

Hochschule Flensburg

MASTER–THESIS

Thema: Integrated Approach to an Algorithm-based Application determining the Use of Blockchain Technology

von: Piet Carstensen

Matrikel-Nr.: 700240

Studiengang: Angewandte Informatik

Betreuer/in und
Erstbewerter/in: Dipl.-Math. Jochen Stamp

Zweitbewerter/in: Dr. Parissa Sadeghi

Ausgabedatum: 24.06.2022

Abgabedatum: 24.11.2022

Selbständigkeitserklärung

Ich versichere, dass ich die vorliegende Thesis ohne fremde Hilfe selbstständig verfasst und nur die angegebenen Quellen benutzt habe.

Inhaltsverzeichnis

1 Einleitung	4
1.1 Zielsetzung	5
1.2 Aufgabenstellung	5
2 Grundlagen	6
2.1 Aufbau einer Web-Anwendung	6
2.1.1 Architektur	8
2.1.2 Technologien	9
2.2 Blockchain	11
2.2.1 Distributed-Ledger-Technology	11
2.2.2 Hash-Funktionen	12
2.2.3 Permissionless vs. Permissioned Blockchain	13
2.2.4 Transaktionen	14
2.2.5 Konsens Protokoll	15
2.2.6 Vergleich Blockchain-Typen und Zentrale Datenbank	18
2.2.7 Blockchain Alternativen	19
3 Analyse	22
3.1 Anforderungen	22
3.1.1 Funktionale Anforderungen	22
3.1.2 Nichtfunktionale Anforderungen:	23
4 Konzeptionierung	25
4.1 Datenbank	25
4.1.1 Datenbankschema	25
4.2 Komponenten	26

INHALTSVERZEICHNIS

4.2.1	Blockchain-Vergleich	27
4.2.2	Blockchain-Empfehlung	30
4.3	Algorithmus	30
4.3.1	Programmatische Umsetzung	30
5	Implementierung und Ergebnisse	34
5.1	Entwicklungsumgebung	34
5.2	Git	34
5.3	Maven	35
5.4	Docker	36
5.5	Frontend	39
5.5.1	Technologien	39
5.5.2	Architektur	40
5.5.3	Benutzeroberfläche	42
5.6	Backend	47
5.6.1	Technologien	47
5.6.2	Architektur	48
5.6.3	REST-API Endpunkte	49
6	Zusammenfassung und Ausblick	52
6.1	Zusammenfassung	52
6.2	Ausblick	53
6.2.1	Technische Erweiterungen	53
6.2.2	Framework als Grundgerüst für weitere Projekte	55
7	Glossar	56
	Abbildungsverzeichnis	58
	Quellcodeverzeichnis	60
8	Anhang	66

1 — Einleitung

Die Blockchain ist eine der Innovationen, die sowohl im akademischen Sektor als auch industriellen Sektor immer mehr Aufmerksamkeit erlangt [55]. Es ist damit neben der Künstlichen Intelligenz (KI) eine der vielversprechenden Technologieentwicklungen der letzten Jahre, um z.B. Geschäftsprozesse zu verbessern [46]. Alleine im Jahr 2018 haben etwa 90% der amerikanischen und europäischen Banken und Finanzinstitute damit begonnen, die Einführung der Blockchain-Technologie zu prüfen. Die Marktgröße (Marktkapitalisierung) der Blockchain-Technologie soll Berichten zufolge bis zum Jahr 2023 23,3 Milliarden US-Dollar und bis zum Jahr 2025 39,7 Milliarden US-Dollar erreichen [22]. Zum Vergleich, Apple hat eine Marktkapitalisierung von 2.901 Milliarden US-Dollar [52]. Das rasante Wachstum der Blockchain erklärt sich auch durch die steigende Nachfrage nach dieser Technologie in nahezu allen Branchen. Dies geht von den Bereichen Finanzdienstleistungen über Konsum- und Industriegüter bis hin zu Medien, Telekommunikation, Verkehr, Gesundheitswesen und dem öffentlichen Dienst [55] [22].

Der Begriff «Blockchain» ist in seinem Ursprung eine Bezeichnung für die Art und Weise, Daten zu strukturieren und zu speichern [38]. Durch das im Jahr 2008 veröffentlichte Whitepaper: **Bitcoin: A Peer-to-Peer Electronic Cash System** bekam der Begriff Blockchain eine neue Bedeutung [45]; Bitcoin ist die erste Kryptowährung, die auf der Blockchain-Technologie basiert. Durch Bitcoin wurde ein Peer-to-Peer-Zahlungssystem erschaffen, das Online-Zahlungsabwicklungen ohne den Bedarf einer Drittpartei ermöglicht [45]. Drittparteien, auch Intermediäre genannt, sind zum Beispiel Banken oder Kreditkartenunternehmen und werden hinzugezogen, um die Integrität der Daten zu gewährleisten oder in diesem Fall eine Transaktion zu bestätigen [59].

Durch den «Blockchain-Hype» entstand ein großes Interesse, das Potenzial der Blockchain-Technologie in eigenen Projekten zu nutzen. Dass die Verwendung nur in bestimmten Fällen und unter bestimmten Voraussetzungen möglich ist, ist vielen potenziellen Nut-

zern nicht klar. Das liegt an der Komplexität der Blockchain.

Ein Artikel aus dem Jahr 2017 über die Evolution der Blockchain-Technologie anhand von Daten der Plattform GitHub zeigt ähnliche Erkenntnisse. GitHub ist eine Plattform, auf der Entwickler Quellcode speichern und verwalten können. Ein Projekt kann privat oder öffentlich zugänglich sein. Eine Vielzahl an Blockchain-Projekten befindet sich auf dieser Plattform. Eine Suche nach Projekten von Privatpersonen, Organisationen, Unternehmen und Institutionen, die die Blockchain-Technologie verwenden, ergibt im Jahr 2022 auf der GitHub-Seite 149.948 Treffer. 2017 waren es lediglich 86.000 Treffer, von denen 92% nicht aktiv weiter entwickelt wurden. Ein Projekt hatte eine durchschnittliche Lebensdauer von 1,22 Jahren [16].

Nicht jeder Anwendungsfall ist für die Blockchain-Technologie geeignet und nicht jede Blockchain passt zu dem Anwendungsfall des Nutzers. Das bedeutet, es sollte zunächst entschieden werden, ob die Verwendung einer Blockchain überhaupt möglich ist und unmittelbar darauf, welche Blockchain geeignet ist. Im laufenden Jahr 2022 existieren etwa 1000 verschiedene Blockchains und 20000 verschiedene aktive Kryptowährungen [40].

1.1 Zielsetzung

Ziel dieser Arbeit ist, die Konzeption und Realisierung einer ersten Version einer Anwendung zur Empfehlung der Blockchain-Technologie abhängig vom Anwendungsfall des Nutzers. Die Komplexität der Blockchain-Technologie hat zur Folge, dass viele sog. «Fachleute» kaum etwas von dieser Technologie verstehen und dessen Potenzial nicht erkennen und nicht anwenden können. Für diesen Fall soll das zu entwickelnde Tool Abhilfe schaffen und dem Nutzer helfen, die Frage auf die Machbarkeit und Notwendigkeit einer Blockchain in seinem Projekt, zu beantworten.

1.2 Aufgabenstellung

Im Rahmen dieser Arbeit sollen Funktionen und Methoden konzipiert und implementiert werden, die integrale Bestandteile einer Anwendung sind. Unabhängig vom Kenntnisstand des Nutzers sollen einfache Fragen bei der Entscheidungsfindung bzgl. der Anwendbarkeit der Blockchain-Technologie helfen. Außerdem sollen für den Fall der Nichtanwendbarkeit der Blockchain-Technologie alternative Vorschläge gemacht werden.

2 — Grundlagen

2.1 Aufbau einer Web-Anwendung

Das World Wide Web hat sich im letzten Jahrzehnt von einem System, das statische Seiten, sog. Websites liefert, zu einem System, das verteilte und komplexe Anwendungen, sog. Web-Anwendungen (Web-Apps) bereitstellt, gewandelt. Die zunehmende Popularität von Web-Anwendungen kann auf mehrere Faktoren zurückgeführt werden; darunter der Remote-Zugriff, die plattformübergreifende Kompatibilität und die schnelle Entwicklung [39]. Die Web-Plattform ist ein komplexes Ökosystem bestehend aus einer großen Anzahl von Komponenten und Technologien, einschließlich HTTP-Protokoll, Webserver, serverseitigen Anwendungsentwicklungstechnologien (z.B. Angular, React, PHP, ...) und Webbrowswer. Die beliebteste Variante ist eine Web-App, die auf einer Client-Server-Architektur basiert. Client ist in diesem Kontext das Programm, welches für «den Zugang zur Web-App» benötigt wird und Server steht für «das Hosten der Web-App»; das heißt, es ist der Ort, an dem das Programm abgelegt wird [62]. Web-Apps benutzen 3 Haupt-Programmiersprachen.

1. HTML – Für das Erstellen von Seiten
2. CSS – Für das Erstellen von Design-LAYOUTS
3. JavaScript – Zum Erzeugen von Logik und dynamischen und interaktiven Content

Die Kombination aus diesen drei Sprachen ergibt das Grundskelett einer Web-App.

Web-Entwickler benutzen verschiedene JavaScript-Frameworks und Bibliotheken. Die beliebtesten sind Angular, React und Vue.js [42]. Im direkten Vergleich zu anderen Technologien, wie z.B. einer mobilen App oder einer Desktop-App, bietet eine Web-App in wesentlichen Bereichen folgende offensichtliche Vorteile [36].

1. Cross-Platform Im Jahr 2022 besitzt fast jedes mobile oder stationäre Endgerät einen Browser und Zugang zum Internet. Das Nutzen von Web-Apps ist überall und mit jedem Gerät möglich, sofern eine Verbindung zum Internet besteht. Die Nutzung eines mobilen oder eines Desktop-Endgeräts macht für die Verwendung der Web-App keinen Unterschied. Die Entwickler von Web-Apps müssen nicht zwei Versionen (Mobil und Desktop) einer Anwendung implementieren. Es muss lediglich das «Styling», also der CSS-Code, für beide Geräte angepasst werden [36].

2. Kosteneffizienz Die Entwicklung einer mobilen App ist überflüssig, da unabhängig vom Endgerät über einen Browser der Zugang zur Anwendung hergestellt wird. Es gibt also keinen signifikanten Unterschied bei der Nutzung einer Web-App, egal für welches Betriebssystem (Android oder IOS) der Nutzer sich auch entschieden hat. Dadurch können Kosten, die für das Entwickeln, Hosten und Deployen einer mobilen App-Variante anfallen, gespart werden [36].

3. Skalierbarkeit Die Web-App kann ohne zusätzlicher Installation eines Programms verwendet werden. Wird der Browser-Tab der Web-App geschlossen, ist das Programm beendet und es existieren keine Dateien des Programms auf dem PC oder Smartphone. Web-Server können Out-Of-Box einen hohen Datenverkehr im Netzwerk behandeln, was das Skalieren der Web-App vereinfacht und die Fehleranfälligkeit verringert [36].

4. Datenspeicherung Angezeigte und erzeugte Daten einer Web-App werden in den meisten Fällen mithilfe einer Datenbank gespeichert. Die Datenbank kann durch einen Cloud-Server z.B. die Google Cloud-Plattform ausgelagert werden. Das Behandeln von Risiken im Bereich Datenverlust wird dadurch an Experten-Teams von Google Cloud abgegeben. Die Ausfälle des Servers werden ebenfalls von der Plattform selbst verwaltet. Viele Web-Frameworks besitzen direkte Anbindungen an diese Cloud-Provider (*SDKs*), wodurch die Verbindung von Cloud und Web-App vereinfacht und gut dokumentiert ist [36].

5. Security Plattformen wie z.B. Google Cloud bieten Möglichkeiten den Web-Server extern zu hosten. Diese externen Server werden von Experten-Teams und -Unternehmen überwacht und verwaltet und sind in der Regel vor typischen Angriffen, wie einem DDoS-Angriff geschützt [36].

2.1.1 Architektur

Eine Web-App-Architektur muss nicht nur effektiv sein, sondern auch mit Zuverlässigkeit, Skalierbarkeit, Sicherheit und Robustheit umgehen können [27]. Web-Apps werden in Frontend und Backend unterteilt. Für die Zusammensetzung aus Frontend und Backend (Architektur) gibt es verschiedene Patterns [17].

Single Page Applications (SPAs) Unter Single Page Application (SPA) versteht man eine Web-App, dessen Inhalt nur einmal geladen und dann dynamisch aktualisiert wird. Für Interaktionen wird der Server nicht zwangsläufig angesprochen. Es bietet eine ähnliche User-Experience wie eine native mobile App. Diese Art von Pattern muss in einigen Fällen damit umgehen können, keine Internetverbindung zu haben bzw. offline zu arbeiten [49].

Server-Side Rendered Applications (SSRs) SSR nennt man das serverseitige Rendern einer Javascript-Framework-Website zu HTML und CSS. Die Berechnung des Renderns wird auf den Server ausgelagert und bleibt dem Client erspart. Wenn eine SSR-App erstellt wird, stellt der Server alle Daten zusammen und stellt bei jeder Anfrage ein neues HTML-Dokument bereit. Sobald der Browser das CSS erhält, kann er die Benutzeroberfläche zeichnen und es ist nicht erforderlich, auf das Laden von JavaScript zu warten. Auf diese Weise wird die Seite schneller geladen [17].

Microservices Microservices gehören zum Typ der Service-Oriented Architekturen (SOAs). Ein Microservice ist ein *lightweight* Service mit einer einzigen Funktionalität. Diese Architektur gilt als effizient und dynamisch. Es kann von Microservice zu Microservice ein anderer Technologie-Stack (Techstack) verwendet werden, sodass Entwickler für Funktionalitäten nicht an ihren *Techstack* gebunden sind [17].

Serverless Architecture Dem Namen nach könnte man bei dieser Architektur erwarten, dass auf den Einsatz von Servern verzichtet wird. Das ist aber nicht der Fall. Serverless bedeutet, dass die Konfigurations- und Administrations-Management-Software nicht benötigt wird. Die gesamte Infrastruktur wird ausgelagert und von einem Drittanbieter bereitgestellt. Entwickler können Programmcode deployen, während ein Cloud-Anbieter Server für die Ausführung ihrer Anwendungen, Datenbanken und Speichersysteme in beliebigem Umfang bereitstellt [17] [15].

2.1.2 Technologien

Zur serverseitigen Umsetzung der Web-App stehen diverse Frameworks in Java zur Auswahl. Innerhalb dieser Arbeit wird das Framework Spring betrachtet, da dieses seit geheimer Zeit marktführend ist und sich besonders gut für diesen Anwendungsfall eignet [53].

Spring

Das Framework Spring vereinfacht das Erstellen von Java-Enterprise-Anwendungen. Es besteht aus vielen unterschiedlichen Modulen, die in einer Art Bibliothek je nach Bedarf und Anwendungsfall eingefügt werden können. Es deckt eine Menge an Anwendungsszenarien ab [51]. Das Framework ist Open-Source und wird von einer aktiven Community unterstützt, die Module in die Spring-Bibliothek hinzufügen [51]. Ein Kernfeature ist *Dependency Injection*, dass es Objekten erlaubt, ihre eigenen Abhängigkeiten (engl. dependencies) zu definieren, die dann vom Spring-Container in sie injiziert werden [51]. Entwickler können dadurch modulare Anwendungen erstellen, die aus losen gekoppelten Komponenten bestehen, die Grundvoraussetzungen für Microservices und verteilte Netzwerkanwendungen bilden. Spring bietet auch integrierte Unterstützung für typische Aufgaben, die eine Anwendung ausführen muss, z.B. Datenbindung, Typkonvertierung, Validierung, Ausnahmebehandlung, Ressourcen- und Ereignisverwaltung und Internationalisierung [51].

Spring-Boot ist ein Tool zur schnelleren und einfacheren Entwicklung von Web-Apps und Microservices [31]. Spring-Boot verwendet einen Ansatz zum Hinzufügen und Konfigurieren von Start-Dependencies, basierend auf den Anforderungen des Projektes. Nach eigenem Ermessen wählt Spring-Boot aus, welche Pakete installiert und welche Standardwerte verwendet werden sollen, anstatt dass diese Entscheidungen vom Nutzer selbst getroffen werden und alles manuell eingerichtet werden muss [31]. Im Anwendungsfall dieser Master-Thesis soll Spring den RESTful-Web-Service (REST-API) darstellen, der dem Frontend eine Schnittstelle für Daten anbietet und gleichzeitig Daten, die aus dem Frontend kommen, in eine Datenbank schreibt.

Angular

Angular ist ein Open-Source JavaScript-Framework, das von Google gepflegt wird [42]. Es umfasst ein komponentenbasiertes Framework für die Erstellung skalierbarer Web-

Anwendungen, eine Sammlung von Bibliotheken, die eine Vielzahl von Funktionen abdecken (darunter Routing, Formularverwaltung, Client-Server-Kommunikation) und eine Reihe von Entwickler-Tools, die beim Entwickeln, Erstellen, Testen und Aktualisieren des Codes unterstützen. Das Angular-Ökosystem besteht aus einer Gruppe von über 1,7 Millionen Entwicklern, Bibliotheksautoren und Autoren von Inhalten [26]. Im Jahr 2022 zählt es zu den beliebtesten Frontend-Frameworks und wird vom Unternehmen Google entwickelt [54]. Angular-Anwendungen können in JavaScript und TypeScript entwickelt werden. Angular Projekte können unterschiedlich strukturiert werden. Ein Ansatz ist die Unterteilung in verschiedene Module.

Components Components (Komponenten) sind die Module, die ein Nutzer sieht, z.B. Navigation-Seiten. Viele Components haben ähnliche Features oder Funktionen, sodass sogenannte «shared components» angewendet werden können. Diese sind generisch konzipiert, sodass sie von verschiedenen Modulen benutzt werden können.

Services Services sind die Schnittstelle zwischen Backend und den angezeigten Daten im Frontend. Werden Daten aus einer Datenbank abgefragt, befinden sich die Funktionen in Service-Dateien. Auch das Mappen von Datenbankentitäten auf Klassen gehört zu den Aufgaben eines Service.

Models Unter Models (Business Modelle) sind die Klassen des Projektes gelistet. Dabei handelt es sich um TypeScript-Interfaces, die als Objekte von anderen Funktionen verwendet werden können. Jedes angefragte Objekt aus der Datenbank wird zu einem Interface konvertiert (mapping), sodass diese auch im Frontend spezifischer behandelt werden können, anstatt sie im JSON-Format zu lassen. Der Codeausschnitt 2.1 zeigt ein Interface Beispiel für Alerts.

