o. S.) hat das schwedische Möbelunternehmen IKEA die Microfrontend Architektur als Hauptstrategie zur Steigerung der Flexibilität seiner Plattform übernommen [96] (vgl. Thinksys 2023: o. S.). Mezzalira (2019a: o. S.) verfolgte einen anderen Ansatz, der eine Integration von ESI und CSI umfasst [92] (vgl. Mezzalira 2019a: o. S.).

Gustaf Nilsson Kotte, einer der leitenden Web-Ingenieure bei IKEA, hat in seinem persönlichen Blog Details darüber veröffentlicht, warum IKEA sich für die Verwendung der Microfrontend Architektur entschieden hat. Er erklärt, dass das Team beschlossen hat, ihr monolithisches E-Commerce-System namens IKEA Retail Web (IRW), das seit mehr als zwei Jahrzehnten im Einsatz war, durch die Microfrontend Architektur zu ersetzen [103] (vgl. Kotte 2023: o. S.).

Die Hauptzielsetzung von IKEA bestand darin, die Website innerhalb von drei bis vier Monaten mit qualitativ hochwertigem Inhalt zu verbessern und aufzurüsten. IKEA suchte eine Lösung, um die Website schnellstmöglich auf den Markt zu bringen, den durch Besucher erzeugten Traffic optimal zu verwalten und eine exzellente Leistung der Website in allen Ländern zu gewährleisten, ohne zusätzliche Last auf die Entwicklerteams zu legen. Gleichzeitig sollte die Sicherheit der Website gegen Cyberangriffe gewährleistet sein [97] (vgl. Kotte 2023: o. S.).

IKEA begann mit der anfänglichen Nutzung von statischen Dateien und der Zusammenarbeit mit einem Content Delivery Network (CDN). Allerdings standen sie bald vor einer grundlegenden Herausforderung: Änderungen an Schlüsselelementen der Website wie dem Header erforderten eine vollständige Neuerstellung der gesamten Website [97] (vgl. Kotte 2023: o. S.). Unter Berücksichtigung der Erfahrungen einiger Teammitglieder im Umgang

mit Edge Side Includes (ESI) hat sich IKEA dazu entschieden, diesen Ansatz zur Erleichterung der Implementierung der Microfrontend Architektur zu nutzen [97] (vgl. Kotte 2023: o. S.). Da der CDN ESI unterstützt, war dies eine einfache Entscheidung.

Gemäß dem Zitat von Kotte (2023: o. S.) könnte beispielsweise die Header-Navigation als eigenes HTML-Fragment extrahiert und mittels ESI wiederverwendet werden. Für die Klärung eines Teamproblems sei es so notwendig, das Team in drei Gruppen zu gliedern: CMS, Produktpräsentation und Framework [97] (vgl. Kotte 2023: o. S.).

Nach Abschluss des Projekts konnte IKEA eine Reduktion der Entwicklungszeit um etwa 50 Prozent verzeichnen, was auch zu einer 75-prozentigen Reduzierung der Ladezeiten der Seiten führte, wie OpenSourcee (2023: o. S.) und Thinksys (2023: o. S.) beschreiben. IKEA betonte, dass sie signifikante Verbesserungen in Bezug auf Code-Wartung und schnellere Bereitstellung von Funktionen erreicht hatten [91, 96] (vgl. OpenSourcee 2023: o. S.; Thinksys 2023: o. S.).

Kotte (2023: o. S.) betonte in seinem Blog, dass eine der Erfolgsgründe für das IKEA-Projekt die Beibehaltung einer kleinen Teamgröße war [97] (vgl. Kotte 2023: o. S.). Dieses Prinzip stimmt mit dem „2-Pizza-Team“-Gesetz überein, auf das auch der Manager von Amazon hingewiesen hatte. Ein Team ist zu groß, wenn es mehr Personen umfasst als mit zwei Pizzen gefüttert werden zu können [104] (vgl. Amazon: o. S.).

6 Diskussion

Die Entwicklung einer qualitativ hochwertigen Webanwendung, die in der Lage ist, alle Bedürfnisse zu erfüllen, ist keineswegs einfach. Sie hängt von verschiedenen Faktoren ab, darunter die Erfahrung des Entwicklungsteams, die Auswahl der geeigneten Architektur, das vollständige Verständnis der Projektdetails, die Festlegung klarer Ziele, die Lösung von Sicherheitsproblemen, der Zugang zu benötigten Ressourcen und die organisatorischen Strukturen. All diese Faktoren können eine wichtige Rolle dabei spielen, den Entwicklungsprozess einer hochwertigen Webanwendung zu beeinflussen. Wenn die Webanwendung von hoher Qualität ist, werden die Zufriedenheit der Benutzer und die Benutzerpräsenz auf der Website gewährleistet. In der modernen Webwelt spielt die Webarchitektur eine entscheidende Rolle und kann selbst einen großen Beitrag zum Erfolg eines Projekts leisten.

Im zweiten Kapitel wurden verschiedene Webarchitekturen wie Dreischicht-, Monolith-, Microservices-, SPA- und MPA-Architekturen ausführlich untersucht. Jede dieser Architekturen kann je nach den Zielen des Projekts und den spezifischen Anforderungen genutzt werden.

Die grundlegende Frage lautete in diesem Zusammenhang, welche Faktoren zur Entstehung dieser Architekturen beigetragen haben. Diese Frage lässt sich mithilfe der ausführlichen Analyse in dieser vorliegenden Masterarbeit zweifellos damit beantworten, dass die Faktoren wie die Skalierbarkeit, Flexibilität, die leichtere Wartbarkeit von Projekten, die Verbesserung der Arbeit von Entwicklerteams und weitere Aspekte hier eine bedeutende Rolle spielen. Nichtsdestotrotz gehört zu den zentralsten Faktoren in der Entwicklung dieser neuartigen Architekturen die Tatsache, dass Webanwendungen zu einem wesentlichen Bestandteil des alltäglichen Lebens geworden sind. Wenn eine Website oder Anwendung entwickelt wird und sie nicht gut mit dem Benutzer kommunizieren kann und nicht effektiv auf die Bedürfnisse des Benutzers reagiert, wird sie zweifellos mit Misserfolg gegenüber Konkurrenten konfrontiert sein. Tatsächlich kann die richtige Wahl der Architektur erheblich zum Erfolg des Projekts beitragen.

Dennoch werden, wie ersichtlich wurde, neue Architekturen gerade im Entstehen begriffen. Der Schwerpunkt dieser Abschlussarbeit lag auf der Untersuchung und dem Vergleich verschiedener Ansätze zur Implementierung von Microfrontends, die als eine innovative Architektur gelten und bisher nicht ausreichend erforscht und untersucht wurden. Da jede Architektur für spezifische technische und organisatorische Herausforderungen entworfen wurde, wird mithilfe dieser Arbeit deutlich, dass es im Bereich der Microfrontendarchitektur verschiedene Ansätze zur Implementierung gibt, um die besten Wege zur Erreichung der Ziele auszuwählen.

Wie im Kapitel 3 dargelegt wurde, ermöglichen Microfrontends, den Frontend-Bereich in kleinere Einheiten aufzuteilen. Diese Aufteilung beschleunigt nicht nur den Entwicklungs- und Bereitstellungsprozess, sondern ermöglicht es auch, jede Einheit separat zu testen und zu warten. Angesichts der beiden Hauptziele der Einführung der Microfrontend, nämlich Reduzierung der Entwicklungszeit und Steigerung der Skalierbarkeit des Frontendbereichs, die in der Vergangenheit nicht dringlich erschienen, wird deutlich, dass diese Skalierbar-

keitsfähigkeit heute aufgrund verschiedener Veränderungen und Verbesserungen in der Benutzererfahrung sowie der zunehmenden Komplexität von Anwendungen klar spürbar ist und schnell zu einem großen Problem werden kann. Wie die Autoren Mezzalira (2021) und Geers (2020) betonen, ist es vor der Annahme Microfronted wichtig, sorgfältig darüber nachzudenken und zu recherchieren, um sicherzustellen, dass sie mit den Zielen des Projekts kompatibel ist und die Anforderungen des Projekts ordnungsgemäß erfüllt.