Codeausschnitt 2.1: TypeScript Interface für Alerts

```
export interface Alert {
    message?: MESSAGES;
    type?: TYPES;
    details?: String;
}

export enum MESSAGES {
    SUCCESS_SUBMIT = "Successfully submitted survey",
    ERROR_ANSWERS = "Error with validation of answers",
}
```

```
ERROR_SURVEY_NOT_FOUND = "Sorry, Survey not found.",
NO_BLOCKCHAIN_RECOMMENDATION = "Dont use a Blockchain Technologies for your
                                project.",
BLOCKCHAIN_RECOMMENDATION = "Blockchain Technologies could be useful for
                                your project.",
}

export enum TYPES {
  DANGER = "alert-danger",
  SUCCESS = "alert-success",
  WARNING = "alert-warning",
}
```

MongoDB

MongoDB ist eine Open-Source NoSQL-Datenbank, die auf dem dokumentenorientierten Modell basiert und mit JavaScript und JSON arbeitet [30]. MongoDB speichert Daten in flexiblen, JSON-ähnlichen Dokumenten, d.h. Felder können von Dokument zu Dokument unterschiedlich sein, und die Datenstruktur kann im Laufe der Zeit geändert werden [43].

2.2 Blockchain

2.2.1 Distributed-Ledger-Technology

Distributed-Ledger-Technology, kurz DLT, beschreibt eine Datenstruktur, die sich über mehrere Computer erstreckt und geografisch über viele Standorte verteilt ist. Die populärste Form und eine Untergruppe der DLT ist die Blockchain. Diese Form einer Datenstruktur beinhaltet eine permanente Chronik von Transaktionen. Bei Blockchains handelt es sich um dezentrale Datenbanken, die chronologische Abfolgen von Datenblöcken enthalten. Ein Datenblock kann eine oder mehrere Transaktionen enthalten und wird über kryptografische Methoden mit anderen Blöcken verbunden bzw. verkettet. Das Vertrauen in das Netzwerk der Blockchain wird durch folgende vier Charakteristika dieser Technologie ermöglicht. [9].

Logbuch Ein Logbuch (engl. ledger) ist eine Kollektion an Transaktionen. In diesem kann jede Transaktion nachvollzogen werden, da es die Historie an Transaktio-

nen abbildet. Jeder Nutzer kann eine eigene Kopie des Logbuchs pflegen. Wenn neue vollständige Knoten (engl. nodes) dem Blockchain-Netzwerk beitreten, suchen sie nach anderen vollständigen Knoten und fordern eine vollständige Kopie des Logbuchs an. Dieses erschwert den Verlust oder die Zerstörung des Logbuchs.

Sicherheit Blockchains werden durch kryptografische Verfahren gesichert. Es ist nach heutigen Stand sichergestellt, dass die Daten im Protokoll nicht manipuliert wurden und dass diese Daten belegbar sind.

Teilung Das Protokoll wird je nach Blockchain an mehrere Teilnehmer verteilt. Das erzeugt eine nachvollziehbare Transparenz über alle Nodes hinweg.

Verteilung Durch die Erhöhung der Anzahl der Nodes wird die Möglichkeit für einen bösartigen Akteur, das von der Blockchain verwendete Konsens Protokoll (engl. consensus protocol) zu beeinflussen, verringert.

In einem Netzwerk, in dem jeder Teilnehmer anonym beitreten darf, ermöglicht die Blockchain ein Level-of-Trust über alle Teilnehmer, ohne dass sich die Teilnehmer untereinander vertrauen müssen. Für ein Netzwerk, das den Zugang strenger kontrolliert und bei dem ein gewisses Vertrauen zwischen Nutzern bestehen kann, tragen die Funktionen der Blockchain dazu bei, das Vertrauen zu stärken. Das erste Netzwerk wird als permissionless und das zweite Netzwerk als permissioned Blockchain bezeichnet (Vgl. zu diesem Absatz: [9]).

2.2.2 Hash-Funktionen

Eine Hash-Funktion ist wie ein deterministischer Fingerabdruck von willkürlichen Daten. Ein Hash ist einfach zu berechnen, aber schwer zurückzurechnen. Die Eingabe für die Hash-Funktion hat eine beliebige Länge, die Ausgabe hat immer eine feste Länge. Die von einer Hash-Funktion zurückgegebenen Werte werden als *Message Digest* oder einfach als Hash-Werte bezeichnet. Folgende Abbildung veranschaulicht eine Hash-Funktion. In dieser wird der Text `Hello World!` in die Hash-Funktion gegeben. Alternativ könnten auch Dateien in die Funktion gegeben werden.

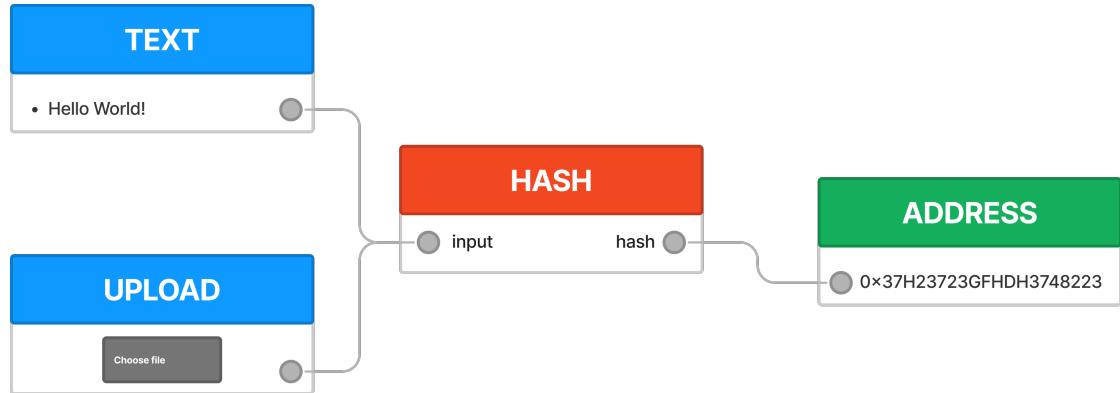


Abbildung 2.1: Hash-Funktion auf der Ethereum Blockchain

Hash-Funktionen kommen in fast allen Anwendungen der IT-Security vor; auch im Bereich der Blockchain. In dieser Domäne werden Hashes zum Beispiel als Identifikationsnummer (ID) verwendet [9]).

2.2.3 Permissionless vs. Permissioned Blockchain

Bei der Unterscheidung zwischen öffentlicher (engl. permissionless) und zugangsbeschränkter (engl. permissioned) Blockchain geht es um die Teilnahme-Beschränkungen bzw. die Rollenverteilung innerhalb des Netzwerkes. Bei permissionless Blockchains kann ein öffentliches Netzwerk genutzt werden, auf welches jeder frei zugreifen kann (d.h. Transaktionen lesen, schreiben und validieren). Jeder kann Blöcke veröffentlichen, ohne die Berechtigung einer Behörde (engl. authority) zu benötigen. Meistens sind permissionless Blockchains Open-Source-Softwares, die jedem frei zur Verfügung stehen, der diese herunterladen möchte. Da auch böswillige Nutzer Blöcke veröffentlichen können, nutzt dieser Typ der Blockchain ein Multiparty-Agreement oder auch Consensus Model (Konsens Protokoll) genannt. Dieses verlangt vom Nutzer Ressourcen aufzuwenden, um einen Block zu veröffentlichen. In einer permissionless Blockchain wird ein Nutzer, der einen Block nicht böswillig und nach Vorschrift, veröffentlicht, mit einer systemeigenen Kryptowährung belohnt. In einem permissioned Blockchain-Netzwerk muss ein Nutzer, der einen Block veröffentlicht, von einer Behörde (zentral oder dezentral) autorisiert werden. Nur autorisierte Nutzer dürfen die Blockchain warten. Es kann festgelegt werden, wer Lesezugriff hat und beschränkt werden, wer Transaktionen ausstellen darf. Permissioned Blockchains werden meist von Open-Source oder Closed-Source-Softwares instanziert und gewartet. Für das Veröffentlichen von Blöcken werden auch Konsens Protokolle

benutzt, die allerdings nicht den gleichen Aufwand wie die permissionless Blockchains benötigen. Aus diesem Grund ist mit diesem Typ Blockchains eine verbesserte Datenverarbeitung möglich [60]. In vielen Fällen werden permissioned Blockchains von Organisationen benutzt, die mehr Kontrolle und Schutz der Blockchain benötigen oder sich nicht vollständig vertrauen. Abhängig davon können Konsens Protokolle gewählt werden, je nachdem wie viel Parteien einander vertrauen. Der Teilnehmerkreis kann offen oder beschränkt sein. Dies wiederum beschreibt die Ausprägung public oder private.

Public vs. Private Die Eigenschaft private bezeichnet ein zusätzliches Rollen- und Rechtekonzept im Blockchain-Netzwerk. Bei den public Blockchains gibt es keine Zugangsbeschränkungen und jeder Teilnehmer, der mitmachen möchte, kann dies tun. Beispiele für diese Art der Blockchains sind Bitcoin und Ethereum.

Private Blockchains sind nur für einen vorab definierten, beschränkten Teilnehmerkreis zugänglich. Die Transaktionen werden nicht von den Teilnehmern direkt gesteuert, sondern indirekt über einen Intermediär. Dieser Intermediär dient zugleich als zentrale Legitimation-Stelle, die das gesamte System kontrolliert.

2.2.4 Transaktionen

Transaktionen sind kryptografisch signierte Anweisungen von einem Konto. Die einfachste Transaktion ist die Überweisung eines Betrages einer Währung von einem Konto auf ein anderes. Der Inhalt einer Transaktion kann bei jeder Blockchain verschieden sein. Eine Transaktion bezieht sich auf eine Aktion, die durch ein externes Konto ausgelöst wird, d.h. ein Konto, das von einem Menschen verwaltet wird. Wenn Bob beispielsweise Alice 1 Bitcoin schickt, muss Bobs Konto belastet werden und Alices Konto muss eine Gutschrift erhalten. Diese Zustands ändernde Aktion findet innerhalb einer Transaktion statt. Die Informationen, die mitgesendet werden, können die Adresse (Kennung) des Absenders, den öffentlichen Schlüssel des Absenders, eine digitale Signatur, die Eingabe und Ausgabe der Transaktion enthalten. Unabhängig davon, wie die Daten gebildet und verarbeitet werden, ist die Feststellung der Gültigkeit und Authentizität einer Transaktion wichtig. Typischerweise ist die Transaktion mit dem zugehörigen privaten Schlüssel des Absenders digital signiert und kann zu jeder Zeit mit dem zugehörigen öffentlichen Schlüssel überprüft werden. Folgende Abbildung zeigt, wie eine Transaktion auf der Ethereum Blockchain aussehen könnte [9].

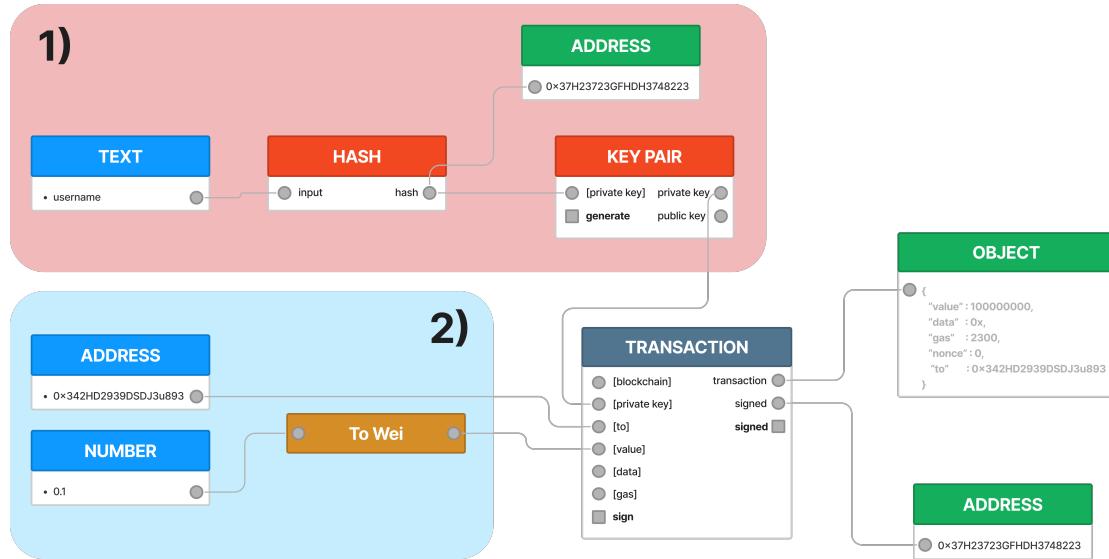


Abbildung 2.2: Transaktionszusammensetzung auf der Ethereum Blockchain

Im oberen-linken Bereich (1) wird der Nutzernname des Absenders der Transaktion in eine Hash-Funktion gegeben. Damit wird ein privater Schlüssel generiert. Diese eindeutige Adresse wird in der Transaktion hinterlegt. Im unteren-linken Bereich (2) wird die Adresse des Empfängers (Hash) in der Transaktion hinterlegt und der Wert der Transaktion festgelegt. In diesem Beispiel werden 0,1\$ in die Ethereum eigene Währung «Wei» umgewandelt und in der Transaktion unter value abgespeichert. Eine Transaktion muss nach Eingabe aller benötigten Daten signiert werden und hat deshalb eine Sign-Methode. Der Prozess des Signierens von Transaktionen beinhaltet eine mathematische Funktion, die sowohl von der Nachricht (den Transaktionsdetails, hier data) als auch von Ihrem privaten Schlüssel abhängt. Das Ergebnis ist eine Signatur, die mit ihrem öffentlichen Schlüssel und der Nachricht (den Transaktionsdaten) von Anderen überprüft werden kann. In öffentlichen Blockchains ist diese Signatur der Transaktion für jeden sichtbar. Die Parameter gas, gas price und nonce können erstmal als Black Box gesehen werden und sind im Beispiel der Ethereum-Blockchain von Bedeutung.

2.2.5 Konsens Protokoll

Beim Eintritt in ein Blockchain-Netzwerk stimmt jeder Teilnehmer den initialen Bedingungen des Systems zu. Die Bedingungen werden im Genesis-Block aufgezeichnet. Jeder weitere Block muss auf Grundlage des vereinbarten Konsens Protokolls der Block-

chain nach dem Genesis-Block hinzugefügt werden. In den Bedingungen des Konsens Protokolls befinden sich z.B. Entscheidungen auf die Frage, welcher Nutzer als Nächstes einen Block veröffentlicht. Im Grunde dient das Protokoll dazu, einer Gruppe von sich gegenseitig misstrauenden Benutzern die Zusammenarbeit zu ermöglichen.

Das Konsens Protokoll übernimmt die Einstimmigkeit über folgende Eigenschaften.

- Der Anfangszustand des Systems wurde vereinbart (Genesis-Block)
- Die Nutzer stimmen dem Konsens Protokoll zu, nachdem die Blöcke dem System hinzugefügt werden
- Jeder Block ist mit dem vorherigen Block verknüpft

Es gibt eine Reihe von Konsens Protokollen. Die populärsten werden im Folgenden vorgestellt.

Proof of Work Beim Proof of Work (PoW)-Protokoll, veröffentlicht ein Full Node den nächsten Block, welcher als Erstes eine rechenintensive Aufgabe löst. Diese Aufgabe gilt als Barriere für das Veröffentlichen von Blöcken. Die Lösung dieser Aufgabe ist der «Proof» (Beweis), dass sie Arbeit geleistet haben. Das Lösen dieser Aufgabe kann beliebig schwer gestaltet werden, die Lösung ist allerdings sehr einfach zu validieren. Ein Nachteil dieses Protokolls ist, dass das Lösen der Aufgaben Energie und Hardware intensiv ist. Das Protokoll wird in permissionless Blockchains eingesetzt.

Proof of Stake Das Proof of Stake (PoS)-Protokoll basiert auf der Idee, dass je mehr ein Nutzer in das System investiert hat, desto wahrscheinlicher ist es, dass er den Erfolg des Systems wünscht und desto unwahrscheinlicher ist es, dass er böswillig handelt. Der Einsatz ist oft ein bestimmter Betrag an Kryptowährung, den der Nutzer des Blockchain-Netzwerks in das System investiert hat. Sobald der Einsatz geleistet wurde, kann die Kryptowährung im Allgemeinen nicht mehr ausgegeben werden. Proof-of-Stake-Blockchain-Netzwerke verwenden die Höhe des Einsatzes eines Nutzers als entscheidenden Faktor für die Veröffentlichung neuer Blöcke. Die Wahrscheinlichkeit, dass ein Nutzer des Blockchain-Netzwerks einen neuen Block veröffentlicht, hängt also vom Verhältnis seines Einsatzes zum Gesamtbetrag der gesetzten Kryptowährung im Blockchain-Netzwerk ab. PoS-Protokolle werden in permissionless Blockchain-Netzwerken eingesetzt. Anders als beim PoW-Protokoll muss keine rechenintensive Aufgabe gelöst werden. Einige Blockchain-Netzwerke verzichten auf eine Belohnung für die Erstellung

von Blöcken; diese Systeme sind so konzipiert, dass die gesamte Kryptowährung bereits unter den Nutzern verteilt ist und nicht ständig neue Kryptowährung erzeugt wird. Es existieren mehrere Arten von Protokollen, basierend auf dem PoS.

- chain-based proof of stake
- byzantine fault tolerance (BFT) proof of stake
- coin age proof of stake
- nothing at stake

Round Robin Im Round Robin Konsens Protokoll wechseln sich Nodes bei der Erstellung von Blöcken ab. Das Protokoll stellt sicher, dass ein alleiniger Knoten keine Mehrheit von Blöcken erstellt. Es hat den Vorteil, dass es einfach zu handhaben ist, keine kryptografischen Rätsel aufgibt und wenig Energie benötigt. In den permissionless Blockchain-Netzwerken, die von den meisten Kryptowährungen verwendet werden, funktioniert dieses Protokoll nicht gut und wird daher in permissioned Blockchains eingesetzt.

Proof of Authority/ Proof of Identity Das Proof of Authority (PoA)-Protokoll stützt sich auf das teilweise Vertrauen der Full Nodes durch ihre bekannte Verbindung zu realen Identitäten. Full Nodes müssen ihre Identität bestätigen und sich innerhalb des Netzwerks verifizieren. Die Idee ist, dass Full Nodes ihre Identität/Reputation einsetzen, um neue Blöcke zu veröffentlichen. Die Nutzer des Blockchain-Netzwerks beeinflussen die Reputation eines Full Nodes direkt, basierend auf dem Verhalten des Full Nodes. Full Nodes können ihren Ruf verlieren, wenn sie in einer Weise handeln, mit der die Nutzer des Blockchain-Netzwerks nicht einverstanden sind, ebenso wie sie an Ruf gewinnen können, wenn sie in einer Weise handeln, mit der die Nutzer des Blockchain-Netzwerks einverstanden sind. Je niedriger die Reputation, desto geringer ist die Wahrscheinlichkeit, einen Block veröffentlichen zu können. Daher ist es im Interesse eines Full Nodes, eine hohe Reputation zu erhalten. Dieses System funktioniert nur in permissioned Blockchain-Netzwerken mit einem hohen Level an Vertrauen [9].

2.2.6 Vergleich Blockchain-Typen und Zentrale Datenbank

TABLE			
	Permissionless Blockchain	Permissioned Blockchain	Zentrale Datenbank
Anzahl der Transaktionen	Gering	Hoch	Sehr Hoch
Latenz	Langsam	Mittel	Schnell
Anzahl der Reader	Hoch	Hoch	Hoch
Anzahl der Writer	Hoch	Gering	Hoch
Konsens Modell	Proof of Work, Proof of Stake	BFT Protokolle	Keine
Zentrale Verwaltung	Nein	Ja	Ja

Abbildung 2.3: Vergleich Blockchain-Typen und Zentrale Datenbank [23]

In folgender Tabelle werden einige Parameter einer permissionless und permissioned Blockchain gegenüber einer traditionellen zentralen Datenbank verglichen. Der Transaktionsdurchsatz (engl. throughput) beschreibt, wie viele Aktionen in einer bestimmten Zeit ausgeführt werden. Im Blockchain-Bereich bezieht sich der Transaktionsdurchsatz darauf, wie schnell eine Blockchain Transaktionen verarbeitet, was üblicherweise in Transaktionen pro Sekunde (TPS) ausgedrückt wird, aber auch in Minuten (TPM) oder Stunden (TPH) angegeben werden kann [11]. Unter Latenz (engl. latency) versteht man in der Datenverarbeitung gewöhnlich die Verzögerung zwischen einer Eingabe und der empfangenen Ausgabe. Bei Kryptowährungen kann sich die Latenz auf zwei verschiedene Verzögerungen beziehen. Die erste ist die Latenz im Netzwerk einer Blockchain und die zweite ist die Latenz an einer Börse. Die Latenzzeit des Blockchain-Netzwerks ist die Zeit zwischen dem Einreichen einer Transaktion an ein Netzwerk und der ersten Bestätigung der Annahme durch das Netzwerk [7]. Die Anzahl der Leser (engl. Reader) und Schreiber (engl. Writer) in einem Blockchain-Netzwerk können durch den Typen einer Blockchain

variieren. Reader und Writer beziehen sich auf einen Node und dessen Rechte im Netzwerk. Das Konsens Protokoll einer Blockchain beinhaltet die Regeln, die jeder Teilnehmer akzeptiert und einhalten muss. Bei privaten oder zentralisierten Blockchains existiert regelmäßig eine zentrale Instanz (zentrale Verwaltung) oder zumindest eine beschränkte Anzahl von Teilnehmern. Den angeschlossenen Netzknoten werden dann unterschiedliche Rechte zugewiesen, und nur eingeladene Teilnehmer können die Transaktionen sehen [21].

2.2.7 Blockchain Alternativen

Es existieren neben der Blockchain andere Technologien, die ähnliche Funktionalitäten aufweisen und unter Blockchain-Alternativen gelistet werden können.

Centralized Databases Eine zentralisierte Datenbank ist eine Datenbank, die sich an einem einzigen Ort befindet und dort gespeichert und gepflegt wird. Diese übernehmen die Verwaltung von Daten in einem hochgradig optimierten Aufzeichnungssystems [35]. Da alle Daten nur an einem einzigen Ort gespeichert werden, ist es einfacher, auf die Daten zuzugreifen und sie zu koordinieren. Da jedoch eine zentrale Behörde alle Daten verwaltet, kann es vorkommen, dass bei einem Systemausfall des zentralisierten Datenbanksystems die tatsächlichen Daten zerstört werden. Während dezentrale Blockchains ein Problem in der Skalierbarkeit aufweisen, zeigen Beispiele wie VISA eine bessere Skalierbarkeit pro Transaktion. Beim Vergleichen der Transaktionsgeschwindigkeit von Bitcoin und VISA ist ein gewaltiger Unterschied festzustellen. Das Bitcoin-Netzwerk kann 4,6 Transaktionen pro Sekunde durchführen, während Visa über 1700 Transaktionen pro Sekunde abwickeln kann. Das bedeutet, dass es 150 Millionen Transaktionen an einem Tag durchführen kann. Andere Blockchains wie z.B. Near werben mit bis zu 100.000 Transaktionen pro Sekunde, verzichten dafür allerdings auf Dezentralisierung und Sicherheit [35]. Dieses Problem wird *Blockchain Trilemma* genannt.

Das Blockchain-Trilemma ist ein Konflikt bei Distributed-Ledger-Technologien, die ein Gleichgewicht zwischen Geschwindigkeit, Sicherheit und Dezentralisierung herstellen müssen. Das bedeutet, dass diese drei Kategorien ohne Kompromisse/Abwägungen nicht bestmöglich optimiert werden können. Am Beispiel Bitcoin ist zu erkennen, dass diese Blockchain auf eine starke Sicherheit durch ihr Konsens Protokoll und die Dezentralisierung, setzt, allerdings dafür die Geschwindigkeit (z.B. Transaktionen pro Sekunde) vernachlässigt [20].

Centralized Ledgers Ein zentralisiertes Hauptbuchsystem nach Tommy Koenen and Erik Poll [35], 2018 (engl. centralized ledger) ist ein System, bei dem eine Zusammenstellung aller Transaktionen von einer einzigen Behörde kontrolliert oder bearbeitet wird, d.h. es wird von einer einzigen Kontrollstelle gesteuert. Es erfasst Daten in einem zentralen Ledger, das auch als Hauptbuch (engl. general ledger) bezeichnet wird. Ein öffentliches Hauptbuch ist nichts anderes als eine Sammlung aller Konten. Lange Zeit wurde ein zentrales Hauptbuch verwendet, um die gesamte Buchhaltung zu verwalten, einschließlich der Aufzeichnung der Finanztransaktionen des Unternehmens mit verschiedenen Unternehmen für Finanzanalysen, Steuerberichte und mehr. Der Nachteil dieses Ansatzes ist, dass er zwar effizient ist, aber das Hauptbuch ist anfällig für Fehler, die von einer zentralen Behörde absichtlich oder unabsichtlich gemacht werden [35].

Distributed Databases Die auf verteilten Ledgern basierende Technologie ist eine neuere Entwicklung von Ledgern, mit der versucht wird, den Buchführungsprozess zu dezentralisieren und eine Fehlerquelle zu beseitigen, indem die zentrale Behörde abgeschafft wird. Die Blockchain Bitcoin ist eines der erfolgreichsten Beispiele für dezentralisierte Hauptbücher. Bei einer verteilten Datenbank handelt es sich um eine Datenbank, die nicht auf einen Computer oder ein Netzwerk beschränkt ist, sondern über mehrere Computer oder Netzwerke verteilt ist, sodass es den Anschein hat, dass es sich um eine einzige Datenbank handelt. Ihre physischen Komponenten befinden sich an verschiedenen Orten, ohne dass sie sich physische Komponenten teilen. Das Hauptziel verteilter Datenbanken besteht darin, effiziente und kostengünstige Dienste anzubieten, die eine hohe Skalierbarkeit aufweisen. Das Hauptziel verteilter Datenbanken besteht darin, effiziente und kostengünstige Dienste mit hoher Skalierbarkeit anzubieten. Dabei ist vor allem zu bedenken, dass eine verteilte Datenbank nicht nur keine Datenredundanz aufweist, sondern auch die Daten- und Netzsicherheit gewährleisten muss. Es besteht die Möglichkeit des Datendiebstahls und des Missbrauchs von Netzwerkressourcen [35].

Cloud Storage Blockchains speichern Daten in Speichersystemen über verschiedene Knotenpunkte hinweg. Allerdings können die Datenmengen in Unternehmen mit der Zeit erheblich wachsen und Probleme beim Replizieren über mehrere Geräte hinweg verursachen. Stattdessen können Unternehmen hochleistungsfähige und äußerst stabile Cloud-Speicherdiensste nutzen, die in hochwertigen Rechenzentren gehostet werden. Cloud-Speicherdaten werden durch eine Hash-Funktion in mehrere verschlüsselte Segmente unterteilt. Die Segmente dieser gesicherten Netzwerke sind über das Netzwerk

verteilt, wobei sich jedes Segment an einem dezentralen Ort befindet. Cloud-Speicher verfügen über robuste Sicherheitsmerkmale wie Transaktions-Ledger, Verschlüsselung mit öffentlichen/privaten Schlüsseln und Hash-Blöcke. Diese Sicherheitsmerkmale gewährleisten eine zuverlässige und starke Verteidigung gegen Angreifer [35].

Decentralized Storage Das dezentrale System (engl. decentralized Storage) kann mit einem Peer-to-Peer-Netzwerk verglichen werden, in dem ein Netzwerk von Nutzern einen Teil der Gesamtdaten speichert und nicht ein zentraler Server, der von einem einzigen Unternehmen oder einer Organisation betrieben wird. Dadurch entstehen sichere, belastbare und skalierbare Speichersysteme. Je nach Netzwerk kann es sich um dezentrale Blockchain-basierte Anwendungen oder Peer-to-Peer-Netzwerke handeln. Blockchain ist eines der am weitesten verbreitetsten digitalen Ledger. Es gibt jedoch auch andere Produkte der Distributed-Ledger-Technologie, die sich als Alternativen zur Blockchain erweisen, wenn es um die gemeinsame Nutzung und Speicherung von Daten geht [35].

Alternative Distributed Ledger Technologies Im Gegensatz zu privaten Blockchains, die in manchen Fällen eine Genehmigung für bestimmte Operationen erfordern, kann eine Anwendung mit dem DAG-Ansatz (Directed Acyclic Graphs) schnell Daten schreiben. Dennoch dauert es eine Weile, bis die Transaktion bestätigt wird. DAGs haben einige große Vorteile gegenüber privaten Blockchains, da sie Daten schnell schreiben können. Allerdings dauert es viel länger, bis eine Transaktion bestätigt wird, als bei der privaten Blockchain, die für bestimmte Vorgänge eine Genehmigung benötigt. Um diese Probleme zu lösen, müssen die Anwendungen die Benutzer benachrichtigen, wenn Konflikte auftreten, und das Protokoll enthält Regeln, die dabei helfen. Nachfolgend werden drei der anderen bestehenden Alternativen zur Blockchain genannt [35]:

- Iota Tangle
- Hashgraph
- R3 Corda

3 — Analyse

In diesem Kapitel werden die Anforderungen an die zu erstellende Software festgelegt sowie eine Bewertung der zu verwendenden Technologien, unter Betrachtung der vorher definierten Anforderungen durchgeführt.

3.1 Anforderungen

Innerhalb dieses Unterkapitels werden die Anforderungen an die Anwendung, die innerhalb dieser Arbeit erstellt werden sollen, beschrieben. Diese unterteilen sich in funktionale und nicht funktionale Anforderungen.

3.1.1 Funktionale Anforderungen

Die funktionalen Anforderungen werden im Folgenden als kurze User Storys formuliert. User Storys beschreiben Anforderungen in Form von «real-world» Beispielen [66]. Häufig werden User Storys in einem agilen Softwareentwicklungsprozess eingesetzt, um anstehende Aufgaben oder Funktionen aus Sicht des Benutzers zu beschreiben [66].

01 - Ich möchte Antworten zu Blockchain-Fragen geben können Der Kunde bekommt eine Reihe von Fragen über seinen Anwendungsfall gestellt und kann diese durch Antwortmöglichkeiten und/ oder freien Eingaben beantworten.

02 - Ich möchte eine Blockchain-Typ-Empfehlung auf Basis der gegebenen Antworten bekommen Der Kunde bekommt nach erfolgreicher Eingabe aller Antworten eine Seite angezeigt, auf der eine Empfehlung des Blockchain-Typs (permissionless oder permissioned und private oder public) abgebildet ist.

03 - Ich möchte eine Blockchain-Anbieter-Empfehlung auf Basis der gegebenen Antworten bekommen Der Kunde bekommt nach erfolgreicher Beantwortung eines zusätzlichen Fragenkatalogs eine Blockchain-Anbieter-Empfehlung (Bitcoin vs. Ethereum vs. Andere).

04 - Die Ergebnisse möchte ich in einer überschaubaren/ vergleichbaren Übersicht abrufen können In einer Übersicht kann sich der Kunde die Empfehlungen anschauen. Es werden sowohl Vor- und Nachteile als auch Gemeinsamkeiten und Unterschiede für Blockchain-Typs und Blockchain-Anbieters aufgelistet.

05 - Die Ergebnisse möchte ich mir auch zu einem späteren Zeitpunkt wieder anschauen können Eine erstellte Empfehlung und die Antworten des Kunden werden gespeichert und können bei Bedarf erneut angefragt werden.

06 - Ich möchte das Framework sowohl an meinen Desktop-PC, als auch auf meinem Smartphone bedienen können Der Kunde hat die Möglichkeit dasselbe Programm auf Desktop und Smartphone zu nutzen. Es gibt keine funktionellen Unterschiede zwischen den Plattformen.

3.1.2 Nichtfunktionale Anforderungen:

Die nichtfunktionalen Anforderungen sind Qualitätskriterien, die an die Software gestellt werden. Dabei leiten sich diese teilweise aus den funktionalen Anforderungen ab. Die erhobenen, nichtfunktionalen Anforderungen werden im Folgenden beschrieben.

Benutzerschnittstelle: Das User-Interface (UI) sollte ansprechend und intuitiv gestaltet werden. Dabei sollen gängige Standards der UI-Gestaltung eingehalten und berücksichtigt werden.

Datenbank: Die Empfehlungen und Antworten eines Nutzers sollen in einer Datenbank gespeichert werden.

Plattformunabhängigkeit: Das Framework wird als Web-App realisiert, dadurch ist eine generelle Plattformunabhängigkeit gegeben. Das Tool sollte allerdings für diverse Browser optimiert werden. Aus diesem Grund ist es notwendig, mindestens die gängigsten Browser zu unterstützen. Dazu zählen:

KAPITEL 3. ANALYSE

- Chrome (ebenfalls Chrome basierte Browser wie Brave und Chromium)
- Safari
- Firefox
- Internet Explorer

Stabilität: Fehler, die zur Laufzeit der Applikation auftauchen, müssen über geeignete Fehlermeldungen dem Nutzer angezeigt und mit Anweisungen versehen werden.

4 — Konzeptionierung

Unter Berücksichtigung der im vorangegangenen Kapitel erarbeiteten Anforderungen, wird innerhalb dieses Kapitels die Applikation beschrieben. Zunächst wird auf die Auswahl der Technologien eingegangen, bevor das Datenbankschema hergeleitet wird. Im weiteren Verlauf des Kapitels werden die unterschiedlichen Komponenten der Web-App und die Entscheidungsfindung, einer zu empfehlenden Blockchain entworfen.

4.1 Datenbank

Aus den Anforderungen an das Framework geht hervor, dass Informationen in einer Datenbank gespeichert werden sollen. Dafür muss zunächst ein geeignetes Datenbankschema konzipiert werden, das die Objekte bzw. Dokumente der MongoDB-Datenbank beschreibt.

4.1.1 Datenbankschema

Abbildung 4.1 zeigt das Datenbankschema des Frameworks. Die Fragen (Question), die der Nutzer gestellt bekommt, werden in der Datenbank abgelegt. Die Fragen bestehen aus einem Index, der angibt, in welcher Reihenfolge die Fragen gestellt werden, einem Text, der die eigentliche Frage beinhaltet und Details, die die Frage nochmals spezifizieren. Der Fragetyp (QuestionType) beschreibt, ob die Frage eine allgemeine (General) oder Blockchain spezifische Frage ist. Im Array der Antwortmöglichkeiten (Answerchoice) werden die dem Nutzer angezeigten Optionen gespeichert, aus denen der Nutzer eine Option pro Frage auswählen kann. Die Answerchoice besteht aus einem Text und einem Attribut `Next`. Next legt die nächste Frage fest, beschreibt also den nächsten Index einer Frage. Das letzte Attribut einer Frage ist ein Array aus Alternativen (Alternative). Jede Antwort (Answer) auf eine Frage wird im Objekt Answer gespeichert.

Eine Antwort besteht aus dem Index der Frage, der ebenfalls wie die ID eine eindeutige Identifikationsmöglichkeit ist und der getroffenen Auswahl an Antwortmöglichkeiten. Ist eine Umfrage beendet, wird ein Umfrageobjekt (Survey) angelegt. Dort wird die Uhrzeit und ein Array mit den Antworten der Umfrage gespeichert. Eine Empfehlung (Recommendation) besteht aus der Survey-ID, dem empfohlenen Blockchain-Typ und einem Array aus empfohlenen Blockchains. Blockchains werden ebenfalls in der Datenbank gespeichert und bestehen aus unterschiedlichen Parametern. Abbildung 4.1 zeigt das Datenbank-Schema des Frameworks.

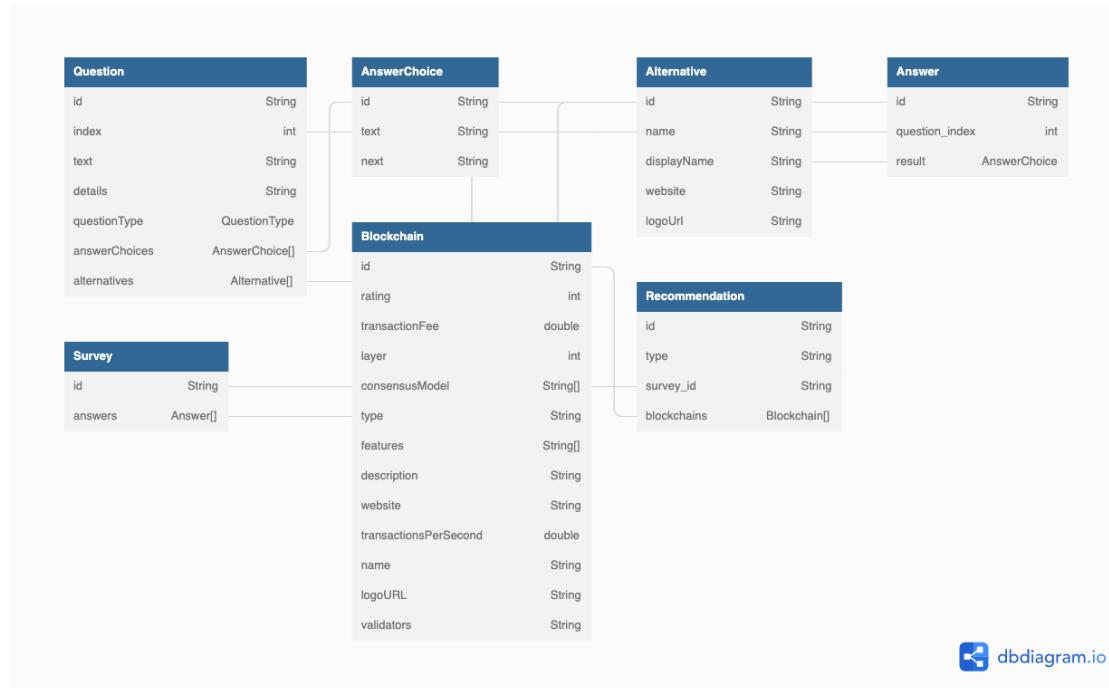


Abbildung 4.1: Datenbank-Schema des Frameworks

4.2 Komponenten

Die Grundfunktionen des Frameworks sollen folgende sein:

- Darstellung von beliebtesten Blockchains mit Vergleichsparametern
- Erklärungsseite mit den wichtigsten Informationen zum Framework
- Fragenkatalog zum Durchklicken für den Nutzer

- Alternativvorschläge, falls eine Blockchain keinen Sinn ergibt oder nicht möglich ist
- Empfehlungsseite des Blockchain-Typs
- Empfehlungsseite des Blockchain-Anbieters

Diese Grundfunktionen werden überwiegend in eigenen Komponenten implementiert, welche die Funktionalitäten größtmöglich kapseln sollen. Innerhalb dieses Unterkapitels werden die verschiedenen Komponenten beschrieben und ggf. anhand von Mockups verdeutlicht.