In dieser Abschlussarbeit bezieht sich eines der Bereiche, der weiterer Forschung bedarf auf die Untersuchung der Anwendungsfälle von Microfrontends. Wie bereits erwähnt wurde, sind Microfrontends keineswegs wie ein Silver Bullet zu betrachten, welche die Fähigkeit hat, alle Probleme zu lösen. In einigen spezifischen Situationen kann die Verwendung von Microfrontend nützlich sein, während sie in anderen Fällen möglicherweise nicht die gewünschte Leistung bietet.

Angesichts der Tatsache, dass die Verwendung von Microfrontends dazu beitragen kann, die Skalierbarkeit in Projekten zu erleichtern [1] (vgl. Geers 2020: 19), lässt sich schließen, dass in großen und mittelgroßen Projekten die Auswahl dieser Architektur eine geeignete Option darstellen könnte, um die Skalierbarkeit zu verbessern und die damit verbundenen Probleme zu reduzieren. Durch die Unterteilung in kleinere Einheiten ermöglicht diese Architektur eine schnellere Verwaltung und Bereitstellung, was in umfangreichen Projekten mit hohem Bedarf an Skalierbarkeit dazu beitragen kann, die damit verbundenen Herausforderungen zu verringern. Andererseits erscheint die Verwendung von Microfrontends sinnvoll in Projekten, in denen bestimmte Teile der Anwendung aktualisiert werden müssen, ohne dass dies Auswirkungen auf andere Teile hat. Tatsächlich ist die Anwendung von Microfrontends nicht auf einen bestimmten Bereich beschränkt und kann in verschiedenen Szenarien eingesetzt werden. Dennoch entfalten sie ihre größte Leistungsfähigkeit und ihren besten Nutzen in Webanwendungen. Obwohl Microfrontends auch in nativen Anwendungen verwendet werden können, sollte nicht vergessen werden, dass dies zusätzlichen Aufwand und Komplexität in den Entwicklungsprozess einbringen kann.

Des Weiteren wurde nach der intensiven Auseinandersetzung mit der Forschungsliteratur deutlich, dass die Autoren Geers (2021: 21) und Brooks (2023: o. S.) im Kollektiv die Meinung vertreten, dass die Verwendung von Microfrontends in kleinen Projekten, die nur aus wenigen Entwicklungsteams bestehen und keine komplexen Kommunikationsanforderungen haben, nicht sinnvoll und zielführend seien [1, 71] (vgl. Geers 2021: 21, Brooks 2023: o. S.). Gerade dieser Punkt kann die Grundlage für ein kritisches Diskussionsthema darstellen. Es ist anzunehmen, dass, wenn ein Entwickler oder eine gewöhnliche Person beschließt, einen Onlineshop zu starten, es wichtig ist, die Entwicklungen der Zukunft zu berücksichtigen, unabhängig davon, ob das Projekt anfangs klein ist und nur begrenzte Dienstleistungen anbietet. Obgleich die Zukunftsprognose in Bezug auf das Feedback für die Website ungewiss ist und es nicht bekannt ist, inwiefern in der Zukunft Produkte hinzuzufügen oder Änderungen vorzunehmen wären. Wenn die kleine Website an Fahrt gewinnt und eine monolithische Architektur im Frontend verwendet wird, ermöglicht dies nicht, Änderungen leicht auf der Website vorzunehmen, und es besteht die Wahrscheinlichkeit, dass Problemen wie die Skalierbarkeit entstehen werden. Andererseits ist der stetige Fort-

schritt in der Technologie zu berücksichtigen, wodurch es möglich ist, später einen Teil der Website mit neuer Technologie zu aktualisieren, um durch die Verwendung geeigneter Architekturen die Website mit minimalem Aufwand und Schwierigkeiten zu optimieren.

Ich vertrete in diesem Diskurs die Meinung, dass es sinnvoll ist, die Zukunft bei der Auswahl der Architektur für das eigene Projekt im Auge zu behalten.

Eine der grundlegenden Herausforderungen, die bei der Implementierung von Microfrontends für Entwickler auftreten können, ähnlich wie bei der Microservices-Architektur, bezieht sich auf die Sicherheit. Obwohl die Verwendung von Microfrontends für Entwickler erhebliche Vorteile bietet, stellt sich die Frage, wie sichergestellt werden kann, dass das mit Microfrontends erstellte Programm in Bezug auf die Sicherheit ein maximales Maß an Schutz und Sicherheit bietet.

Aufgrund der Komplexität dieser Architektur und der Aufteilung des Systems in kleinere Einheiten steigen auch die Sicherheitsherausforderungen. Es werden Fragen aufgeworfen, die möglicherweise Zweifel an der Verwendung von Microfrontends in Projekten aufkommen lassen. Obwohl die Aufteilung des Programms in kleinere Einheiten die Entwickler erleichtern kann, kann diese Unterteilung auch Sicherheitsprobleme verstärken. Möglicherweise tauchen potenzielle Fragen auf, die geklärt und gelöst werden müssen, was zu Unsicherheiten bezüglich der Verwendung von Microfrontends in Projekten führen kann.

Ist es notwendig, für jeden Teil des Programms spezifische Sicherheitsmaßnahmen zu berücksichtigen, und gibt es eine Lösung, um Sicherheitsprobleme in Microfrontends zu stabilisieren? Obwohl Microfrontends von der Architektur der Microservices inspiriert sind, warum wurde das Sicherheitsproblem nicht vor der Erstellung von Microfrontends berücksichtigt?

Entwickler sollten sich bewusst sein, dass die Sicherheit von Microfrontends nicht nur auf der Server Side, sondern auch auf der Client Side beachtet werden muss. Die Implementierung von bewährten Sicherheitspraktiken, regelmäßige Sicherheitsüberprüfungen und Schulungen für Entwickler in sicheren Codierungspraktiken sind entscheidend, um die Integrität und Sicherheit von Microfrontends zu gewährleisten.

Ein weiteres Problem, das mit den Kommunikationen zwischen den Microfrontend verbunden ist, erfordert ähnlich wie bei den Microservices zusätzliche Tools für die Interaktion miteinander. Eine wichtige Frage wird aufgeworfen: Warum wurde bei der Entwicklung dieser Architektur nicht aktiv nach einer logischen Lösung für die Kommunikationsprobleme gesucht, die bereits bei den Microservices vorhanden waren? Obwohl diese Architektur positive Erfahrungen aus den Microservices übernommen hat, bleibt die Frage, warum keine effektiven Lösungen für die Kommunikationsprobleme in dieser Architektur präsentiert wurden.

In Bezug auf die Implementierungsansätze von Microfrontends wurden Eigenschaften untersucht, die dazu beitragen, ein besseres Verständnis für die Auswahl des geeigneten Ansatzes und die Erfüllung der Projektziele bei der Implementierung von Microfrontends zu gewinnen. Allerdings stieß ich während der Recherchen auf eine grundlegende Heraus-

forderung, nämlich den Mangel an spezialisierten Ressourcen in diesem Bereich und die Meinungsverschiedenheiten der Autoren.