4.2.1 Blockchain-Vergleich

Bei dieser Komponente handelt es sich um eine Liste von Blockchains, die filterbar und sortierbar sein soll. Die Liste von Blockchains ist mit den beliebtesten Blockchains gefüllt. Im ersten Schritt wurde eine Liste von allen existierenden Blockchains angelegt. Es existieren bereits Listen mit allen aktiven Blockchains, denen regelmäßig neue Blockchains hinzufügt werden. Ein Beispiel ist Golden [25]. Mithilfe dieser Seite können bis zu 260 unterschiedliche Blockchains betrachtet werden. Das Problem der Liste ist, dass nur zu sehen ist, dass diese Blockchain existiert. Es ist nicht zu erkennen, wie erfolgreich eine Blockchain der Liste ist und wie viele Projekte auf Basis der Blockchains bisher erstellt wurden. Die Marktkapitalisierung (Börsenwert) einer Blockchain, also die Anzahl der umlaufenden Coins multipliziert mit dem aktuellen Marktpreis eines einzelnen Coins, soll für die Liste der Blockchains des Frameworks nicht betrachtet werden. Ein Grund ist, dass viele Anbieter privater Blockchains keinen Fokus auf eine Marktkapitalisierung legen und deswegen nicht entschieden werden kann, wie erfolgreich die Blockchain ist.

Aus diesem Grund und aufgrund der Unvollständigkeit anderer Blockchain-Listen wurde eine eigene Liste erstellt. Diese wird auf der Plattform Airtable gehostet. Abbildung 4.2 zeigt einen Ausschnitt von 10 Blockchains aus der Airtable Liste. Die Liste ist mit 117 Blockchains gefüllt und im Gegensatz zu anderen Listen (die im Internet veröffentlicht sind), verfügt sie über eine Spalte **Rating**. Ein Rating geht von 1 bis 5 Sternen und setzt sich aus der Anzahl an erfolgreich umgesetzten Projekten auf Basis einer Blockchain zusammen. Die Anzahl ist ermittelbar über die Plattform Github, auf der viele Projekte im Bereich der Blockchain gespeichert und dokumentiert sind. Private Projekte können durch dieses Verfahren nicht eingeschlossen werden, da diese nicht öffentlich zugänglich

KAPITEL 4. KONZEPTIONIERUNG

sind. Eine Blockchain mit 5 Sternen z.B. Ethereum ist demnach eine Blockchain mit vielen Referenzen zu aktiven Github-Projekten. Ein Projekt auf Github wird als «aktiv betreut» bezeichnet, wenn durch die Commit-Aktivität und dem Release-Intervall eine Weiterentwicklung zu erkennen ist.

Layer 1 Protocols *public*								
	Name	Brief (Marketing) Description	Website	Whitepaper/Docs	Type	BC Profile	Rating	...
1	Ethereum	Biggest smart contract platform	https://ethereum.org/	https://ethereum.org/...	Public chain	https://blockchai...	★★★★★	
2	Bitcoin	First, and most secure PoW chain.	https://bitcoin.org/de/	https://bitcoin.org/bit...	Public chain	https://blockchai...	★★★★★	
3	Polkadot	Blockchain ecosystem with share...	http://polkadot.network/	https://polkadot.netw...	Public chain	https://blockchai...	★★★★★	
4	Cosmos	Blockchain ecosystem of separat...	https://cosmos.network/	https://cosmos.netw...	Public chain	https://blockchai...	★★★★★	
	<input checked="" type="checkbox"/> Binance smart chain	Eth fork with solid adoption of do...	https://www.binance.or...	https://www.binance....	Public chain		★★★	
6	Near	Sharded PoS smart contract bloc...	https://nearprotocol.co...	https://near.org/pape...	Public chain	https://blockchai...	★★★	
7	Avalanche	Smart contract blockchain	https://www.avalabs.org/	https://www.avalabs....	Public chain		★★★	
8	Solana	High-throughput layer 1 blockch...	https://solana.com/	https://solana.com/s...	Public chain		★★★	
9	Moonbeam	Generalistic EVM compatible par...	https://moonbeam.net...	https://docs.moonbe...	Public chain		★★★	
10	Acala	EVM compatible parachain with ...	https://acala.network/	https://github.com/A...	Public chain		★★★	

Abbildung 4.2: Ausschnitt der vollständigen Blockchain-Liste mit 117 Blockchains auf der Plattform Airtable [1]

Airtable bietet die Möglichkeit, Einträge (Blockchains) zu sortieren und zu filtern; allerdings nur in Kombination mit Schreib-Rechten im Airtable-Projekt. Da ein Nutzer nur Leserechte bekommen soll und keine Veränderungen an der Datenbasis vornehmen darf, aber dennoch Funktionen wie das Sortieren und Filtern der Blockchains nutzen soll, wird eine eigene Implementation einer Tabelle konzeptioniert. Diese wird mit dem Frontend-Framework Angular umgesetzt. Die Tabelle soll nach Name, Beschreibung, Konsens Protokoll, Typ, Rating, Transaktionen pro Sekunde, Transaktionskosten, Features (z.B. Smart Contracts) und Anzahl der validierenden Knoten sortiert werden können. Initial ist die Tabelle anhand des vergebenen Ratings sortiert. Das Konsens Protokoll wie z.B. Proof of Work oder Proof of Stake bestimmt nicht nur, was eine Blockchain zu einem bestimmten Zeitpunkt erhalten soll, sondern kann auch andere Informationen beinhalten. Ein Beispiel ist das Proof of Work Protokoll, das für seinen hohen Energieverbrauch (benötigte Rechenleistung) bekannt ist. Die Anzahl der validierenden Knoten ist nicht als Ganzzahl, sondern als eigen erstellter Parameter mit den Optionen **low** und **high** abgebildet. Ersteres beschreibt, dass wenige validierenden Knoten im Netzwerk sind und Zweiteres, dass viele validierende Knoten im Netzwerk sind. Transaktionen pro Sekunde,

Transaktionskosten und die Anzahl der validierenden Knoten sollen von einer externen API-Schnittstelle angefragt und dann in Airtable gespeichert werden. Airtable bietet dazu für jede Tabelle eine eigene API mit Dokumentation und vorgefertigten Methoden.

Es soll eine Filterfunktion für den Typ, die Anzahl der validierenden Knoten, den Features (z.B. Smart Contracts) und der Anzahl der Transaktionen geben. In den vorliegenden Whitepapers der gelisteten Blockchains werden diese Kombinationen aus Parametern sehr detailreich beschrieben. Die Transaktionen pro Sekunde beziehen sich auf die Anzahl der Datentransaktionen, die ein Computernetzwerk innerhalb einer Sekunde durchführen kann. Transaktionen pro Sekunde ist keine Angabe, die nur innerhalb der Blockchain Domäne existiert. Der weltweit führende Anbieter von digitalen Zahlungen, VISA, gibt an, dass er mehr als 1.700 Transaktionen pro Sekunde verarbeiten kann und durchschnittlich 150 Millionen Transaktionen pro Tag abwickelt [8]. Die Blockchain Bitcoin kann derzeit 7 TPS verarbeiten und beschränkt eine Transaktion auf 250 Byte, während Ripple ungefähr 1.500 TPS verarbeiten kann. Je nach Anwendungsfall müssen viele oder wenige Transaktionen mit verschiedener Größe verarbeitet werden können. Beim Filter wird zwischen **public permissioned**, **private permissioned** und **public permissionless** Blockchains unterschieden. Je nach Anwendungsfall wird z.B. keine Beschränkung der Teilnehmer am Netzwerk (keine private permissioned Blockchain) benötigt.

Unter den filterbaren Parametern gibt es für die erste Version des Frameworks nur die Option, den Filter «Smart Contracts» an- und auszuschalten. Die Anzahl der validierenden Knoten gibt einem Nutzer des Frameworks die Optionen **low** und **high** zurück. Wenn eine Transaktion an die Blockchain übermittelt wird, führen die Validierungsknoten ein verteiltes Konsens-Protokoll aus, führen die Transaktion aus und speichern die Transaktion und die Ergebnisse in der Blockchain. Die Validierungsknoten entscheiden, welche Transaktionen und in welcher Reihenfolge Transaktionen der Blockchain hinzugefügt werden. Wenige Validierungsknoten machen ein Blockchain-Netzwerk automatisch zentraler. Wenn eine Aktion von wenigen Parteien abhängig ist, müssen auch weniger Parteien böswillig sein, um das Netzwerk anzugreifen. Bei vielen Validierungsknoten wird von einem dezentralen Blockchain-Netzwerk gesprochen. Ein Beispiel für diese Art der Blockchain ist Bitcoin mit 12.000 validierenden Computern im Netzwerk [32].

4.2.2 Blockchain-Empfehlung

Bei dieser Komponente handelt es sich um das Kernfeature des Frameworks. Das Ergebnis der hinterlegten Funktion der Komponente ist die Empfehlung eines Blockchain-Typs und eines Blockchain-Anbieters (Funktionale Anforderungen 02 - Ich möchte eine Blockchain-Typ-Empfehlung auf Basis der gegebenen Antworten bekommen und 03 - Ich möchte eine Blockchain-Anbieter-Empfehlung auf Basis der gegebenen Antworten bekommen). Dafür werden dem Nutzer im ersten Schritt Fragen gestellt, die dieser dann mithilfe der vordefinierten Antwortmöglichkeiten beantwortet. Wird eine Frage so beantwortet, dass das Framework entscheidet keine Blockchain zu empfehlen, soll der Nutzer dies direkt im Anschluss erfahren und keine weiteren Fragen gestellt bekommen. Daraufhin bekommt der Nutzer Alternativen zu einer Blockchain vorgeschlagen, je nachdem an welchem Punkt dieser im Aktivitätsdiagramm ausgestiegen ist. Neben der eigentlichen Frage soll ein Hilfstext bei wenig Erfahrung oder Wissen in dem Bereich Blockchain bei der Beantwortung der Frage bzw. dem Verständnis der Frage unterstützen. Eine Progress-Bar (Fortschrittsanzeige) soll dem Nutzer anzeigen, was für einen prozentualen Anteil der Fragen bereits beantwortet wurde und wie viel bis zur Empfehlung fehlt.

4.3 Algorithmus

Folgender Abschnitt beschreibt den Algorithmus der Blockchain-Empfehlung. Durch Grafiken und Beschreibungen soll die Idee dahinter vermittelt werden.

4.3.1 Programmatische Umsetzung

Der Nutzer bekommt spezielle Fragen zum Thema Blockchain gestellt. Abbildung 4.3 zeigt das Aktivitätsdiagramm für die Blockchain-Empfehlung.

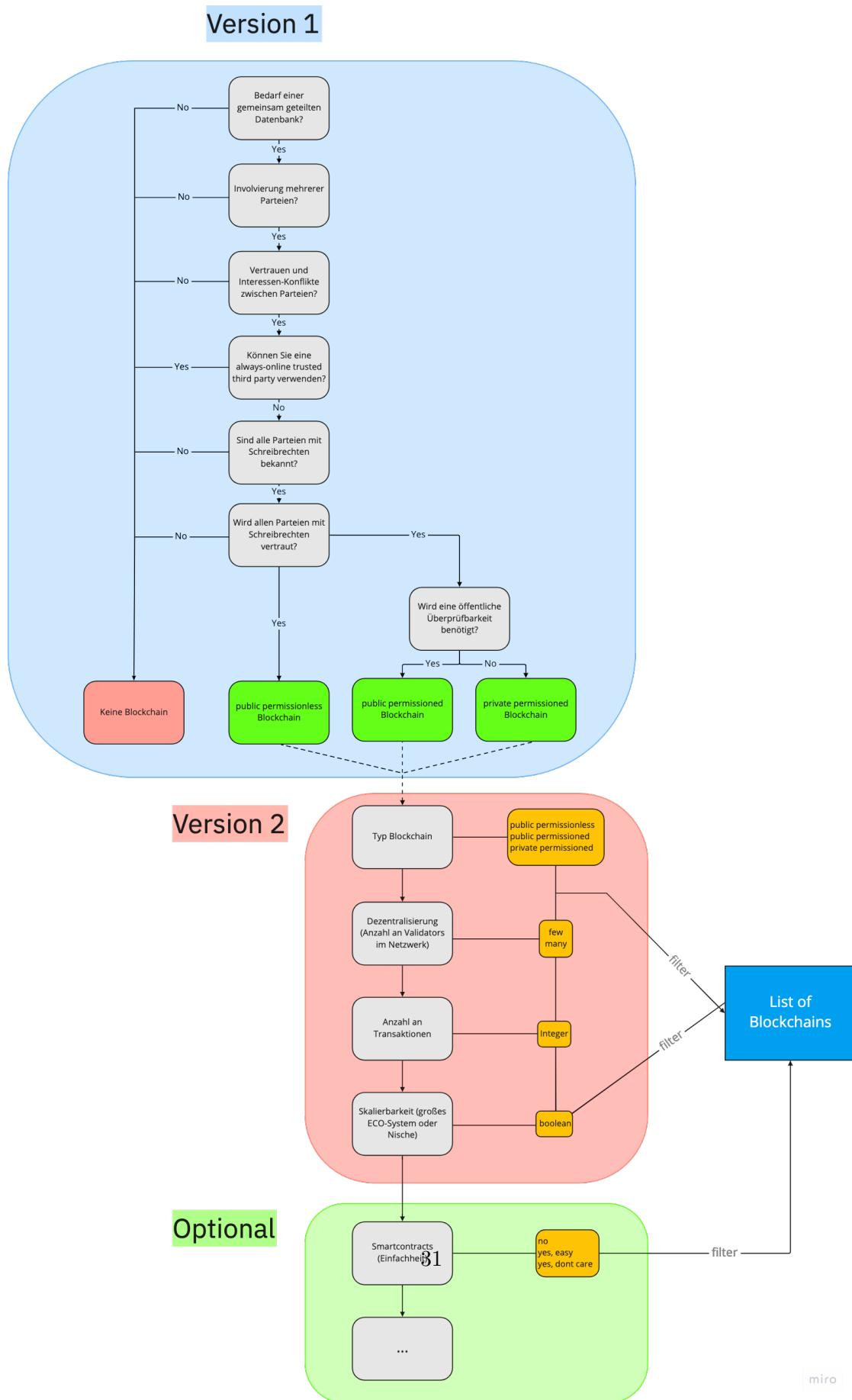


Abbildung 4.3: Aktivitätsdiagramm der Blockchain-Empfehlung

Im Folgenden werden die im Diagramm genannten Fragen genauer beschrieben und erläutert.

1. Gibt es einen Bedarf einer gemeinsam geteilten Datenbank? Die Grundaufgabe einer Blockchain ist es, eine dezentrale Datenbank darzustellen. Im ersten Schritt soll herausgefunden werden, ob eine traditionelle Datenbank die gleiche Funktion wie eine Blockchain bieten kann. Wenn keine Daten abgespeichert werden müssen, kann die Notwendigkeit einer Blockchain infrage gestellt werden [23]. Sollen Daten persistent abgespeichert werden, muss die Zahl der Daten betrachtet werden. Je höher die Anzahl, desto höher können auch die Kosten für die aktive Benutzung der Blockchain werden. Bei einem großen Datensatz, der durch einen Verifikationsprozess gelangen muss, leidet die Geschwindigkeit der Blockchain. Es existieren Ausnahmen, die für diesen Anwendungsfall konzeptioniert worden sind und große Datenmengen bzw. Transaktionen pro Sekunde unterstützen. Um diese Skalierbarkeitsprobleme zu verhindern, kann alternativ zur Blockchain auf eine off-chain Datenbank oder simpler, auf eine konventionelle Datenbank zurückgegriffen werden [6].

Wird diese Frage mit einem «Nein» beantwortet, werden Alternativen vorgeschlagen. Die Alternativen auf diese Frage sind **shared documents**. Das sind Dokumente, die nach Freigabe von Anderen gelesen und bearbeitet werden können. Anbieter von shared documents sind Microsoft Excel, Google Sheets, Airtable und Zoho Sheets.

Sind mehrere Parteien involviert? Diese Frage zielt auf die Dezentralität einer Blockchain bzw. der Datenbank ab. Existieren im Anwendungsfall mehrere Parteien, die interagieren und mehr als eine Partei die Dokumente hinzufügen und updaten kann, so ist die Verwendung einer Blockchain machbar.

Es existieren auch Fälle, bei denen nicht jeder Writer auch einen Knoten im Blockchain-Netzwerk betreibt; z.B. bei einem Bezahlungssystem, das von einer Gruppe von Banken verwaltet wird und Millionen an Endkunden auf mobilen Geräten hat, die alle mit dem eigenen Banksystem kommunizieren. Die Endkunden können Transaktionen erstellen, ohne dass sie selber ein Knoten sind [44].

Wenn diese Frage mit «Nein» beantwortet wird, werden Alternativen vorgeschlagen und die Blockchain-Technologie ist überflüssig. Alternativen sind zentrale Datenbanksysteme. Da wenige Parteien involviert sind, braucht es keine Dezentralisierung der Datenbank. Die Alternativvorschläge sind Anbieter wie z.B. MongoDB, Postgre oder Cloud-Anbieter

wie Google-Cloud, IBM Cloudant, Microsoft Azure Cloud oder AWS-Cloud.

Haben die involvierten Parteien Vertrauen und/oder Interessen-Konflikte?

Ein Vorteil der Blockchain ist, dass ein Trust-Free-Eco-System aufgebaut wird. Damit kann jede involvierte Partei darauf vertrauen, dass alle gespeicherten Daten korrekt und vollständig sind, anstatt sich darauf zu verlassen, dass Parteien nicht böswillig sind.

Die Frage, die gestellt werden muss, ist, wie die Parteien in der Anwendung zueinander stehen. Wenn sich alle Parteien bei Informationsangaben vollständig aufeinander verlassen können, ist eine Blockchain nicht notwendig. Wenn Parteien Vertrauenskonflikte aufweisen, ist eine Blockchain zu empfehlen. Wenn die Parteien Interessenkonflikte aufweisen oder den Daten einer Partei nicht absolut vertraut werden kann, kann die Blockchain die automatische Datenüberprüfung und Speicherung ermöglichen. Dadurch können Parteien zuverlässig miteinander interagieren. Die Alternativvorschläge auf die Frage sind die gleichen, wie in der vorherigen Frage.