Exemplarisch ist in diesem Zusammenhang Geers (2020: 11) hinsichtlich der Verwendung von ESI in Serve Side Composition zu nennen. Dies steht im Gegensatz zu Mezzalira (2021: 108), welcher ESI im Wesentlichen als eine Technik bezeichnet, die bei der Edge Side Composition verwendet wird. Mezzalira (2021: 108) verweist hierbei außerdem auf keinerlei Hinweise, inwiefern diese Technik auf der Serverseite umsetzbar ist. Dies führt während der Analyse zu Irritationen darüber, ob diese Technik für die Server Side oder die Edge Side verwendet wird [1, 6] (vgl. Geers 2020: 11, Mezzalira 2021: 108).

Ferner behauptet Geers (2020: 11), dass Taylor und Podium für die Server Side verwendet werden können. Nach einer vertieften Auseinandersetzung konnte in der Forschung von unter anderem Mezzalira (2021:102) die Ansicht gefunden werden, dass dies tatsächlich spezialisierte Microfrontend-Frameworks sind, die Entwicklern bei der Server Side Entwicklung helfen und zur Implementierung von Microfrontends auf der Server side beitragen können [1, 6] (vgl. Geers 2020:11, Mezzalira 2021: 102).

Ein weiterer zu kritisierender Aspekt betrifft ebenso Geers (2020). Dort wird zur Client Side Composition jede Thematik an verschiedenen Stellen des Werkes behandelt. Geers erläutert zunächst iFrames (vgl. Geers 2020: 33) diskutiert Ajax (vgl. Geers 2020: 42) und beleuchtet dann den Ansatz der Web Componenten (vgl. Geers 2020: 85). Die Herausforderung, die sich mir hierbei aufdrängt, besteht erstens darin zu verstehen, warum diese Aspekte nicht sequenziell und in einem dedizierten Kapitel behandelt werden. Zweitens bleibt unklar, weshalb Geers (2020) keinerlei Verweis auf die Modul Federation macht, die es ermöglicht, Microfrontends auf der Client Side zu implementieren [1] (vgl. Geers 2020: 33 ff., 42 ff., 85 ff.). Dadurch resultiert Verwirrung beim Verständnis der Inhalte, insbesondere wenn Geers (2020) bei der Erklärung von Ansätzen nicht auf spezifische Details wie Skalierbarkeit, Flexibilität und andere Fachbegriffe eingeht. Auch im späteren Verlauf der Publikation behandelt Geers (2020) im vierten Kapitel die Server Side Compoition, was tatsächlich eines der anspruchsvollsten Kapitel darstellt [1] (vgl, Geers 2020: 59 ff.). Mezzalira (2021) erläutert in Kapitel 3 detaillierter, welche Entscheidungen vor Beginn eines Microfrontend-Projekts getroffen werden müssen. Dennoch sind in einigen Abschnitten Schwierigkeiten vorzutreten. Im Anschluss an Kapitel 3 wird eine kurze Erklärung zu den Composition, die zur Implementierung von Microfrontends führen gemacht sowie horizontale und verticale Aufteilungen behandelt [6] (vgl. Mezzalira 2021: 23 ff.). Jedoch wird es im Kapitel 4 erneut verwirrend, wenn mehr Informationen über den Implementierungsansatz von Microfrontends gefordert wird. Im ersten Abschnitt des Kapitels 4 werden zwar erneut horizontale und verticale Aufteilungen behandelt, aber im weiteren Verlauf des Kapitels geht Mezzalira (2021) erneut auf die Architektur der verticalen und horizontalen Aufteilungen ein, was für den Leser Herausforderungen bei dem vollständigen Verständnis und der Interpretation des Themas schafft [6] (vgl. Mezzalira 2021: 39–115). So wird exemplarisch die Architektur der horizontalen Aufteilung in Bezug auf Ansätze auf der Client-, Server- und Edge-Side behandelt. Um jedoch jeden Abschnitt gründlich zu verstehen, fühlte ich die Notwendigkeit umfassender Recherchen zu jedem von Mezzalira geschriebenen Abschnitt,

insbesondere in Bezug auf Server- und Edge-Side Composition.

Eine weitere Herausforderung war, dass Mezzalira (2021) im Kapitel 3 erklärt, dass man für die Implementierung von Microfrontends in der verticalen Aufteilung nur die Client Side Compostion verwenden könne und dass dies die beste geeignete Option sei. Jedoch erklärte Mezzalira (2020) in einem eigenen Blogbeitrag, dass die vertikale Aufteilung auch für die Server side oder sogar die Client side verwendet werden könne [6, 75] (vgl. Mezzalira 2021: o. S., Mezzalira 2020: o. S.).

Für mich war der herausfordernde Teil dieser Abschlussarbeit der Vergleich der Ansätze. Dies lag nicht nur an der begrenzten Verfügbarkeit von Artikeln und vielfältigen Ressourcen, sondern auch an der Notwendigkeit weiterer Forschung, um die Genauigkeit und Glaubwürdigkeit sicherzustellen. Jede der Eigenschaften, obwohl das Konzept und die Gesamtstruktur klar waren, erforderte weitere Untersuchungen, um die Auswirkungen und den Vergleich in verschiedenen Implementierungsansätzen von Microfrontends zu verstehen. Es scheint, dass praktische Arbeiten zu diesen Eigenschaften notwendig sind, um bessere Ergebnisse aus dem Vergleich dieser Eigenschaften zu erhalten.

Wie bereits erwähnt, kann die Implementierung von Microfrontends horizontal oder vertikal erfolgen. In dieser Arbeit liegt der Hauptfokus auf der Implementierung mit der horizontalen Aufteilungsmethode. Für die horizontale Aufteilung kann aus drei vorhandenen Kombinationen gewählt werden, während die vertikale Aufteilung nur auf der Client Side die beste Option zu sein scheint.

In Bezug auf die Implementierungsmethode für Microfrontend, die auf der Client side ausgeführt wird und horizontal aufgeteilt ist, konnte Folgendes festgestellt werden: Die Forschungsergebnisse zeigen, dass die Verwendung von Iframes, obwohl sie eine starke Isolation der Sandbox ermöglichen (was keine der anderen Lösungen im vorherigen Abschnitt bieten kann), auch erhebliche Nachteile mit sich bringt. Wie bereits in Abschnitt 4.1.1 erwähnt, eignet sich die Verwendung von Iframes sehr gut für Desktop-Anwendungen, B2B Anwendungen und Internetanwendungen. Dennoch ist diese Herangehensweise für Projekte, die eine hervorragende Leistung oder starke SEO benötigen, nicht geeignet. Diese Herausforderung kann eine wichtige Rolle bei der Umsetzung von Microfrontend mit Iframes spielen.

Da die Leistung einer Website die Benutzererfahrung beeinträchtigen und den Benutzer dazu veranlassen kann, zu konkurrierenden Websites zu wechseln, ist es von entscheidender Bedeutung. Obwohl Iframes die einfachste Methode zur Herstellung von Verbindungen zwischen Microfrontend sind, müssen auch andere Aspekte berücksichtigt werden, die eine wichtige Rolle im Fortschritt eines Projekts spielen. Wie bereits festgestellt, mag dieser Ansatz möglicherweise nicht für jedes Projekt geeignet sein und hängt stark von den spezifischen Anforderungen des Projekts ab. Fragen, wie ob diese Methode alle unsere Bedürfnisse erfüllt oder nicht, müssen berücksichtigt werden.