Können Sie eine Trusted-Third-Party verwenden? Wenn in einem Anwendungsfall verschiedene Parteien Vertrauenskonflikte haben, dann kann eine Trusted-Third-Party (TTP) Abhilfe schaffen. Es gibt zwei Arten einer TPP. Die eine TTP ist immer online, sodass Schreib-Operationen an diese delegiert werden und diese als Prüfer für Zustandsübergänge fungieren kann. Die andere TTP ist für gewöhnlich offline und operiert als Certificate Authority (CA) in einer Blockchain, in der alle Parteien, die Schreib-Operationen ausführen dürfen, bekannt sind [23]. Eine TTP ist allerdings nicht in jedem Anwendungsfall gewollt und möglich. Wenn die TTP nicht vertrauenswürdig ist oder böswillig scheint, entstehen neue Vertrauenskonflikte. Wenn auf eine TTP verzichtet werden soll, ermöglicht die Blockchain Peer-to-Peer Transaktionen ohne dritte Partei. Dadurch können Transaktionen von Nodes, die eine eigene Kopie der Datenbank pflegen, unabhängig verifiziert und verarbeitet werden [44].

Sind alle Parteien mit Schreibrechten bekannt? Je nachdem, ob eine öffentliche Überprüfbarkeit erforderlich ist, kann jeder Teilnehmer in der Blockchain den Zustand «lesen» (public permissioned Blockchain). Die Anzahl der Leser kann allerdings auch eingeschränkt werden (private permissioned Blockchain). Wenn die Anzahl der Schreiber nicht festgelegt und den Teilnehmern nicht bekannt ist, wie es bei vielen Kryptowährungen wie Bitcoin der Fall ist, ist eine public permissionless Blockchain eine geeignete Lösung.

5 — Implementierung und Ergebnisse

Das nachfolgende Kapitel beschreibt das Vorgehen zur konkreten Implementierung der im vorherigen Kapitel beschriebenen Konzepte. Dazu werden zunächst unterstützende Methoden und Maßnahmen zur Entwicklung beschrieben, bevor dann auf die Implementierung der Komponenten eingegangen wird.

5.1 Entwicklungsumgebung

In der Entwicklung des Frameworks wurde die IDE (Integrated Development Environment) IntelliJ Idea Ultimate von der Firma JetBrains verwendet. Diese verfügt im Gegensatz zu der kostenlosen Version IntelliJ Idea Community Edition über einen offiziellen Support der verwendeten Frameworks Angular und Spring [34]. Mithilfe von sog. Plugins, die auch als Erweiterungen der IDE bezeichnet werden, kann eine Umgebung geschaffen werden, in der sowohl das Angular-Frontend, das Spring-Boot-Backend und die MongoDB Datenbank mit zusätzlichen Funktionen und Features erweitert werden.

5.2 Git

Der zu erstellende Code wird mithilfe von Git versioniert [12] und im Firmen eigenen GitLab abgelegt. Dieses bietet u.a. die Sicherheit, dass der Code nicht auf einem lokalen Rechner verloren geht und um zusätzlich eine geräteunabhängige Entwicklung zu ermöglichen. Darüber hinaus lassen sich verschiedene Versionen von Daten miteinander vergleichen, die es erleichtern, Änderungen nachzuvollziehen [12]. GitLab ist ein Anbieter, der zusätzlich eine visuelle Oberfläche und das einfache Anbinden von CI/CD-Prozessen ermöglicht und die Software Git verwendet [24]. Ein Vorteil in der Verwendung

von GitLab besteht darin, dass alle Teammitglieder in jeder Projektphase zusammenarbeiten können. GitLab bietet eine gute Nachvollziehbarkeit von der Planung bis zur Erstellung, um Entwicklern zu helfen, den gesamten DevOps-Lebenszyklus zu automatisieren [24]. In diesem Projekt wurde GitLab benutzt, um verschiedene Versionen des Frameworks zu verwalten und eine Pipeline zu starten, die das Produkt zusammenbauen und in die Google Cloud deployen soll.

Abbildung 5.1 zeigt die Pipeline in GitLab. Sobald Änderungen am Quellcode vorgenommen werden, werden diese in das GitLab Repository übernommen. Darauf folgend wird ein Prozess namens «Cloud Build» gestartet. Dieser Prozess klont den bestehenden Quellcode aus dem GitLab Verzeichnis und erzeugt ein neues Docker-Image. Cloud Build sorgt dafür, dass das neu erstellte Docker-Image in das sog. «Artifact-Repository» gespeichert wird und weißt dann den «Cloud Run» Prozess an, das neu erstellte Docker-Image zu verwenden. Cloud Run erwartet, dass ein Docker-Image ausgeführt wird und fragt das Artifact-Repository nach diesem Image. Das Artifact-Repository stellt dann dieses Image bereit. Artifact-Respository und der Prozess Cloud Run befinden sich in der Google Cloud. Die Google Cloud bietet dem Endkunden eine Schnittstelle zur Verwendung des Frameworks («Client App») und fungiert somit als Host.

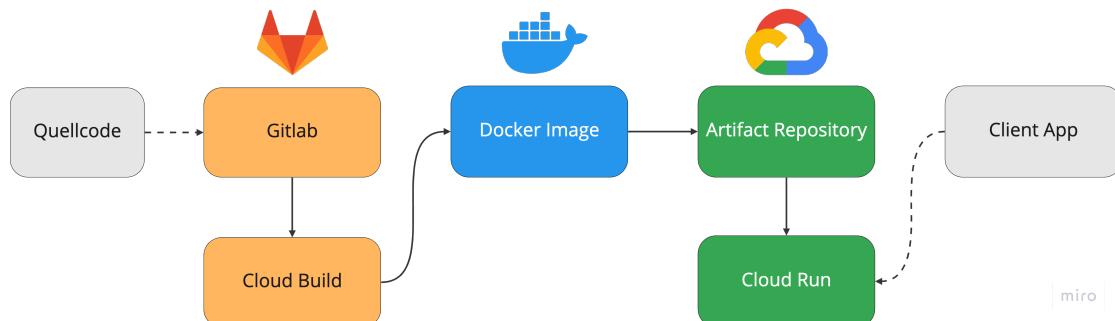


Abbildung 5.1: Pipeline des Frameworks in GitLab

5.3 Maven

Maven ist ein Open-Source-Build-Tool, das von der Apache Group entwickelt wurde, um mehrere Projekte gleichzeitig zu erstellen, zu veröffentlichen und bereitzustellen und so die Projektverwaltung zu verbessern. Maven bietet automatisiertes Abhängigkeitsmanagement und ermöglicht es Entwicklern, Build-Abhängigkeiten in sog. `pom.xml`-Dateien festzulegen. Das Tool verwaltet diese Abhängigkeiten automatisch und lädt sie bei Bedarf

herunter.

Für den Start des Frontends und des Backends unterstützt eine jeweilige «Maven-Datei». Damit GitLab das bereits genannte Docker-Image erzeugen kann, musste in den Dateien festgelegt werden, welche Frameworks und Technologien benutzt werden. Die `pom.xml` Datei des Spring-Boot-Backends wird im Codeausschnitt (siehe Anhang 8.1) gezeigt. Zunächst wurde festgelegt, wie das Projekt benannt werden soll. Unter «properties» wurde die verwendete Programmiersprache und Version festgehalten. Anschließend kann unter «dependencies» die Liste von Frameworks und Erweiterungen hinzugefügt werden. In diesem Codeausschnitt des Spring-Boot-Backends werden beispielsweise `springdoc`, `springframework.boot` und `querydsl` verwendet.

5.4 Docker

Docker bietet die Möglichkeit, eine Anwendung in eine lose isolierten Umgebung (virtuellen Umgebung), einem Container, zu verpacken und auszuführen. Die Isolierung und Sicherheit ermöglicht es, viele Container gleichzeitig auf einem bestimmten Host-Gerät auszuführen. Die Container sind lightweight und enthalten alles, was für die Ausführung der Anwendung erforderlich ist, sodass sich nicht darauf verlassen werden muss, dass alle benötigten Technologien vorab auf dem Host-Gerät installiert sind. Im Projekt wurde außerdem eine Docker-Compose-Datei für das Starten der Docker-Container benutzt. Jedes Teilprojekt (Frontend, Backend, Datenbank) besitzt eine eigene Dockerfile im Root-Verzeichnis des Projektes, aus der die Docker-Compose-Datei die Rahmenbedingungen für ein Docker-Image bekommt. Abbildung 5.2 zeigt, wie der Docker-Prozess funktioniert und Codeausschnitt 5.1 zeigt den Inhalt der Docker-Compose-Datei.

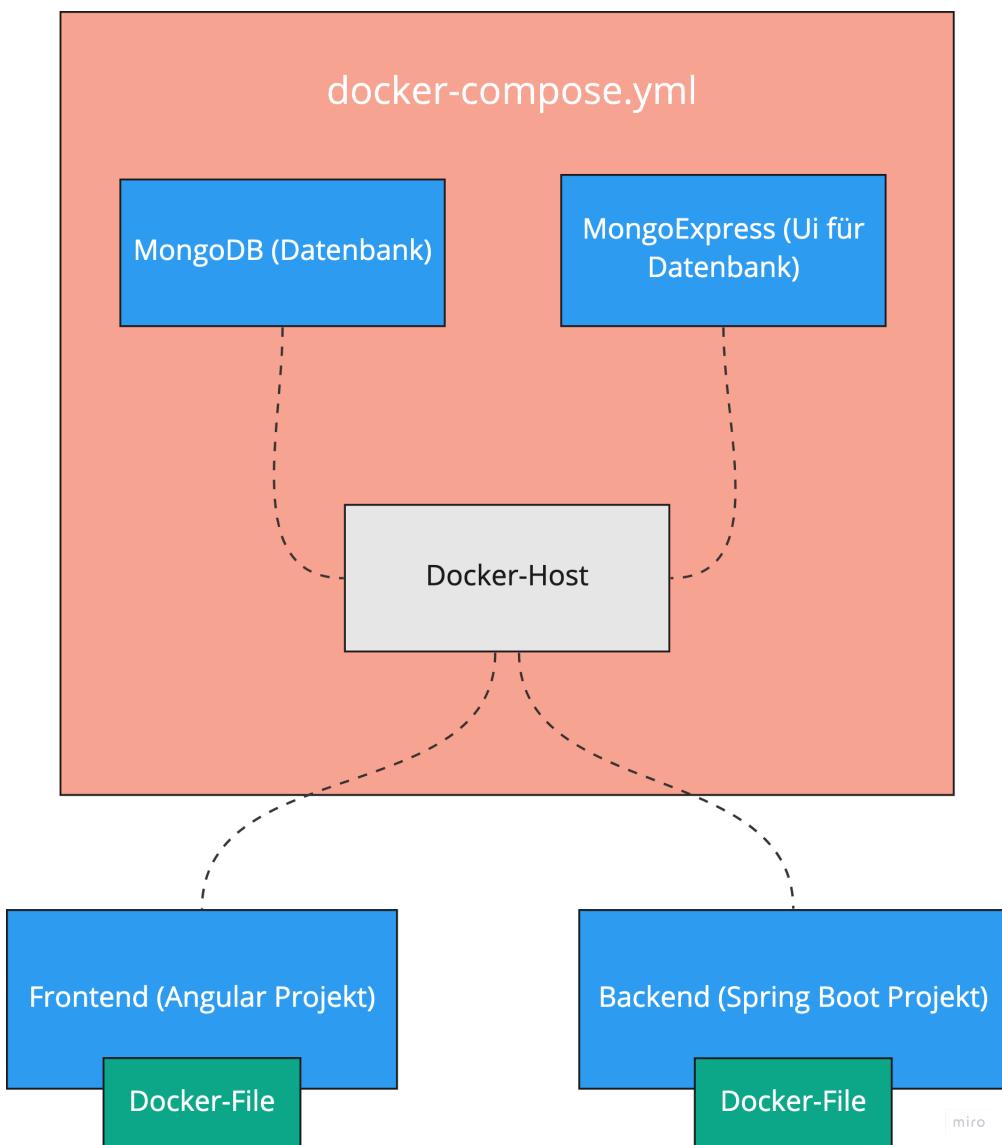


Abbildung 5.2: Diagramm für den Ablauf des Docker Prozesses

In der Abbildung 5.2 ist zu erkennen, dass insgesamt 4 Services existieren; die Services `backend-service`, `mongodb`, `frontend-service` und `mongo-express`. Der `frontend-service` und `backend-service` besitzen sog. Docker-Dateien. Diese beinhalten Parameter zur Erstellung eines Docker-Containers. Die Services `mongodb` und `mongo-express` besitzen keine eigene Docker-Datei, denn ihre Parameter sind in der Docker-Compose-Datei bereits definiert. Beim Start des Docker-Prozesses liest der Docker-Host aus der `docker-compose.yml` und den Docker-Dateien der beiden Services die Parameter aus.

Codeausschnitt 5.1 zeigt die Docker-Compose-Datei. Der Backend-Service beschreibt das Spring-Boot-Backend, das beim Compilieren in der IDE unter dem Containernamen `backend-service` erzeugt wird, weshalb in der Docker-Compose-Datei dieser Name angegeben ist. Dieser Service ist abhängig vom `mongodb` Container, der die Datenbank stellt. Das heißt, das Backend wird erst gestartet, wenn die Datenbank erfolgreich gestartet wurde. Der MongoDB Container speichert die Datenbank-Daten im Docker-Container. Das wird durch das Attribut `volumes` festgelegt. Außerdem müssen noch der `Root-Username` und das `Root-Password` ergänzt werden, um den Zugang nicht öffentlich zu machen.

Codeauschnitt 5.1: Inhalt der Docker-Compose.yaml Datei

```
version: '3.7'
services:
  backend-service:
    build: ./backend-service
    restart: always
    container_name: backend-service
    ports:
      - "9123:9123"
    depends_on:
      - mongodb

  mongodb:
    image: mongo
    container_name: mongodb
    restart: always
    ports:
      - "27017:27017"
    volumes:
      - data:/data
    environment:
      - MONGO_INITDB_ROOT_USERNAME=rootuser
      - MONGO_INITDB_ROOT_PASSWORD=rootpass

  mongo-express:
    image: mongo-express:latest
    container_name: mongo-express
    restart: always
```

```

ports:
  - "8081:8081"
environment:
  - ME_CONFIG_MONGODB_ADMINUSERNAME=rootuser
  - ME_CONFIG_MONGODB_ADMINPASSWORD=rootpass
  - ME_CONFIG_MONGODB_SERVER=mongodb
volumes:
  data: { }

```

5.5 Frontend

5.5.1 Technologien

Angular Das Angular-Projekt läuft auf Version 14 (Release Juni 2022) und besteht aus verschiedenen Untergruppen. Dazu zählen Components, Services, Models und Routing-Klassen. Abbildung 5.3 zeigt den Aufbau des Frontends in der IDE.

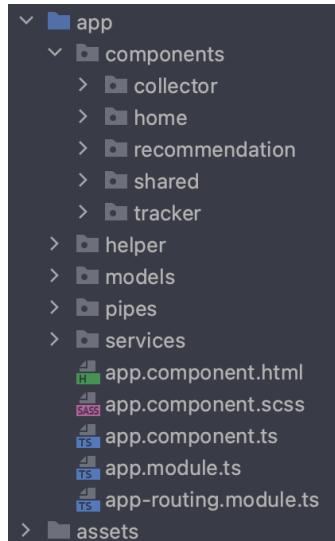


Abbildung 5.3: Struktur des Frontend-Projektes

Tailwind-CSS Tailwind-CSS ist ein CSS-Framework, bei dem der Nutzen im Vordergrund steht [14]. Im Gegensatz zu anderen CSS-Frameworks wie Bootstrap oder Materialize-CSS hat es keine vordefinierten Komponenten. Stattdessen arbeitet Tailwind-CSS auf einer niedrigeren Ebene und bietet eine Reihe von CSS-Hilfsklassen. Codeaus-

schnitt 5.2 zeigt ein Beispiel für die Verwendung von Tailwind-CSS in Angular bzw. HTML-Dateien. Die vordefinierten CSS-Klassen können unter dem «class» (Klassen) Attribut einer HTML-Komponente verwendet werden. Zeile 1 gibt beispielsweise an, dass als Hintergrundfarbe slate-100 verwendet werden soll (`bg-slate-100`), abgerundete Ecken entstehen sollen (`rounded-xl`) und ein Padding von 8 Pixeln um die HTML-Komponente «figure» entstehen soll(`p-8`).

Codeauschnitt 5.2: Tailwind-CSS Beispiel [58]

```
<figure class="bg-slate-100 rounded-xl p-8">
  
  <div class="pt-6 space-y-4">
    <blockquote>
      <p class="text-lg font-medium">
        Tailwind CSS is the only framework that I've seen scale
        on large teams. It's easy to customize, adapts to any design,
        and the build size is tiny.
      </p>
    </blockquote>
  </div>
</figure>
```

5.5.2 Architektur

Das Frontend besteht aus verschiedenen Komponenten (siehe Abbildung 5.4 und Abschnitt 2.1.2). Jede Komponente besteht aus einer Benutzeroberfläche (in HTML und CSS entwickelt) und einer Logik, die in TypeScript realisiert wurde. Die Root-Komponente (in Angular) ist die Komponente, die alle Komponenten kennt und durch das sog. Routing entscheiden kann, welche angezeigt werden soll. Die Home-Komponente besteht aus einer Blockchain-Liste (siehe Abschnitt 4.2.1) und einer Informations-Komponente, die dem Nutzer erklären soll, wie die Anwendung zu bedienen ist. Die Blockchain-Liste benutzt weitere Komponenten aus dem Verzeichnis der «shared components» (siehe Abbildung 5.5). Die Methoden für das Sortieren und Filtern wurden in shared components ausgelagert, da beide Funktionen lose gekapselt sein sollten, sodass sie in weiteren Komponenten implementiert werden können. Die Kontakt-Komponente verfügt über ein Impressum und die Funktion zum Absenden eines Kontaktformulars bei Fragen oder Problemen des Kunden. Als letzte Komponente wurde die Umfrage-Komponente

erstellt. Diese steht für den Umfragenkatalog, den der Nutzer angezeigt bekommt (siehe Abschnitt 4.3.1).

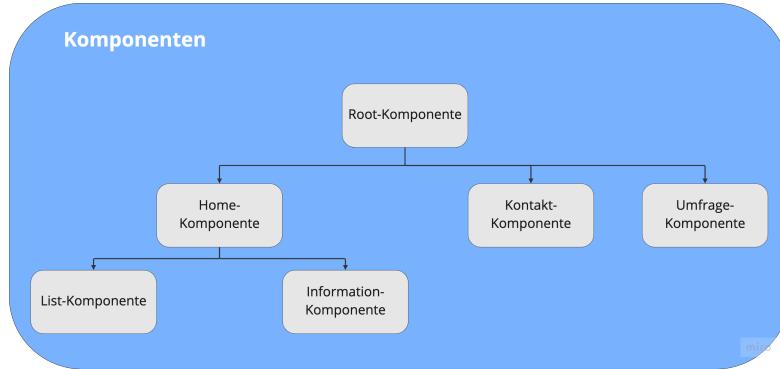


Abbildung 5.4: Komponenten im Angular-Frontend

In Abbildung 5.5 werden die shared components des Frontends angezeigt. Die Header-Komponente ist die Navigation der Anwendung, die klassischerweise auf jeder Seite des Frameworks zur Verfügung steht. So kann der Nutzer von überall auf die geforderte Seite navigieren. Die Error-Komponente wird ebenfalls von mehreren Komponenten verwendet. Diese zeigt dem Nutzer nützliche Fehlermeldungen und Erfolgsmeldungen an.