Die Forschung zeigt, dass die Verwendung von Iframes in Projekten Vorteile wie starke Isolation, Unterstützung für alle Browser, gute Bereitstellungsmöglichkeiten und angemessene Skalierbarkeit bietet, jedoch auch mit Schwächen konfrontiert ist. Zu den Schwächen, die

meine Recherche bei der Verwendung von Iframes für die Entwicklung von Microfrontends aufzeigt, gehören mangelnde Flexibilität, schlechte Leistung bei der Verwendung mehrerer Iframes gleichzeitig und die Komplexität, die durch die wiederholte Verwendung in mehreren Iframes entsteht.

Da jedoch die Leistung und Flexibilität von Websites an Bedeutung gewonnen haben und die meisten Benutzer eine Vielzahl von Geräten verwenden, um auf Websites zuzugreifen, stellt die Verwendung von Iframes in Projekten für responsive Design Websites eine Herausforderung dar. Darüber hinaus kann die Hauptherausforderung im Bereich End-to-End-Testing und die Komplexität der Testdurchführung ernsthafte Probleme im Entwicklungsprozess verursachen.

Der nächste Ansatz, um Microfrontends auf der Client Side zu implementieren, ist die Module Federation. Durch die Verwendung dieses Ansatzes können Entwickler den Code in kleinere und unabhängige Module aufteilen und sie bei Bedarf laden. Dies ermöglicht es Microfrontends, sich unabhängig voneinander zu entwickeln und zu deployen. Obwohl die Module Federation hauptsächlich bei Microfrontends verwendet wird, kann sie auch in Projekten nützlich sein, in denen mehrere Programme integriert werden müssen oder wenn ein Drittanbietermodul in die Anwendung integriert werden soll. Dieser Ansatz, obwohl er Vorteile für Entwickler mit sich bringt, hat auch erhebliche Nachteile.

Meiner Meinung nach könnte einer der größten Herausforderungen bei der Verwendung von Module Federation zwei Aspekte sein: Erstens hängt die Module Federation stark mit den Tools für den Webpack zusammen, und jede Änderung oder Aktualisierung auf eine neue Version kann signifikante Auswirkungen auf die Anwendung haben. Zweitens könnte es für Entwickler, die mit dem Konzept der Module Federation aufgrund ihrer Neuheit nicht vertraut sind, erforderlich sein, Schulungen und eine tiefere Kenntnis zu erhalten.

Eine weitere Methode, die in der Implementierung von Microfrontends auf der Client Side diskutiert wurde, ist die Verwendung von Web Components. Forschungsergebnisse zeigen, dass dieser Ansatz wie andere Ansätze auch eine akzeptable Skalierbarkeit bietet und mit praktisch allen verfügbaren Frameworks kompatibel ist. Dennoch kann dieser Ansatz für Entwickler in Bezug auf Web Component herausfordernd sein, da Probleme im Zusammenhang mit Browser-Seiten und SEO auftreten können.

Es ist davon auszugehen, dass Unternehmen und Entwicklungsteams nach Benutzerfreundlichkeit suchen, um ihre Websites so einfach wie möglich mit den vorhandenen Tools zu starten. Während das Hinzufügen zusätzlicher Tools die Komplexität nicht zwangsläufig erhöht, erfordert es sicherlich Schulungen. Wenn das Team nicht im Allgemeinen mit den vorhandenen Tools vertraut ist, besteht auch die Möglichkeit von Problemen. Web Componenten bieten Bequemlichkeit in Bereichen wie Bereitstellung, Skalierbarkeit, Testbarkeit und guter Leistung. Aber auch hier kann ein kleiner Fehler große Auswirkungen haben.

Meiner Meinung nach ist dieser Ansatz, obwohl er in Umfragen als die beste Lösung für die Entwicklung und Implementierung von Microfrontends bezeichnet wurde, möglicherweise herausfordernd in Bezug auf die Browser-Seiten, da man nicht absolut sicher sein

kann, dass die Anwendung auf allen Geräten ausgeführt wird. Außerdem muss ich darauf hinweisen, dass beim Einsatz zusätzlicher Tools die Frage aufkommt, ob Aktualisierungen dieser Tools Auswirkungen auf das Projekt haben könnten oder nicht.

In Bezug auf die Server Side Composition behauptete der Mezzalira (2021), dass dieser Ansatz sowohl bei Horizontaler als auch Vertical Split eine Anwendung findet. In der Praxis hat er jedoch nur den horizontalen Split erläutert und sich nicht näher mit der Vertical Split befasst. Dies könnte Fragen aufwerfen wie zum Beispiel, wie dies für die Vertical Split implementiert werden könnte oder ob es zusätzlicher Tools oder weiterer Schulungen bedarf. Obwohl diese Methode eine gute Performance bietet, weist sie Mängel in der Skalierbarkeit auf, besonders wenn versucht wird, Microfrontends in dieser Richtung zu implementieren. Die Skalierbarkeit dieses Ansatzes muss sorgfältig analysiert und überprüft werden, was zusätzlichen Zeitaufwand und Kosten verursachen kann.

Ein Problem, das ich in Bezug auf diesen Ansatz feststellen konnte, ist, warum verschiedene Frameworks dafür entwickelt wurden und ob es bestimmte Schwächen gibt. Dieser Ansatz ist im Vergleich zu anderen Ansätzen, die auf der Client Side ausgeführt werden und eine schwache Leistung und SEO aufweisen, effektiv. Es sollte jedoch beachtet werden, dass die Implementierung dieses Ansatzes äußerst komplex, herausfordernd und anspruchsvoll ist, da viele bewegliche Teile darin vorhanden sind und auch die Deploymentschritte im Programm nicht einfach sind. Dieser Ansatz erfordert eine genauere Verwaltung.

Ein weiterer Ansatz, der die Implementierung von Microfrontends ermöglicht, ist die Edge Side Composition, die mit vielen Herausforderungen verbunden ist und meiner Meinung nach als eine der schwierigste Ansatz für die Implementierung von Microfrontends betrachtet wird. Dies liegt zum einen daran, dass es nur horizontal aufgeteilt werden kann. Zum anderen implementiert jeder CDN-Anbieter es auf eine spezifische Weise, was seine Verwendung erschwert. Auf der anderen Seite zeigen Forschungsergebnisse, dass dieser Ansatz erhebliche Schwächen in Bezug auf Testbarkeit und Leistung aufweist.

Die Forschungsergebnisse zeigen auch, dass die Verwendung der genannten Methoden für die Entwicklung von Microfrontends stark von den Zielen und Anforderungen des Projekts abhängt. Wie bereits erwähnt, wird der Einsatz des IFrames Ansatzes nicht empfohlen, wenn in dem Projekt Leistungsfaktoren und starke SEO vorhanden sind und es besser wäre, sich auf andere Ansätze wie die Server Side Composition zu konzentrieren.

In Bezug auf die Verwendung von Microfrontends durch Unternehmen hat die Forschung gezeigt, dass jedes Unternehmen je nach seinen Zielen und Bedürfnissen einen der erklärten Ansätze gewählt hat. Die Mehrheit von ihnen hat verschiedene Frameworks für die Implementierung der Microfrontend Architektur veröffentlicht. Aber die Frage, die sich hier stellt, lautet, warum diese Frameworks entwickelt wurden und warum die oben genannten Ansätze allein nicht den Anforderungen der betreffenden Unternehmen entsprochen haben.

Es kann argumentiert werden, dass Unternehmen vielleicht versuchen, sich von ihren Konkurrenten abzuheben und durch die von ihnen geschaffenen Frameworks nicht nur ihre Benutzerbasis erweitern, sondern auch in der heutigen Web-Welt eine Aussage machen

wollen. Während des Schreibens dieser Abschlussarbeit war es nicht leicht, Informationen darüber zu erhalten, welche Schritte die Unternehmen unternommen haben, um die Microfrontend Architektur umzusetzen. Nach eingehender Recherche wurde deutlich, dass der Hauptgrund für das Verbergen der Arbeitsschritte sowohl das Vorhandensein von Mitbewerbern als auch der Schutz von Daten ist.

Dennoch kann abschließend gesagt werden, dass vor Beginn eines Projekts mit Microfrontend Architektur alle Aspekte sorgfältig überprüft und die Erfahrungen der Entwickler berücksichtigt werden sollten. Es sollte ebenso herangezogen werden, dass das Projekt, der verwendete Ansatz, die Zukunft sowie alle Vor- und Nachteile bekannt sind.

7 Fazit und Ausblick

7.1 Zusammenfassung der Ergebnisse

Diese Masterarbeit präsentiert eine eingehende Untersuchung und einen Vergleich verschiedener Ansätze zur Implementierung von Microfrontends. Es ist jedoch wichtig zu betonen, dass die Erstellung einer kleinen Anwendung, die von wenigen kleinen Teams entwickelt werden soll, keine bedeutende Herausforderung darstellt. Im Frontend-Bereich der Anwendung sollte eine monolithische Strategie vermieden werden, da Microfrontends eine bessere Skalierbarkeit und Flexibilität bieten. Monolithische Programme sind begrenzt in ihrer Fähigkeit, Änderungen im Laufe der Zeit und mit der Erweiterung der Anwendung vorzunehmen. Zudem erfordert die Zusammenarbeit mehrerer Teams eine erhebliche Komplexität der Anwendung, die ein einzelner Entwickler nicht vollständig erfassen kann.

Die Idee der Microfrontends wurde 2016 eingeführt und nutzt die Grundsätze der Microservices Architektur für den Frontend-Bereich von Anwendungen. Das Ziel besteht darin, die Wartezeit zwischen den Teams zu reduzieren. Die Microfrontend-Architektur verbessert die Herausforderungen der komplexen Anwendungen, insbesondere in Bezug auf Skalierbarkeit, Flexibilität und Wartbarkeit. Durch die Aufteilung von Frontend-Anwendungen in kleinere Einheiten erzielen Entwicklungsteams Vorteile wie schnellere Entwicklung, bessere Bereitstellung, einfachere Skalierbarkeit und effizienteres Programmmanagement. Jedes Microfrontend kann von seinem eigenen Team entwickelt, getestet und bereitgestellt werden, ohne die anderen Teams zu beeinträchtigen. Die vorliegende Untersuchung zeigt, dass Microfrontends sorgfältig geprüft und analysiert werden müssen, bevor sie in einem Projekt implementiert werden. Obwohl diese Architektur zu einer erhöhten Komplexität des Gesamtsystems und zu einer höheren Arbeitsbelastung führt, ist sie eine geeignete Option für mittlere und große Projekte und erleichtert die Skalierbarkeit in solchen Projekten. Die Anwendung dieser Architektur ist in kleinen Teams möglicherweise praktisch nutzlos. Es gibt jedoch alternative Architekturen, die basierend auf den Projektzielen geeignetere Optionen darstellen könnten. Es ist wichtig, eine gründliche Analyse durchzuführen und die Entscheidungen auf fundierten Argumenten zu basieren.

Vor der Implementierung von Microfrontends müssen Entscheidungen getroffen werden, die einen signifikanten Einfluss auf den Erfolg oder Misserfolg des Projekts haben können. Dies umfasst die Implementierung der Microfrontends, verfügbare Composition sowie die Gestaltung von Routing- und Kommunikationsstrategien.

Die Forschungsergebnisse zeigen, dass Microfrontends sowohl vertical als auch horizontal definiert werden können und es drei Ansätze zur Composition gibt: Server Side, Client Side und Edge Side. In der Regel wird das Client Side Routing implementiert, wenn die Client Side betroffen ist, während das Edge Side oder Server Side Routing entsprechend auf der Edge oder der Server Side stattfindet. In der Kommunikation zwischen ihnen sollte normalerweise vermieden werden, direkt miteinander zu kommunizieren. Falls notwendig, können jedoch Ansätze wie die Verwendung von Event Emitters, Custom Events, Query

Strings und Web Storage genutzt werden.

Bei der Unterscheidung zwischen vertical und horizontal Split wird die Client Side Composition als beste Wahl für die vertical Split betrachtet. Im Gegensatz dazu erlaubt die horizontale Aufteilung die Verwendung aller drei Methoden der Client Side-, Server Side-, und Edge Side Composition zur Implementierung von Microfrontends. Es existieren verschiedene Ansätze zur Implementierung von Microfrontends, die je nach Zielen und Art des Projekts variieren können. Eine Möglichkeit besteht darin, Client Side Composition und Techniken wie IFrames, Web Componenten und Module Federation zu verwenden. Die Implementierung von Microfrontends kann sowohl auf der Server Side als auch auf der Edge Side erfolgen.

Es ist wichtig zu beachten, dass jeder der untersuchten Ansätze in dieser Abschlussarbeit seine eigenen Vor- und Nachteile hat. Alle diese Ansätze erleichtern praktisch die Skalierbarkeitsprobleme und tragen zur erfolgreichen Bereitstellung der Anwendung bei. Die Auswahl hängt jedoch von den spezifischen Zielen und Anforderungen des Projekts ab. Wenn SEO oder Performance in Ihrem Projekt eine zentrale Bedeutung haben, ist der Server Side Ansatz möglicherweise die beste Option.

Es existieren keine universelle Lösung für die Implementierung von Microfrontends. Die optimalen Methoden hängen stark von den Kontexten und Anwendungsfällen ab. Entwickler sollten sorgfältig abwägen, welcher Ansatz am besten zu Ihren individuellen Anforderungen passt. Dabei müssen Aspekte wie Skalierbarkeit, Wartbarkeit und Performance berücksichtigt werden.

Es ist klar, dass es schwierig ist, alle Projektziele mit einem Ansatz zu erfüllen. Die Integration neuer Tools für Microfrontends ist jedoch ein vielversprechender Weg, um eine vollständig einsatzbereite Architektur zu schaffen. Dies wird von vielen Experten in der Frontend-Community unterstützt. Microfrontends sind eine innovative Architektur, die noch nicht ausreichend erforscht wurde. Weitere Untersuchungen zu dieser Architektur sind notwendig.

7.2 Ausblick und zukünftige Entwicklungen

Um die verschiedenen Ansätze zur Entwicklung von Microfrontends eingehend zu untersuchen, ist es ratsam, quantitative Forschung in Zusammenarbeit mit Unternehmen durchzuführen. Dadurch kann ein tieferes Verständnis für die praktischen Anwendungsmöglichkeiten und Effektivität der verschiedenen Ansätze gewonnen werden. Um die Auswahl und Implementierung von Microfrontend-Frameworks in realen Projekten zu verbessern, sind zusätzliche Untersuchungen zu spezifischen Frameworks erforderlich, um ihre Anwendungen sowie Vor- und Nachteile besser zu verstehen. Ein vertieftes Wissen über die verschiedenen verfügbaren Frameworks wird dabei helfen, fundierte Entscheidungen zu treffen.

Die Erforschung der Modularität von Ansätzen und der Teamkoordination bei der Verwendung von Microfrontends ist entscheidend, um optimale Entwicklungspraktiken und effiziente Arbeitsabläufe zu fördern. Sicherheitsprobleme im Zusammenhang mit der Verwendung von Microfrontends müssen identifiziert und gelöst werden, um die Sicherheit von

Anwendungen zu gewährleisten. Weitere Untersuchungen und praktische Erfahrungen zur Implementierung von Microfrontends sowie deren Integration mit neuen Tools und Technologien sind notwendig.