Abbildung 5.5: Geteilte Komponenten (sog. shared components) im Angular-Frontend

Abbildung 5.6 zeigt die Services des Frontends (siehe Abschnitt 2.1.2). Der Name des Services steht für seine Funktion. Im Folgenden wird beispielhaft der Aufbau des «Blockchain-Service» erläutert. Der Blockchain-Service kommuniziert mit der REST-API Schnittstelle des Backends, wenn Objekte vom Typ «Blockchain» angefragt, gespeichert, upgedatet oder gelöscht werden sollen. Eine detaillierte Beschreibung der Rest-Schnittstelle und Kommunikation wird im Abschnitt 5.6.3 erläutert.

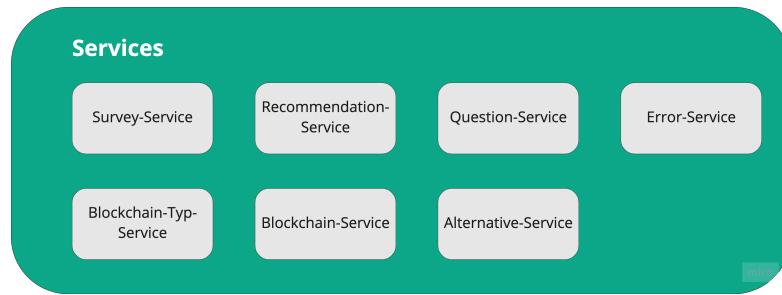


Abbildung 5.6: Services im Frontend

Codeausschnitt 5.3 zeigt die Funktion `all`, die eine Anfrage an die REST-API Schnittstelle stellt und alle Objekte vom Typ Blockchain anfordert. Diese Anfrage funktioniert über eine *HTTP – Anfrage*. Die Parameter, die zusätzlich beim GET-Request mitgegeben werden, sind die Filteroptionen, um nach bestimmten Blockchains zu filtern. In diesem Beispiel hat die Schnittstelle die URL `/api/v1/blockchains`. Zurückgegeben wird ein `Observable` vom Typ `Blockchain[]` (Blockchain Array). Falls die Anfrage scheitert, wird durch den Programmcode `pipe(retry(1))` festgelegt, ein weiteres Mal die Anfrage zu starten. Scheitert diese erneut, wird die Funktion `handleError` aus dem `Error-Service` aufgerufen, die dem Nutzer über die Error-Komponente eine sinnvolle Fehlernachricht anzeigt.

Codeausschnitt 5.3: blockchain.service.ts Ausschnitt

```
public all(filter: Filter): Observable<Blockchain[]> {
    const params = this.setParams(filter);
    return this.http.get<Blockchain[]>('/api/v1/blockchains', {params})
        .pipe(retry(1), catchError(this.errorService.handleError));
}
```

5.5.3 Benutzeroberfläche

In folgendem Abschnitt wird die erstellte Benutzeroberfläche des Frameworks beschrieben. Die Beschreibung ist in zwei Abschnitte unterteilt; es wurde sowohl eine Prototyp-Version der UI, als auch eine erweiterte Version der UI konzeptioniert. Die Implementierung der erweiterten Version war in Rahmen dieser Thesis nicht möglich.

Homescreen Abbildungen 5.7 und 5.8 zeigen die beiden Varianten des Homescreens. Dieser besteht aus einer Navigationsleiste und der Informationsseite. Die Navigations-

KAPITEL 5. IMPLEMENTIERUNG UND ERGEBNISSE

leiste gibt dem Nutzer Zugang zum Fragenkatalog ([Find Blockchains](#)) und zu der Blockchain-Liste ([Blockchains](#)). Die Informationsseite gibt dem Nutzer die nötigen Informationen über die Dauer des Fragenkatalogs und den Schritten. Die erweiterte Version (Abbildung 5.8) wurde mit einem modernen Theme umgesetzt. Auch hier wurde eine Navigationsleiste erstellt. Zusätzlich gibt es die Einträge, «Experten-Seite» und «Kontakt-Seite». Die Experten-Seite verfügt über eine zukünftige Funktion, um potenzielle Kunden zu kontaktieren. Dort sollen Mitarbeiter der Firma dargestellt werden, die Experten auf dem Gebiet der Blockchain sind. Kunden, die Interesse an einem Projekt haben, können dort die nötigen Informationen bekommen.

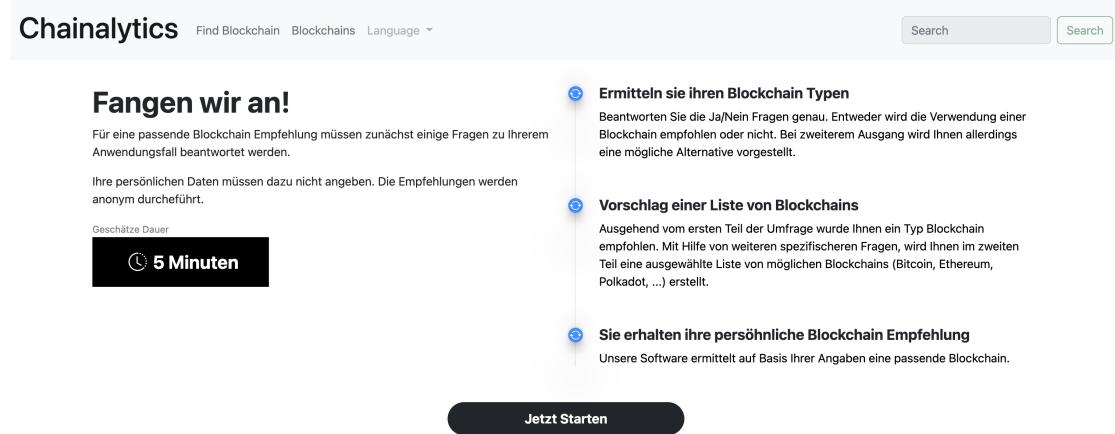


Abbildung 5.7: Homescreen Prototyp

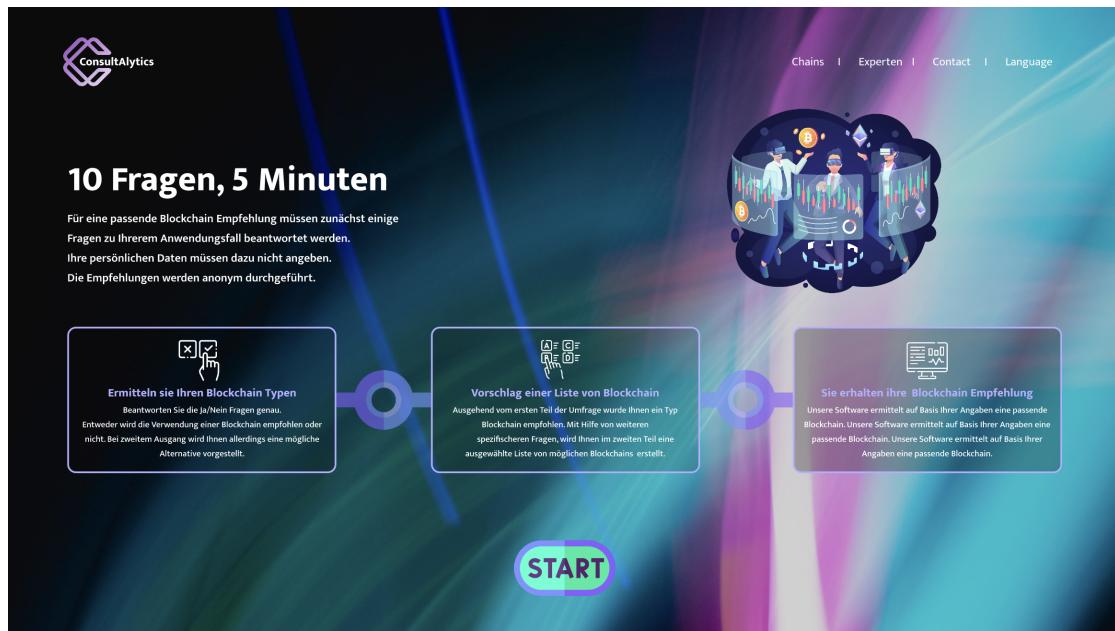


Abbildung 5.8: Homescreen erweiterte Version

Fragenkatalog Die UI für den Fragenkatalog ist in Abbildung 5.9 und 5.10 dargestellt. Die Seite verfügt ebenfalls über die Navigationsleiste. Die Progress-Bar füllt sich, wenn der Nutzer die Fragen beantwortet (vgl. Abschnitt 4.2.2). Im unteren Teil wird die eigentliche Frage zusammen mit dem Index der Frage angezeigt. Die Abbildung zeigt die erste Frage. Durch Drücken der Schaltflächen «Richtig» oder «Falsch» kann der Nutzer die Frage beantworten und wird nach Eingabe auf die nächste Seite weitergeleitet. Unterhalb der Frage ist eine detailliertere Beschreibung zusehen, falls die Frage nicht eindeutig nachvollziehbar ist. Die Komponenten der Prototyp-Version und der erweiterten Version der UI unterscheiden sich in der Funktionalität nicht.

KAPITEL 5. IMPLEMENTIERUNG UND ERGEBNISSE

The screenshot shows a user interface for a blockchain-related quiz or knowledge base. At the top, there's a navigation bar with the logo 'Chainalytics' and links for 'Find Blockchain', 'Blockchains', 'Language', and two search fields. Below the header is a progress bar indicating '14%' completion. The main content area features a question titled '2. Involvierung mehrerer Parteien?' (Involvement of multiple parties?). Two buttons are present: 'Richtig' (Correct) and 'Falsch' (False). A small callout box below the buttons provides additional context: '↓ Mehr Informationen zur Frage' (More information about the question), followed by a detailed explanation: 'Wird eine dezentrale Datenbank benötigt? Können mehrere Parteien durch die Anwendung interagieren und mehr als eine Partei Dokumente verwalten, ist eine Blockchain möglich'. The overall design is clean and modern.

Abbildung 5.9: Fragenkatalog der Prototyp-Version

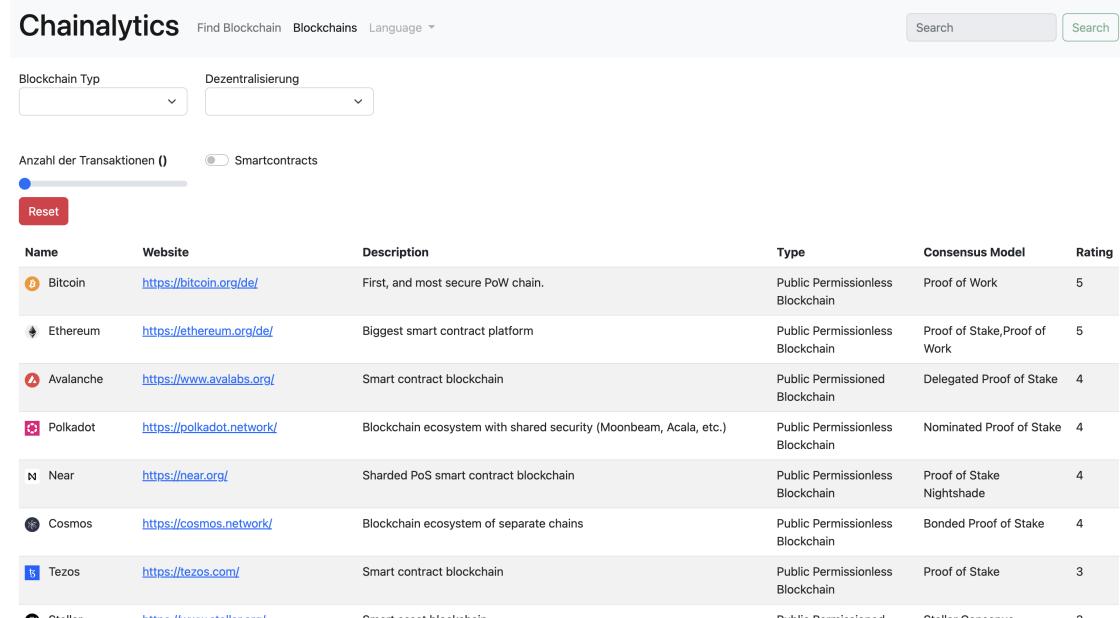
This screenshot displays the extended version of the Chainalytics questionnaire. The layout is similar to the prototype, featuring a dark background with colorful light streaks. The top navigation includes the 'ConsultAnalytics' logo, a progress bar at 10%, and links for 'Chains', 'Wiki', 'Contact', and 'Language'. The central question is '1. Bedarf einer gemeinsamen geteilten Datenbank?' (Requirement for a common shared database). Two large rectangular buttons are shown: one labeled 'FALSCH' (False) with a red X icon and another labeled 'RICHTIG' (True) with a green checkmark icon. Below each button is a small explanatory text: 'Hier gibt es mehr info zur Frage, Blabla mehr Details. Etc' (There is more information about the question, Blabla more details. Etc). A hand cursor icon is positioned between the two buttons, pointing towards the 'RICHTIG' button. The overall design is more dynamic and visually appealing than the prototype version.

Abbildung 5.10: Fragenkatalog erweiterte Version

Blockchain-Liste Die «Blockchain-Liste» Komponente ist in Abbildung 5.11 und 5.12 zu sehen. Die Funktionalität und Bedienbarkeit beider Versionen sind identisch, lediglich die Benutzeroberflächen unterscheiden sich. Durch die Schaltflächen **Blockchain-Typ**, **Dezentralisierung**, **Anzahl der Tansaktionen** und **Smart Contracts** kann die Liste der Blockchains gefiltert werden. Die Sortierung ist über einen Klick auf die jeweilige

KAPITEL 5. IMPLEMENTIERUNG UND ERGEBNISSE

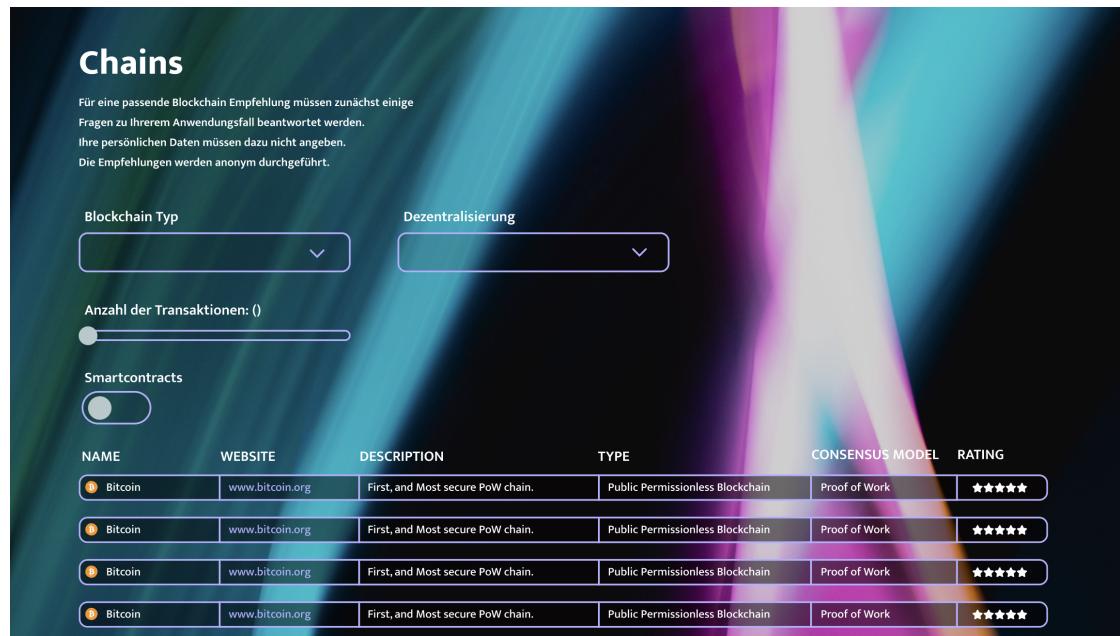
Spalte in der Liste möglich.



The screenshot shows a web application interface titled "Chainalytics". At the top, there are search fields for "Find Blockchain", "Blockchains", and "Language", along with a "Search" button. Below these are dropdown menus for "Blockchain Typ" and "Dezentralisierung", and a slider for "Anzahl der Transaktionen ()". A "Smartcontracts" toggle switch is also present. A "Reset" button is located at the bottom left of the form area. The main content is a table listing various blockchains:

Name	Website	Description	Type	Consensus Model	Rating
Bitcoin	https://bitcoin.org/de/	First, and most secure PoW chain.	Public Permissionless Blockchain	Proof of Work	5
Ethereum	https://ethereum.org/de/	Biggest smart contract platform	Public Permissionless Blockchain	Proof of Stake, Proof of Work	5
Avalanche	https://www.avalabs.org/	Smart contract blockchain	Public Permissioned Blockchain	Delegated Proof of Stake	4
Polkadot	https://polkadot.network/	Blockchain ecosystem with shared security (Moonbeam, Acala, etc.)	Public Permissionless Blockchain	Nominated Proof of Stake	4
Near	https://near.org/	Sharded PoS smart contract blockchain	Public Permissionless Blockchain	Proof of Stake, Nightshade	4
Cosmos	https://cosmos.network/	Blockchain ecosystem of separate chains	Public Permissionless Blockchain	Bonded Proof of Stake	4
Tezos	https://tezos.com/	Smart contract blockchain	Public Permissionless Blockchain	Proof of Stake	3

Abbildung 5.11: Blockchain-Liste Prototyp-Version



The screenshot shows a web application interface titled "Chains". It includes a message about blockchain recommendations and user privacy. Below the message are dropdown menus for "Blockchain Typ" and "Dezentralisierung", a slider for "Anzahl der Transaktionen ()", and a "Smartcontracts" toggle switch. The main content is a table listing multiple entries of the blockchain "Bitcoin":

NAME	WEBSITE	DESCRIPTION	TYPE	CONSENSUS MODEL	RATING
Bitcoin	www.bitcoin.org	First, and Most secure PoW chain.	Public Permissionless Blockchain	Proof of Work	★★★★★
Bitcoin	www.bitcoin.org	First, and Most secure PoW chain.	Public Permissionless Blockchain	Proof of Work	★★★★★
Bitcoin	www.bitcoin.org	First, and Most secure PoW chain.	Public Permissionless Blockchain	Proof of Work	★★★★★
Bitcoin	www.bitcoin.org	First, and Most secure PoW chain.	Public Permissionless Blockchain	Proof of Work	★★★★★

Abbildung 5.12: Blockchain-Liste erweiterte Version

5.6 Backend

5.6.1 Technologien

Spring Boot Es wurde die neuste Version von Spring Boot (Spring-Boot 2.7) (vgl. Abschnitt 2.1.2) verwendet. Durch eine Erweiterung namens **QueryDSL** kann die Kommunikation zwischen der MongoDB Datenbank und dem Spring Backend vereinfacht werden.