Die Erforschung der praktischen Anwendungsmöglichkeiten und Herausforderungen bei der Implementierung in verschiedenen Projektumgebungen wird wertvolle Erkenntnisse liefern. Eine Analyse der wirtschaftlichen und organisatorischen Auswirkungen von Microfrontends in Projekten trägt dazu bei, die potenziellen Kosten- und Effizienzvorteile dieser Architektur besser zu verstehen und fundierte Entscheidungen bei der Auswahl und Implementierung zu treffen. Die Erstellung praxisorientierter Schulungsressourcen und Leitfäden wird Entwicklern und Teams eine klare Richtlinie für die korrekte Nutzung von Microfrontends bieten und den Implementierungsprozess erleichtern.

Literatur

[1] Geers, M. (2020). Micro Frontends in Action. Manning Publications Co. Shelter Island, NY 11964, ISBN 9781617296871, URL: https://www.manning.com/books/micro-frontends-in-action

[2] Wanjala, S. T. (2022). A framework for implementing micro frontend architecture. International Journal of Web Engineering and Technology. Volume: 17, Number: 4, (pp 337 - 352), URL: https://doi.org/10.1504/IJWET.2022.129251

[3] Pavlenko, A., Askarbekuly, N., Megha, S. and Mazzara, M., (2020). Micro-frontends: application of microservices to web front-end, Journal of Internet Services and Information Security (JISIS), Volume: 10, Number: 2, (pp 49-66), DOI: https://doi.org/10.22667/JISIS.2020.05.31.049

[4] Cam, J (2019). Micro Frontends URL: https://www.martinfowler.com/articles/micro-frontends.html . (besucht am 10.03.2024)

[5] Geers, M. (2017). Micro frontends (extending the microservice idea to frontend development). URL: https://micro-frontends.org/ . (besucht am 10.03.2024)

[6] Mezzalira, L. (2021). Building Micro-Frontends: Scaling Teams and Projects Empowering Developers.Published by O'Reilly Media, Inc.,1005 Gravenstein Highway North, Sebastopol, CA 95472, ISBN:9781492082996. URL: https://www.oreilly.com/library/view/building-micro-frontends/9781492082989/

[7] Taibi, D., Mezzalira, L. (2022). Micro-Frontends: Principles, Implementations, and Pitfalls. ACM Sigsoft Software Engineering Notes, Volume: 47 Number: 4, (pp 25 – 29), DOI: https://doi.org/10.1145/3561846.3561853

[8] Technology Radar Thoughtworks. (2020). Micro Frontends. URL: https://www.thoughtworks.com/radar/techniques/micro-frontends . (besucht am 10.03.2024)

[9] Bühler, F., Barzen, J., Harzenetter, L., Leymann, F., Wundrack, P. (2022). Combining the Best of Two Worlds: Microservices and Micro Frontends as Basis for a New Plugin Architecture. In: Barzen, J., Leymann, F., Dustdar, S. (eds) Service-Oriented Computing. SummerSOC 2022. Communications in Computer and Information Science, vol 1603. Springer, Cham. DOI: https://doi.org/10.1007/978-3-031-18304-1_1

[10] Petcu, A., Frunzete, M., Stoichescu, A. D. (2023). BENEFITS, CHALLENGES, AND PERFORMANCE ANALYSIS OF A SCALABLE WEB ARCHITECTURE BASED ON MICRO-FRONTENDS.(pp 319 – 334). https://www.scientificbulletin.upb.ro/rev_docs_arhiva/reze1d_965048.pdf

[11] Peltonen, S., Mezzalira, L., Taibi, D. (2021). Motivations, benefits, and issues for adopting Micro Frontends: A Multivocal Literature Review. Information and Software Technology, Volume: 136, Number: 106571, (pp 01 – 18). DOI: https://doi.org/10.1016/j.infsof.2021.106571

[12] Stefanovska, E., Trajkovik, V. (2022). Evaluating Micro Frontend Approaches for Code Reusability. In: Zdravkova, K., Basnarkov, L. (eds) ICT Innovations 2022. Reshaping the Future Towards a New Normal. ICT Innovations 2022. Communications in Computer and Information Science, vol 1740. Springer, Cham , 93-106. DOI: https://doi.org/10.1007/978-3-031-22792-9_8

[13] Yang, C., Liu, C., Su, Z. (2019).Research and Application of Micro Frontends. In IOP Conference Series: Materials Science and Engineering, Volume: 490, Number: 6 , (pp 1-7). DOI: https://doi.org/10.1088/1757-899x/490/6/062082

[14] Mena, M., Corral, A., Iribarne, L., Criado, J. (2019).A Progressive Web Application Based on Microservices Combining Geospatial Data and the Internet of Things. IEEE Access. Volume: 7. (pp 104577–104590). URL: https://doi.org/10.1109/access.2019.2932196

[15] Wang, D., Yang, D., Zhou, H., Wang, Y., Hong, D., Dong, Q., Song, S. (2020).A novel application of educational management information system based on micro frontends.Procedia Computer Science, 176, (pp 1567–1576). DOI: https://doi.org/10.1016/j.procs.2020.09.168

[16] React , The library for web and native user interfaces, URL: https://react.dev/ . (besucht am 10.03.2024)

[17] https://angular.io/ . (besucht am 10.03.2024)

[18] https://vuejs.org/ . (besucht am 10.03.2024)

[19] https://svelte.dev/ . (besucht am 10.03.2024)

[20] single-spa , A javascript router for front-end microservices, URL: https://single-spa.js.org/ . (besucht am 10.03.2024)

[21] P. Huang: 2018, Mooa framework, A micro-frontend Framework for Angular from single-spa on. URL: https://github.com/phodal/mooa . (besucht am 10.03.2024)

[22] IBM . What is Three-Tier Architecture. URL: https://www.ibm.com/topics/three-tier-architecture . (besucht am 10.03.2024)

[23] O'Reilly . Three-tier client-server architecture in Architectural Patterns by Pethuru Raj, Anupama Raman, Harihara Subramanian . URL : https://www.oreilly.com/library/view/architectural-patterns/9781787287495/df0e98f1-0190-42e2-a9b1-69f050a03a4e.xhtml . (besucht am 10.03.2024)

[24] Mohamed, K. , Wijesekera, D. (2012). Performance analysis of web services on mobile devices. Procedia Computer Science, 10, 744–751. DOI: https://doi.org/10.1016/j.procs.2012.06.095

[25] What is the Difference Between SOAP and REST? . Amazon Web Services, Inc URL: https://aws.amazon.com/compare/the-difference-between-soap-rest/. (besucht am 10.03.2024)

[26] Kornienko, D. V., S. V. Mishina, and M. O. Melnikov. (2021).The single page application architecture when developing secure web services . Journal of physics, 2091(1), 012065. DOI: https://doi.org/10.1088/1742-6596/2091/1/012065

[27] What is an API (Application Programming Interface)?. URL: https://aws.amazon.com/what-is/api/ . (besucht am 10.03.2024)

[28] Fielding RT (2000) Architectural styles and the design of network-based software architectures.. Dissertation, University of California, Irvine, CA , URL: https://ics.uci.edu/~fielding/pubs/dissertation/top.htm . (besucht am 10.03.2024)

[29] Sunyaev, A. (2020). Web Services. In: Internet Computing. Springer, Cham. https://doi.org/10.1007/978-3-030-34957-8_6

[30] Gillis, A. S. (2021). REST API (RESTful API). ComputerWeekly.de. URL: https://www.computerweekly.com/de/definition/RESTful-API . (besucht am 10.03.2024)

[31] About gRPC URL: https://grpc.io/about/ . (besucht am 10.03.2024)

[32] IONOS editorial team. (2020). gRPC – paving the way to the client-server communication of the future . IONOS Digital Guide. URL: https://www.ionos.com/digitalguide/server/know-how/an-introduction-to-grpc/ . (besucht am 10.03.2024)

[33] Introduction to gRPC . URL: https://grpc.io/docs/what-is-grpc/introduction/ . (besucht am 10.03.2024)

[34] What is gRPC: Main Concepts, Pros and Cons, Use Cases https://www.altexsoft.com/blog/what-is-grpc/ . (besucht am 10.03.2024)

[35] De Lauretis, L. (2019). From Monolithic Architecture to Microservices Architecture. IEEE International Symposium on Software Reliability Engineering Workshops (ISSREW). Berlin, Germany. (pp. 93-96). DOI: https://doi.org/10.1109/issrew.2019.0005

[36] Harris, C. Microservices vs. monolithic architecture. URL: https://www.atlassian.com/microservices/microservices-architecture/microservices-vs-monolith . (besucht am 10.03.2024)

[37] Kalske, M., Mäkitalo, N., Mikkonen, T. (2018). Challenges When Moving from Monolith to Microservice Architecture. In: Garrigós, I., Wimmer, M. (eds) Current Trends in Web Engineering. ICWE 2017. Lecture Notes in Computer Science(), Volume: 10544, (pp 32-47), Springer, Cham. https://doi.org/10.1007/978-3-319-74433-9_3

[38] Benavente, V., Yantas, L., Moscol, I., Rodríguez, C. R., Inquilla, R., Pomachagua, Y. (2022). Comparative analysis of microservices and monolithic architecture. 2022 14th International Conference on Computational Intelligence and Communication Networks (CICN). Al-Khobar, Saudi Arabia. (pp 177-184). DOI: https://doi.org/10.1109/cicn56167.2022.10008275

[39] Ren, Z., Wang, W., Wu, G., Gao, C., Chen, W., Wei, J., Huang, T. (2018). Migrating Web Applications from Monolithic Structure to Microservices Architecture. In Proceedings of the 10th Asia-Pacific Symposium on Internetware (Internetware '18). Association for Computing Machinery, New York, NY, USA, Article 7, (pp 1–10). DOI: https://doi.org/10.1145/3275219.3275230

[40] Ahmad, A:(2023) .Monolithic vs. Service-Oriented vs. Microservice Architecture: Top Architectural Design Patterns. Design Gurus: One-Stop Portal For Tech Interviews. URL: https://www.designgurus.io/blog/Monolithic-Service-Oriented-Microservice-Architecture . (besucht am 10.03.2024)

[41] Dragoni, N., Giallorenzo, S., Lafuente, A. L., Mazzara, M., Montesi, F., Mustafin, R. (2017).Microservices: yesterday, today, and tomorrow. In Springer eBooks (S. 195–216). DOI : https://doi.org/10.1007/978-3-319-67425-4_12

[42] Blinowski, G. J., Ojdowska, A., Przybyłek, A. (2022). Monolithic vs. Microservice Architecture: A Performance and Scalability Evaluation. IEEE Access, Volume: 10, (pp 20357–20374). DOI: https://doi.org/10.1109/access.2022.3152803

[43] Abgaz, Y. M., McCarren, A., Eklund, P., Solan, D., Lapuz, N., Bivol, M., Jackson, G., Yılmaz, M., Buckley, J., Clarke, P. M. (2023).Decomposition of Monolith Applications Into Microservices Architectures: A Systematic Review. IEEE Transactions on Software Engineering. Volume:49, Number: 8, (pp 4213–4242). DOI: https://doi.org/10.1109/tse.2023.32872

[44] M. Villamizar et al. (2015),Evaluating the monolithic and the microservice architecture pattern to deploy web applications in the cloud,2015 10th Computing Colombian Conference (10CCC), Bogota, Colombia, 2015, pp. 583-590, DOI: https://doi.org/10.1109/ColumbianCC.2015.7333476

[45] Avenga Team (2022) , Microservices vs. monoliths: which architecture will be best for your product? URL : https://www.avenga.com/magazine/microservices-vs-monoliths/?region=de . (besucht am 10.03.2024)

[46] Beznos, M (2023), Microservices vs monolith: Which architecture is the best choice for your business? . URL : https://www.n-ix.com/microservices-vs-monolith-which-architecture-best-choice-your-business/ . (besucht am 10.03.2024)

[47] Ozkaya, M (2023), When to use Monolithic Architecture . Published in Design Microservices Architecture with Patterns , Principles - Medium . URL : https://medium.com/design-microservices-architecture-with-patterns/when-to-use-monolithic-architecture-57c0653e245e . (besucht am 10.03.2024)

[48] Sarwar, A (2020), Monolithic Architecture: What, Why and When . Published in The Startup - Medium . URL : https://medium.com/swlh/

monolithic-architecture-what-why-and-when-986dc5d5ce03 . (besucht am 10.03.2024)

[49] Davis, A. (2023). Monolithic vs Microservices Architecture: Pros, Cons and Which to Choose. URL: https://www.openlegacy.com/blog/monolithic-application . (besucht am 10.03.2024)

[50] Lewis, J., Fowler, M. (2014). Microservices (a definition of this new architectural term). URL: https://martinfowler.com/articles/microservices.html . (besucht am 10.03.2024)

[51] Richardson, C. (2015). Introduction to Microservices. URL: https://www.nginx.com/blog/introduction-to-microservices . (besucht am 10.03.2024)

[52] Kharenko, A. (2015). Monolithic vs. Microservices Architecture. Published in Microservices Practitioner Articles. URL: https://articles.microservices.com/monolithic-vs-microservices-architecture-5c4848858f59 . (besucht am 10.03.2024)

[53] Wang, A., Tonse, S. (2013). Announcing ribbon: tying the netflix mid-tier services together. URL: http://techblog.netflix.com/2013/01/announcing-ribbon-tying-netflix-mid.html . (besucht am 10.03.2024)

[54] Ponce, F. A., Márquez, G., Astudillo, H. (2019).Migrating from monolithic architecture to microservices: A Rapid Review.38th International Conference of the Chilean Computer Science Society (SCCC).Concepcion, Chile. (pp. 1-7). DOI: https://doi.org/10.1109/sccc49216.2019.8966423

[55] Nunes, L., Santos, N., Rito Silva, A. (2019). (pp 37-52) From a Monolith to a Microservices Architecture: An Approach Based on Transactional Contexts, In: Bures, T., Duchien, L., Inverardi, P. (eds) Software Architecture. ECSA 2019. Lecture Notes in Computer Science(), vol 11681. Springer, Cham. DOI: https://doi.org/10.1007/978-3-030-29983-5_3

[56] Haq, S. (2018). Introduction to Monolithic Architecture and MicroServices Architecture. URL: https://medium.com/koderlabs/introduction-to-monolithic-architecture-and-microservices-architecture-b211a5955c63 . (besucht am 10.03.2024)

[57] Kaur, A. (2021). What is Single page application (SPA)? Pros and cons with examples. Insights - Web and Mobile Development Services and Solutions. URL: https://www.netsolutions.com/insights/single-page-application/ . (besucht am 10.03.2024)

[58] Ly, A. (2022). Pros and Cons between Single Page and Multi-Page apps. Medium. URL: https://andrewly.medium.com/pros-and-cons-between-single-page-and-multi-page-apps-8f4b26acd9c9 . (besucht am 10.03.2024)