QueryDSL QueryDSL ist ein Framework, das die Erstellung von typsicherer SQL-ähnlichen Abfragen für mehrere Backends einschließlich JPA, MongoDB und SQL in Java ermöglicht [47]. QueryDSL bietet vordefinierte Funktionen zum Abfragen der MongoDB Datenbank [47]. Codeausschnitt 5.4 zeigt die Klasse **AlternativRepository**, die Java-Funktionen mithilfe von QueryDSL definieren kann. Die Klasse AlternativRepository erbt von der Klasse QuerydslPredicateExecutor. Die Funktion `findByIds` ist also auch der Klasse AlternativRepository bekannt. Dadurch, dass QueryDSL das Datenbankschema kennt, werden die sog. CRUD-Operationen vordefiniert.

Codeausschnitt 5.4: Codeausschnitt AlternativRepository

```
public interface AlternativeRepository  
extends MongoRepository<Alternative, String>,  
    QuerydslPredicateExecutor<Alternative> {  
  
    List<Alternative> findByIds(List<String> ids);  
}
```

OpenAPI Die OpenAPI-Spezifikation (OAS) definiert eine standardisierte, sprachunabhängige Schnittstelle zu RESTful-APIs, die es sowohl Menschen als auch Computern ermöglicht, die Fähigkeiten des Dienstes zu entdecken und zu verstehen, ohne auf den Quellcode, die Dokumentation oder die Überprüfung des Netzwerkverkehrs zugreifen zu müssen [57]. Wenn sie korrekt definiert ist, kann ein Verbraucher den entfernten Dienst mit einem Minimum an Implementierungslogik verstehen und mit ihm interagieren [57].

5.6.2 Architektur

Abbildung 5.13 zeigt die Architektur der Spring-Boot Anwendung. Für die Spring-Boot Anwendung wird eine 3-Layer-Architektur genutzt; Controller-Layer, Service-Layer und DAO- oder Repository-Layer. Im Controller-Layer wird die REST-API entwickelt und wird deshalb auch API-Layer genannt. In Controller-Klassen (Java Klassen) werden diese definiert. Im Service-Layer wird die Business-Logik erstellt und wird daher auch Business-Logik-Layer genannt. Im DAO-Layer wird die Datenbank bezogene Logik festgehalten und ist verantwortlich, mit der Datenbank zu kommunizieren. Ein Client fragt die Anwendung durch den API-Layer an. Mithilfe von *DTOs* werden Daten zwischen Client und Anwendung ausgetauscht. Das Medien-Typ-Format ist JSON.



Abbildung 5.13: Diagramm Spring-Boot Architektur

Abbildung 5.14 zeigt das UML-Komponentendiagramm der Spring-Boot Anwendung dieser Thesis. UML-Komponentendiagramme dienen der Modellierung der physischen Aspekte objektorientierter Systeme, die zur Visualisierung, Spezifizierung und Dokumentation komponentenbasierter Systeme [19] benutzt wurden. Komponentendiagramme sind im Wesentlichen Klassendiagramme, die sich auf die Komponenten eines Systems konzentrieren und häufig zur Modellierung der statischen Implementierungssicht eines Systems verwendet werden [19].

Das Backend (Spring-Boot-Backend) der Anwendung verfügt über eine Beziehung (engl. relationship) mit 3 Artefakten (engl. artifacts). Artifacts sind physische Entitäten, die in diesem Fall lokal auf dem Host-Gerät gespeichert sind [18]. Alternatives, Questions und Blockchain-Types sind vom Typ «Artifact» genauer vom Typ «File». Diese Dateien werden nicht über einen Port oder eine Schnittstelle hinzugefügt.

Airtable ist ein externes Tool, dass sich nicht im gleichen System wie die anderen Komponenten befindet. Die Liste von Blockchains kann durch den API-Zugang (auch provided interface genannt [19]) des Backends an die Anwendung bzw. die Backend-Komponente gegeben werden.

Die Backend-Komponente stellt für das Anfragen der in der Datenbank gespeicherten Daten ein Interface für das Angular-Frontend. Die Angular-Komponente kommuniziert

lediglich mit der Backend-Komponente und benötigt keine weiteren Beziehungen, da alle Funktionalitäten über die Backend-Komponente abgedeckt sind.

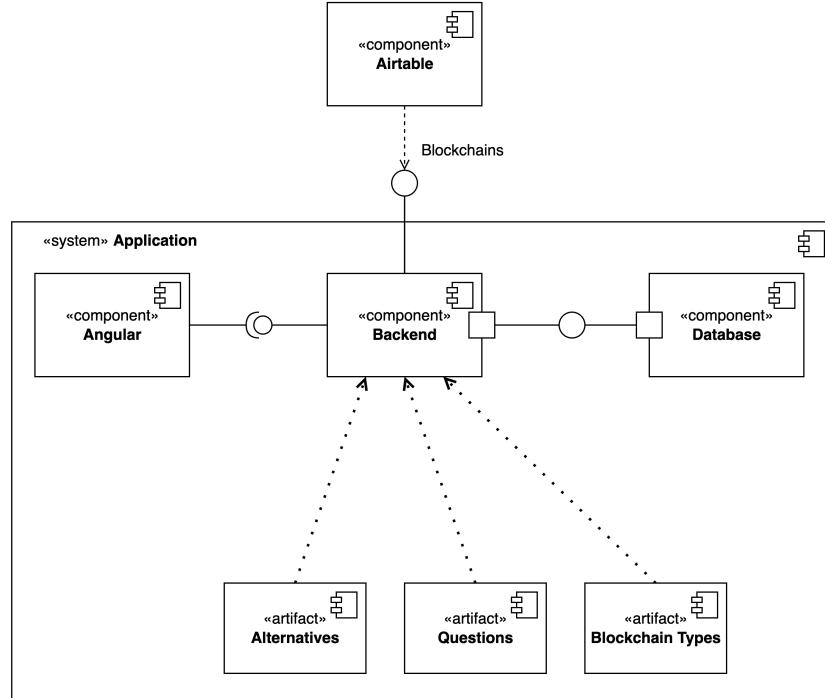


Abbildung 5.14: Komponenten-Diagramm der Anwendung

5.6.3 REST-API Endpunkte

Über der URL `localhost:9123/swagger-ui/index.html` stellt das OpenAPI-Framework eine Benutzeroberfläche zum Verwalten der REST-API Endpunkte zur Verfügung. MongoDB gehört zu den NoSQL Datenbanken und wurde bereits in Abschnitt 2.1.2 beschrieben. Das Datenbankschema wurde ebenfalls im Kapitel der Konzeptionierung 4.1.1 beschrieben. Mongo-Express ist eine webbasierte MongoDB Verwaltungsoberfläche. Es läuft ebenfalls in einem separaten Docker-Container. Beim Start des MongoDB Docker-Containers werden aus Dateien, die im JSON-Format gespeichert sind, Einträge für die Datenbank ausgelesen und dann in der Datenbank gespeichert. Dazu gehören die Alternativen der Blockchain, die Liste der Blockchain und der Fragenkatalog. Die Blockchains werden durch ein Skript mit den in Airtable angelegten Blockchains synchronisiert, weshalb beim Stoppen der Datenbank die Liste der Blockchains wieder gelöscht wird und beim nächsten Start erneut abgefragt wird. Abbildung 5.15 zeigt das Sequenzdiagramm

der Datenbank.

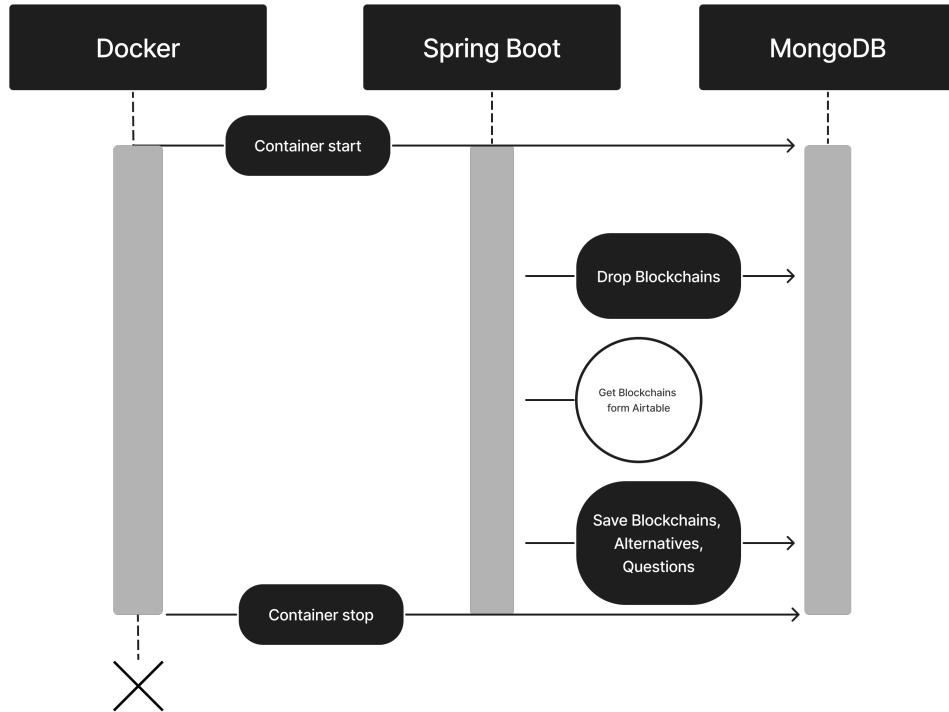


Abbildung 5.15: Sequenzdiagramm des Backends, der Datenbank und Docker

Die Funktion `importBlockchains` wird beim Start des Spring-Boot-Servers ausgeführt und fragt die Rest-API Schnittstelle des Tools Airtable ab und updatet mit den empfangenen Daten die JSON-Dateien, die lokal im Projekt liegen. Sie erzeugt aus den Daten Instanzen der Klasse `BlockchainDTO`, die das gleiche Format, wie die Entität `Blockchain` aus dem Datenbankschema, haben. Codeausschnitt 5.5 zeigt die in Java geschriebene Funktion.

Codeausschnitt 5.5: Funktion zum Importieren der Blockchains

```

public static List<BlockchainDto> importBlockchain() {
    GsonBuilder gsonBuilder = new GsonBuilder();
    HttpResponse<String> response = null;
    
```

```
// set authorization header
HttpRequest request = HttpRequest
    .newBuilder()
    .GET()
    .uri(URI.create("https://api.airtable.com/v0/..."))
    .setHeader("Authorization", "Bearer k32s*****")
    .build();

// try to send get-request, handle exceptions
try {
    response = httpClient.send(request,
        HttpResponse.BodyHandlers.ofString());
} catch (IOException | InterruptedException e) {
    e.printStackTrace();
}

// build JSON Objects from AirTable response
List<BlockchainDto> result = new ArrayList<>();
JSONObject obj = new JSONObject(response != null ? response.body() : null);
JSONArray records = obj.getJSONArray("records");
for (int i = 0; i < records.length(); i++) {

    // transform entries from attribute "field" to BlockchainDTO class
    result.add(gsonBuilder.create()
        .fromJson(records.getJSONObject(i)
            .getJSONObject("fields")
            .toString(), BlockchainDto.class));
}
return result;
}
```

6 — Zusammenfassung und Ausblick

Dieses Kapitel fasst abschließend die Ergebnisse der vorliegenden Arbeit zusammen. Neben der Ergebnisbetrachtung erfolgt auch eine kritische Bewertung der Ergebnisse. Des Weiteren wird im Ausblick auf nützliche Funktionalitäten und Features eingegangen, die der Anwendung noch hinzugefügt werden könnten.

6.1 Zusammenfassung

Das Ziel dieser Arbeit war die Konzeption und Realisierung einer ersten Version eines Frameworks zur Empfehlung der Blockchain-Technologie abhängig vom Anwendungsfall des Nutzers. Zu diesem Zweck wurden zunächst funktionale und nicht-funktionale Anforderungen (vgl. 3.1) an die verschiedenen Komponenten erhoben. Folgend wurde eine Web-Anwendung konzeptioniert. Dazu gehörten das Datenbank-Schema, der Blockchain-Vergleich und die Blockchain-Empfehlung in Form eines Fragenkatalogs mit selbst definierten Fragen anhand der erarbeiteten Erkenntnisse aus den Grundlagen der Blockchain-Technologie. Bei der Umsetzung des Entwurfs (vgl. Kapitel Implementierung und Ergebnisse 5) traten keine Probleme auf.

Anhand der Zielsetzung dieser Arbeit (vgl. 1.1) und den Anforderungen an die Anwendung (vgl. 3.1) kann die Aussage getroffen werden, dass beide Rahmenbedingungen erfüllt wurden. Es wurde eine in sich vollständiges Anwendung (Framework) inklusive Frontend-Applikation, Backend-Applikation und Datenbank entwickelt. Dieses kann bei Bedarf in einer Cloud oder auf einem Server gehostet werden, sodass Endkunden das Produkt mit einem öffentlichen Zugang nutzen können.

Weitere Ergebnisse Es existieren viele Blockchains, die allerdings durch die Priorität auf die Marktkapitalisierung oder zu wenig Aufmerksamkeit kurzlebig sein können. Nur wenige Blockchains z.B. Bitcoin oder Ethereum haben eine hohe Anzahl an aktiven Nutzern, die stetig steigt. Ein Ergebnis dieser Arbeit ist, dass bei der Auswahl der Blockchains eher auf die «populären» Blockchains gesetzt werden sollte. Ansonsten kann der Fall eintreten, dass ein Nutzer zwar die passende Blockchain findet, diese allerdings nach wenigen Monaten eingestellt wird. In diesem Fall hätte der Nutzer die Nachteile einer populären Blockchain (z.B. Transaktionen pro Sekunde) vernachlässigen sollen.

Des Weiteren existieren viele Alternativen zur Blockchain, die sich in ihren Funktionalitäten mit denen der Blockchain-Technologie überschneiden.

6.2 Ausblick

6.2.1 Technische Erweiterungen

Im Rahmen einer zukünftigen Arbeit könnte die Anwendung um die folgenden Funktionen erweitert werden.

Aufnahme weiterer Blockchains In der Liste von detailliert beschriebenen Blockchains befinden sich zum Zeitpunkt Oktober 2022 insgesamt 22 Blockchains. Diese wurden anhand ihres Whitepapers kategorisiert und ins Gesamtgeschehen eingeordnet. In der auf Airtable gehosteten Liste befinden sich allerdings 117 Blockchains. Für eine präzisere Empfehlung einer Blockchain-Technologie wäre es von Vorteil, dass weitere Blockchains den 22 bestehenden hinzugefügt werden. Dazu müssten die Whitepaper ebenfalls betrachtet werden. Im Zuge der Aufnahme der sog. Layer-1 Blockchain-Technologien könnten ebenfalls *Layer – 2* der Anwendung hinzugefügt werden. Diese haben grundlegende Unterschiede gegenüber den Layer-1 Blockchains [61].

Benutzeroberfläche Wie im Kapitel 5 beschrieben, existiert eine erweiterte Version der Benutzeroberfläche. Durch die lose Kopplung und die Struktur der Angular-Anwendung, muss anhand des Mockups neuer HTML- und CSS-Code (Tailwind-CSS Code) programmiert werden, der dann die bestehenden HTML- und CSS-Dateien der Anwendung ersetzt. Die Funktionen beider Benutzeroberflächen bleiben gleich.

Blockchain Wikipedia (Wiki) Ein Wiki könnte als zusätzliche Funktion der Anwendung hinzugefügt werden. Die nicht bekannten Begriffe sollen den Nutzern erklärt

werden. Ein Wiki (z.B. Blockchain Basics) beschreibt wie in einem Glossar die wichtigsten Begriffe rund um die Blockchain. Falls ein Nutzer einen Begriff nicht kennt, kann er diesen dort nachschlagen.

Sprache Die erste Implementierung der Anwendung war grundsätzlich auf Deutsch. Das «LanguageTool» ist eine weitere Funktion der Anwendung und könnte die Anwendung in beliebigen Sprachen anbieten. Durch eine Schaltfläche in der Navigationsleiste kann ein Nutzer die Sprache auswählen.

«i18next» ist ein internationalization-framework, das in und für JavaScript geschrieben wurde [3] und für das LanguageTool geeignet ist. Abbildung 6.1 zeigt die in Angular angelegte Ordnerstruktur. In der `de.json` Datei werden alle Inhalte der Web-Anwendung auf Deutsch abgespeichert und in der `en.json` auf Englisch.

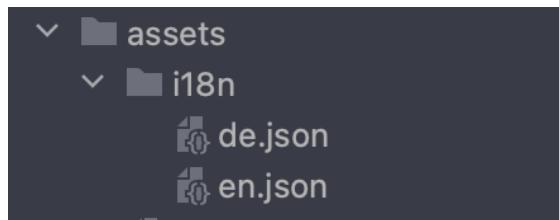


Abbildung 6.1: Ordnerstruktur Angular `assets`-Ordner

Die vom Nutzer ausgewählte Sprache der Anwendung, in diesem Fall Deutsch oder Englisch, wird im Speicher (sog. localstorage) des Browsers abgelegt. i18n liefert zusätzlich verschiedene Möglichkeiten, die beiden oben gezeigten Dateien einzubinden. Der Codeausschnitt 6.1 zeigt eine Möglichkeit. Innerhalb des HTML-Elementes `h4` wird der Text für das in den JSON-Dateien festgelegte Element 'test' eingebunden. Durch den Aufruf `| i18n` wird eine *Angular – Pipe* gestartet, die vom i18n-Framework definiert ist. Diese Pipe greift auf den Speicher des Browsers zu und ermittelt die ausgewählte Sprache und schaut dann entweder in der `de.json` oder `en.json` nach.

Codeausschnitt 6.1: Beispiel i18n Einbindung

```
<h4> {{ 'test' | i18n }} </h4>
```

Diese Funktion wurde in der Anwendung bereits implementiert. Es fehlen allerdings die Übersetzungen in den beiden JSON-Dateien.

Aufnahme von Blockchain «real-world» Projekten In einer Liste von Projekten, die auf der Blockchain-Technologie basieren, könnte ein Nutzer einen Überblick über bestehende Projekte bekommen. Die Website <https://www.stateofthedapps.com/dapps> ist ein Beispiel dafür, wie eine solche Seite aussehen könnte (siehe Abbildung 6.2).