[59] Ghosh, A. (2023). Single page application vs multi page application. Ellow Talent. URL: https://ellow.io/single-page-application-vs-multi-page-application/ . (besucht am 10.03.2024)

[60] MDN Web Docs Resources for Developers,by Developers, URL: https://developer.mozilla.org/en-US/ . (besucht am 10.03.2024)

[61] SPA (Single-page Application) - MDN Web Docs Glossary: Definitions of web-related terms , URL: https://developer.mozilla.org/en-US/docs/Glossary/SPA . (besucht am 10.03.2024)

[62] Ryte Wiki. (o. D.). AJAX. URL: https://de.ryte.com/wiki/AJAX . (besucht am 10.03.2024)

[63] Akil, S. (2021). Single page application. URL: https://t2.sa/en/blog/Single-Page-Application . (besucht am 10.03.2024)

[64] Digital Vitarag. (2023). 8 Best Single-Page Application Frameworks for Web App Development. Medium. URL: https://medium.com/@vitarag/8-best-single-page-application-frameworks-for-web-app-development-eb81b035c545 . (besucht am 10.03.2024)

[65] Chiradeep BasuMallick (2023) , What Is a Framework? Definition, Types, Examples, and Importance . Spiceworks. URL: https://www.spiceworks.com/tech/tech-general/articles/what-is-framework/ . (besucht am 10.03.2024)

[66] React Developer Tools | REACT. URL: https://react.dev/learn/react-developer-tools . (besucht am 10.03.2024)

[67] DevTools Overview | Angular. URL: https://angular.io/guide/devtools . (besucht am 10.03.2024)

[68] Installation | Vue DevTools. URL: https://devtools.vuejs.org/guide/installation.html . (besucht am 10.03.2024)

[69] Ağaç, R. (2023). Microfrontend architecture: Advantages and disadvantages. Medium. URL: https://rasidagac.medium.com/micro-frontend-architecture-advantages-and-disadvantages-acf45e20c6c9 . (besucht am 10.03.2024)

[70] Shpak, A. (2023). Micro Frontend architecture: What, why and how to use it. Euristiq. URL: https://euristiq.com/micro-frontend-architecture/ . (besucht am 10.03.2024)

[71] Brooks, G. (2023). Micro Frontends: pros and cons. Published in Fabrity Software House. URL: https://fabrity.com/blog/micro-frontends-pros-and-cons/ . (besucht am 10.03.2024)

[72] Saring, J. (2020) . 11 Micro Frontends Frameworks You Should Know Published in ITNEXT . URL : https://itnext.io/

11-micro-frontends-frameworks-you-should-know-b66913b9cd20 . (besucht am 10.03.2024)

[73] Luigi Framework. URL : https://luigi-project.io/ . (besucht am 10.03.2024)

[74] Piral URL: https://piral.io/ . (besucht am 10.03.2024)

[75] Mezzalira, L. (2020). Micro-frontends in context. URL : https://increment.com/frontend/micro-frontends-in-context/ . (besucht am 10.03.2024)

[76] Mezzalira, L. (2019),Micro-frontends decisions framework . URL : https://lucamezzalira.medium.com/micro-frontends-decisions-framework-ebcd22256513 . (besucht am 10.03.2024)

[77] W3Schools. HTML <iframe> Tag . (besucht am 10.03.2024)

[78] Valdes, A. (2023). Microfrontends using React: The complete guide. ClickIT. URL: URL: https://www.clickittech.com/developer/microfrontends/amp/. (besucht am 10.03.2024)

[79] Jaeyow. (2022). Micro-frontends building blocks: Webpack Module Federation. URL : https://fullstackdeveloper.tips/microfrontend-building-blocks-module-federatio . (besucht am 10.03.2024)

[80] Jackson, Z. (2020). Webpack 5 Module Federation: A game-changer in JavaScript architecture. URL: https://medium.com/swlh/webpack-5-module-federation-a-game-changer-to-javascript-architecture-bcdd30e02669 . (besucht am 10.03.2024)

[81] ElHousieny, R. (2021). Microfrontends with Module Federation: What, Why,and How. URL: https://levelup.gitconnected.com/microfrontends-with-module-federation-what-why-and-how-845f06020ee1 . (besucht am 10.03.2024)

[82] Webpack Module Federation. URL: https://webpack.js.org/concepts/module-federation/ . (besucht am 10.03.2024)

[83] Pros and Cons of Module Federation: A Comprehensive Overview. URL: https://module-federation.io/docs/en/mf-docs/0.2/pros-cons/ . (besucht am 10.03.2024)

[84] Lakshman, P. (2022). Module Federation using Webpack 5. The Micro-frontend journey. Published in Walmart Global Tech Blog. URL : https://medium.com/walmartglobaltech/module-federation-using-webpack-5-the-micro-frontend-journey-719688c5d73b . (besucht am 10.03.2024)

[85] Web Component - Introduction . URL: https://www.webcomponents.org/introduction . (besucht am 10.03.2024)

[86] Stencil.js. URL: https://stenciljs.com/ . (besucht am 10.03.2024)

[87] Ionic Framewok . URL: https://ionicframework.com/ . (besucht am 10.03.2024)

[88] Stencil- FAQ. URL: https://stenciljs.com/docs/faq . (besucht am 10.03.2024)

[89] Can I use . URL: https://caniuse.com/ . (besucht am 10.03.2024)

[90] Custom Elements (V1) | Can I use. support tables for HTML5, CSS3, etc. URL: https://caniuse.com/custom-elementsv1 . (besucht am 10.03.2024)

[91] OpenSourcee. (2023). Companies already using micro frontends and why.DEV Community. URL: https://dev.to/opensourcee/companies-already-using-micro-frontends-and-why-o37 . (besucht am 10.03.2024)

[92] Mezzalira, L (2019a), Adopting a Micro-frontends architecture, URL: https://medium.com/dazn-tech/adopting-a-micro-frontends-architecture-e283e6a3c4f3 . (besucht am 10.03.2024)

[93] Was ist Netflix? . Hilfe-Center URL: https://help.netflix.com/de/node/412 . (besucht am 10.03.2024)

[94] Lattice JS | Celestial Systems URL: https://latticejs.com/ . (besucht am 10.03.2024)

[95] Possumato, M et al. (2021). How we build micro frontends with Lattice - Netflix TechBlog . Medium. URL: https://netflixtechblog.com/how-we-build-micro-frontends-with-lattice-22b8635f77ea . (besucht am 10.03.2024)

[96] Thinksys. (2023). Embracing Scalability with Micro frontend Architecture in 2023 . Micro Frontend Architecture: Complete Guide 2023. ThinkSys Inc. URL: https://thinksys.com/development/micro-frontend-architecture/ . (besucht am 10.03.2024)

[97] Kotte, G. N. (2023). History of IKEA.com: Static files and Microfrontends . Published in Flat Pack Tech . Medium . URL : https://medium.com/flat-pack-tech/history-of-ikea-com-static-files-and-microfrontends-6def9d7c4285 . (besucht am 10.03.2024)

[98] Amazon.Two-Pizza Teams,

URL : https://docs.aws.amazon.com/whitepapers/latest/introduction-devops-aws/two-pizza-teams.html . (besucht am 10.03.2024)

[99] Prajwal, Y., Parekh, J.V. and Shettar, R., (2021). A brief review of micro-frontends . United International Journal for Research and Technology, Volume: 2, Issue: 8, (pp 123-127), URL: https://uijrt.com/articles/v2/i8/UIJRTV2I80017.pdf

[100] FasterCapital , https://fastercapital.com/de/keyword/web-frameworks.html. (besucht am 10.03.2024)