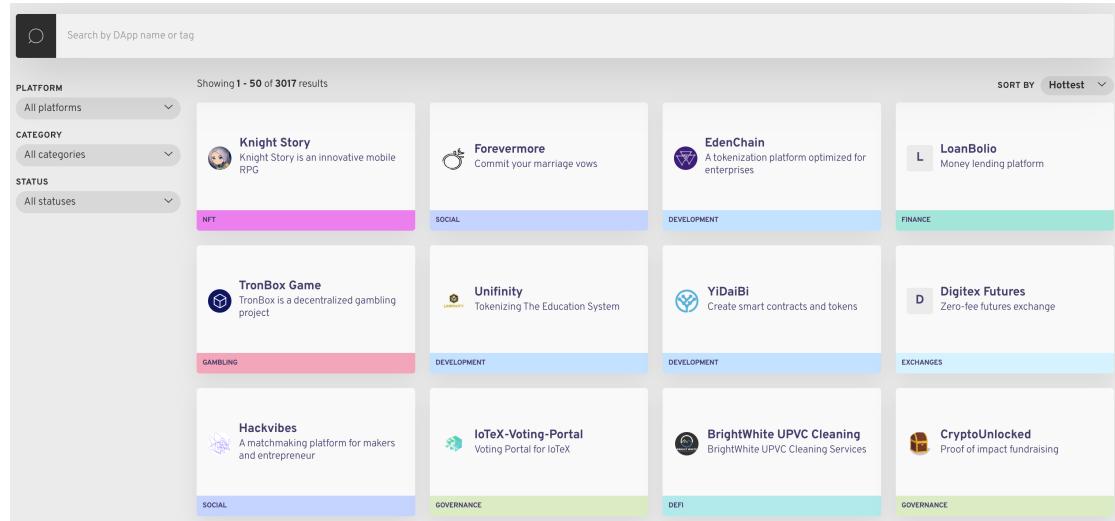


Abbildung 6.2: dApps auf der Seite [stateofthedapps.com](https://www.stateofthedapps.com)

6.2.2 Framework als Grundgerüst für weitere Projekte

Im Zuge der Implementierung dieser Anwendung sind auch andere Domänen für Empfehlungen möglich. Das Grundgerüst der Anwendung aus Frontend, Backend und Datenbank ist sehr generisch konzeptioniert, sodass es nach einigen Änderungen möglich wäre, z.B. Frontend-Technologien zu vergleichen anstatt Blockchains. In der Zukunft könnte die in dieser Thesis beschriebene Anwendung nur eine Kategorie der zu empfehlenden Technologien sein. Entwickler bzw. Nutzer der Anwendung könnten nicht nur Interesse haben herauszufinden, welche Blockchain für ihren Anwendungsfall Sinn ergibt, sondern welche Frontend- oder Backend-Technologie.

Weitere Vergleichs-Technologien könnten sein:

- Frontend-Technologien
- Backend-Technologien
- Design-Patterns

7 — Glossar

Kompilieren Das Kompilieren (engl. to compile) ist die Erstellung eines ausführbaren Programms aus einem in einer kompilierten Programmiersprache geschriebenen Codes. Die Kompilierung ermöglicht es dem Computer, das Programm auszuführen und zu verstehen, ohne dass die zur Erstellung verwendete Programmiersoftware benötigt wird [28].

Host Ein Host, auch als Internet-Knoten bezeichnet, ist ein Computer oder ein anderes Gerät, das an ein Computernetz angeschlossen ist. Ein Netzwerk-Host kann Benutzeranfragen bearbeiten, einschließlich des Angebots von Diensten, Softwareanwendungen und Informationsressourcen für Benutzer oder andere Knoten im Netz [29].

Deploy Die Softwarebereitstellung umfasst alle Schritte, Prozesse und Aktivitäten, die erforderlich sind, um ein Softwaresystem oder eine Aktualisierung für die vorgesehenen Benutzer verfügbar zu machen. Heute stellen die meisten IT-Organisationen und Softwareentwickler Software-Updates, Patches und neue Anwendungen mit einer Kombination aus manuellen und automatisierten Prozessen bereit [56].

SDK SDK steht für Software Development Kit und beschreibt eine Gruppe an Werkzeugen, die das Programmieren von mobilen Anwendungen ermöglichen [65].

DDoS-Attack Ein verteilter Denial-of-Service-(DDoS)-Angriff ist ein böswilliger Versuch, den normalen Traffic eines Zielservers, -dienstes oder -netzwerks zu stören, indem das Ziel oder die umliegende Infrastruktur mit einer Flut von Internet-Traffic überlastet wird [13].

Pattern Pattern oder Design Pattern beschreibt die Architektur eines Programmes.

Lightweight Lightweight ist eine entkoppelte und einfachere bzw. weniger komplexe Variante einer Anwendung im Bereich der Softwareentwicklung.

Technologie-Stack Ein Technologie-Stack, auch Solution-Stack, Technologieinfrastruktur oder Datenökosystem genannt, ist eine Liste aller Technologiedienste, die zum Erstellen und Ausführen einer einzelnen Anwendung verwendet werden. Die Seite Facebook zum Beispiel besteht aus einer Kombination von Codierungs-Frameworks und Sprachen, darunter JavaScript, HTML, CSS, PHP und ReactJS. Das ist der „Tech-Stack“ von Facebook [41].

RESTful Web Service Eine RESTful-API ist eine Schnittstelle mit der zwei Computersysteme Informationen auf sichere Weise über das Internet austauschen können. Die meisten Geschäftsanwendungen müssen mit anderen internen und externen Anwendungen kommunizieren, um verschiedene Aufgaben zu erfüllen [2].

Konsens Protokoll Der Zweck eines Konsens Protokolls (engl. consensus mechanism) besteht darin, einen Konsens zwischen den Teilnehmern darüber zu erzielen, was eine Blockchain zu einem bestimmten Zeitpunkt enthalten soll (einschließlich neuer Blöcke). Zu den Begriffen, die zur Beschreibung von Konsens Protokollen im Zusammenhang mit Blockchain-Technologien verwendet werden, gehören *proof of work* oder *proof of stake* [33].

Microservice Die Microservice-Architektur ist ein Ansatz, bei dem eine Anwendung aus mehreren kleinen, lose gekoppelten Services bereitgestellt wird. Dabei übernimmt jeder Service eine andere Kernfunktion der Funktionalität der Anwendung. Hingegen der monolithischen Software-Architektur werden Microservices unabhängig voneinander entwickelt und bereitgestellt. Jeder Service stellt dabei einen eigenen Prozess dar. Datenbanken oder Datenmodelle werden nicht untereinander aufgeteilt, jeder Microservice verfügt und verwaltet seine eigene Datenbank und Daten selbst. [50].

CI/CD CI/CD ist eine Methode zur Bereitstellung von Anwendungen für Kunden durch die Einführung von Automatisierung in den Phasen der Anwendungsentwicklung. Die wichtigsten Konzepte, die CI/CD zugeschrieben werden, sind continuous integration (CI), continuous delivery und continuous deployment. CI/CD ist eine Lösung für die Probleme, die die Integration neuen Codes für Entwicklungs- und Betriebsteams verursachen kann (auch bekannt als integration hell) [48].

Observable Angular verwendet Observables als Schnittstelle, um eine Vielzahl von üblichen asynchronen Operationen zu behandeln. Zum Beispiel: Das HTTP-Modul verwendet Observables, um AJAX-Anfragen und -Antworten zu verarbeiten. Die Module Router und Forms verwenden Observables, um auf Benutzereingabeereignisse zu warten und darauf zu reagieren [5].

Data Transfer Model (DTO) In der Programmierung ist ein Datentransferobjekt (DTO) ein Objekt, das Daten zwischen Prozessen transportiert wird. Der Grund für seine Verwendung liegt darin, dass die Kommunikation zwischen Prozessen in der Regel über entfernte Schnittstellen (z. B. Webdienste) erfolgt, bei denen jeder Aufruf eine kostspielige Operation darstellt [63].

Second Layer Blockchains Erlauben Skalierbarkeit und erhöhten Throughput (Transaktions-Geschwindigkeit) während die Integrität der Ethereum Blockchain sichergestellt wird. Das heißt Dezentralisierung, Transparenz und Sicherheit bestehen und gleichzeitig wird der Fußabdruck verringert (weniger Gas bedeutet weniger Energieverbrauch, was wiederum weniger Kohlenstoff bedeutet). Das Ethereum Hauptnetz ist einer der mehrheitlich genutzten Chains, aber hat dennoch einige Nachteile. Der Throughput ist sehr langsam (13 Transaktionen pro Sekunde) und die „Gas“ Gebühren sind hoch. Layer-2s basieren auf der Ethereum-Blockchain und sorgen für sichere, schnelle und skalierbare Transaktionen [61].

Angular-Pipe Pipes sind einfache Funktionen, die als Template verwendet werden können, um einen Eingabewert zu akzeptieren und einen transformierten Wert zurückzugeben. Pipes sind nützlich, weil sie in der gesamten Anwendung verwendet werden können, wobei jede Pipe nur einmal deklariert werden muss; zum Beispiel eine Pipe, um ein Datum als 15. April 1988 und nicht im rohen String-Format anzuzeigen [4].

Abbildungsverzeichnis

2.1	Hash-Funktion auf der Ethereum Blockchain	13
2.2	Transaktionszusammensetzung auf der Ethereum Blockchain	15
2.3	Vergleich Blockchain-Typen und Zentrale Datenbank [23]	18
4.1	Datenbank-Schema des Frameworks	26
4.2	Ausschnitt der vollständigen Blockchain-Liste mit 117 Blockchains auf der Plattform Airtable [1]	28
4.3	Aktivitätsdiagramm der Blockchain-Empfehlung	31
5.1	Pipeline des Frameworks in GitLab	35
5.2	Diagramm für den Ablauf des Docker Prozesses	37
5.3	Struktur des Frontend-Projektes	39
5.4	Komponenten im Angular-Frontend	41
5.5	Geteilte Komponenten (sog. shared components) im Angular-Frontend . .	41
5.6	Services im Frontend	42
5.7	Homescreen Prototyp	43
5.8	Homescreen erweiterte Version	44
5.9	Fragenkatalog der Prototyp-Version	45
5.10	Fragenkatalog erweiterte Version	45
5.11	Blockchain-Liste Prototyp-Version	46
5.12	Blockchain-Liste erweiterte Version	46
5.13	Diagramm Spring-Boot Architektur	48
5.14	Komponenten-Diagramm der Anwendung	49
5.15	Sequenzdiagramm des Backends, der Datenbank und Docker	50
6.1	Ordnerstruktur Angular assets -Ordner	54
6.2	dApps auf der Seite stateofthedapps.com	55

Quellcodeverzeichnis

2.1	TypeScript Interface für Alerts	10
5.1	Inhalt der Docker-Compose.yaml Datei	38
5.2	Tailwind-CSS Beispiel [58]	40
5.3	blockchain.service.ts Ausschnitt	42
5.4	Codeausschnitt AlternativRepository	47
5.5	Funktion zum Importieren der Blockchains	50
6.1	Beispiel i18n Einbindung	54
8.1	Inhalt der Docker-Compose.yaml Datei	66

Literatur

- [1] Airtable. *Blockchain List*. 2022. URL: <https://airtable.com/shrIq8Y61VTI5BMBW/tbl07naRWlDijfSep> (besucht am 31.08.2022).
- [2] Amazon. *What is Restful Api?* 2022. URL: <https://aws.amazon.com/de/what-is/restful-api/> (besucht am 31.08.2022).
- [3] Angular. *Angular Internationalization*. 2022. URL: <https://angular.io/guide/i18n-overview>.
- [4] Angular. *Angular Pipes*. 2022. URL: <https://angular.io/guide/pipes>.
- [5] Angular. *Observable*. 2022. URL: <https://angular.io/guide/observables-in-angular>.
- [6] Marten Risius Asger Balle Pedersen. *A Ten-Step Decision Path to Determine When to Use Blockchain Technologies*. 2019. URL: <https://fantom.foundation/fantom-faq/#problem> (besucht am 31.08.2022).
- [7] Binance. *Latency*. 2022. URL: <https://academy.binance.com/en/glossary/latency> (besucht am 31.08.2022).
- [8] Bitpanda. *Das Problem der Skalierbarkeit des Bitcoin Netzwerkes*. 2022. URL: <https://www.bitpanda.com/academy/de/lektionen/das-problem-der-skalierbarkeit-des-bitcoin-netzwerks/> (besucht am 31.08.2022).
- [9] *Blockchain Technology Overview*. 2018. URL: <https://nvlpubs.nist.gov/nistpubs/ir/2018/nist.ir.8202.pdf> (besucht am 31.08.2022).
- [10] 101 Blockchains. *Blockchain vs Relational Database*. 2021. URL: <https://101blockchains.com/blockchain-vs-relational-database> (besucht am 31.08.2022).
- [11] Coin Market Cap. *Throughput*. 2022. URL: <https://coinmarketcap.com/alexandria/glossary/throughput> (besucht am 31.08.2022).
- [12] Scott Chacon und Ben Straub. *Pro git*. Springer Nature, 2014.
- [13] Cloudflare. *What is DDOS Attack?* 2022. URL: <https://www.cloudflare.com/de-de/learning/ddos/what-is-a-ddos-attack/> (besucht am 31.08.2022).

LITERATUR

- [14] CodingTheSmartWay.com. *Tailwind CSS for Beginners*. 2020. URL: https://www.youtube.com/watch?v=j_5-LISy9Qg&feature=emb_imp_woyt.
- [15] Datadog-Knowledge-Center. *Serverless Architecture Overview*. URL: <https://www.datadoghq.com/knowledge-center/serverless-architecture/> (besucht am 31.08.2022).
- [16] Deloitte. *Evolution of blockchain technology*. 2017. URL: <https://www2.deloitte.com/us/en/insights/industry/financial-services/evolution-of-blockchain-github-platform.html>.
- [17] Iryna Deremuk. *Modern Web Application Architecture Explained: Components, Best Practices and More*. 2021. URL: <https://litslink.com/blog/web-application-architecture> (besucht am 31.08.2022).
- [18] Kirill Fakhroutdinov. *UML Artifact*. 2016. URL: <https://www.uml-diagrams.org/artifact.html>.
- [19] Kirill Fakhroutdinov. *UML Component Diagrams*. 2016. URL: <https://www.uml-diagrams.org/component-diagrams.html>.
- [20] Fantom. *Blockchain Trilemma*. 2022. URL: <https://fantom.foundation/fantom-faq/#problem> (besucht am 31.08.2022).
- [21] Fintech. *DLT Blockchain*. 2022. URL: https://www.bafin.de/DE/Aufsicht/FinTech2/InnovativeFinanztechnologien/DLT_Blockchain/DLT_Blockchain_artikel.html (besucht am 31.08.2022).
- [22] Fortunly. *48 Blockchain Statistics 2022: Interesting Facts You Should Know*. 2022. URL: <https://fortunly.com/statistics/blockchain-statistics/> (besucht am 31.08.2022).
- [23] Arthur Gervais. “Do you need a Blockchain?” In: (2019). URL: <https://eprint.iacr.org/2017/375.pdf> (besucht am 31.08.2022).
- [24] GitLab. *The One DevOps Platform GitLab*. 2022. URL: <https://about.gitlab.com/>.
- [25] Golden. *Blockchain List*. 2022. URL: <https://golden.com/> (besucht am 31.08.2022).
- [26] Google. *What is Angular*. 2022. URL: <https://angular.io/guide/what-is-angular> (besucht am 31.08.2022).
- [27] Hackr.io. *What is Web Application Architecture? Components, Models, and Types*. 2022. URL: <https://hackr.io/blog/web-application-architecture-definition-models-types-and-more> (besucht am 31.08.2022).
- [28] Computer Hope. *What is Compile?* 2022. URL: <https://www.computerhope.com/jargon/c/compile.htm> (besucht am 31.08.2022).

- [29] Computer Hope. *What is Host?* 2022. URL: <https://www.computerhope.com/jargon/h/hostcomp.htm> (besucht am 31.08.2022).
- [30] IBM. *MySQL vs. MongoDB: What's the Difference?* 2021. URL: <https://www.ibm.com/cloud/blog/mysql-vs-mongodb> (besucht am 31.08.2022).
- [31] IBM. *What is Java Spring Boot.* 2020. URL: [https://www.ibm.com/cloud/learn/java-spring-boot#:~:text=Java%20Spring%20Boot%20\(Spring%20Boot,ability%20to%20create%20standalone%20applications](https://www.ibm.com/cloud/learn/java-spring-boot#:~:text=Java%20Spring%20Boot%20(Spring%20Boot,ability%20to%20create%20standalone%20applications) (besucht am 31.08.2022).
- [32] ideas4development. *Bitcoin waste energy.* 2022. URL: <https://ideas4development.org/en/bitcoin-waste-energy/> (besucht am 31.08.2022).
- [33] IGI-global. *Consens Protocol.* 2022. URL: <https://www.igi-global.com/dictionary/consensus-protocol/53540> (besucht am 31.08.2022).
- [34] JetBrains. *IntelliJ IDEA Ultimate vs IntelliJ IDEA Community Edition.* 2022. URL: <https://www.jetbrains.com/products/compare/?product=idea&product=idea-ce>.
- [35] Tommy Koenen und Erik Poll. *Blockchain Alternatives.* 2022. URL: <https://blog.fasset.com/blockchain-alternative/>, <http://tommykoens.com/wp-content/uploads/2018/09/blockchain-alternative.pdf> (besucht am 31.08.2022).
- [36] Peter Koffer. *Website vs. Web Application - What's The Difference?* 2022. URL: <https://mdevelopers.com/blog/website-vs-web-application-what%27s-the-difference>.
- [37] Neil Kolban. *Gitlab Pipeline with Google Cloud.* 2021. URL: <https://medium.com/google-cloud/using-gitlab-and-cloud-build-to-achieve-ci-cd-for-cloud-run-4c6db26f04ed>.
- [38] Tiana Laurence. *Blockchain for dummies.* John Wiley & Sons, 2019.
- [39] Xiaowei Li und Yuan Xue. "A survey on web application security". In: *Nashville, TN USA* 25.5 (2011), S. 1–14.
- [40] Thomas McGovern. *HOW MANY BLOCKCHAINS ARE THERE IN 2022?* 2022. URL: <https://earthweb.com/how-many-blockchains-are-there/>.
- [41] Mixpanel. *What is a technology stack?* 2022. URL: <https://mixpanel.com/blog/what-is-a-technology-stack/> (besucht am 31.08.2022).
- [42] Anton Moiseev und Yakov Fain. *Angular Development with TypeScript.* Simon und Schuster, 2018.
- [43] MongoDB. *What is MongoDB?* 2022. URL: <https://www.mongodb.com/de-de/what-is-mongodb> (besucht am 31.08.2022).

LITERATUR

- [44] Multichain. *Avoiding pointless Blockchain Project*. 2022. URL: <https://www.multichain.com/blog/2015/11/avoiding-pointless-blockchain-project/> (besucht am 31.08.2022).
- [45] Satoshi Nakamoto. "Bitcoin: A peer-to-peer electronic cash system". In: *Decentralized Business Review* (2008), S. 21260.
- [46] Jonas Groß Philipp Sandner. *Blockchain, IoT und KI – eine vielversprechende Mischung*. 2020. URL: <https://www.capital.de/wirtschaft-politik/blockchain-iot-und-ki-eine-vielversprechende-mischung>.
- [47] QueryDSL. *QueryDSL*. 2022. URL: <https://github.com/querydsl/querydsl>.
- [48] Redhat. *What is CI/CD?* 2022. URL: <https://www.redhat.com/en/topics/devops/what-is-ci-cd> (besucht am 31.08.2022).
- [49] Sana. *Was ist eine Single Page Application (SPA)?* URL: <https://www.sana-commerce.de/ecommerce-erklaert/was-ist-eine-single-page-application/> (besucht am 31.08.2022).
- [50] Torben Schuhmacher. "Konzeption und prototypische Realisierung einer Microservice-Architektur mit Cloud-native Technologien". In: (2021). URL: https://pcarstensen.github.io/Bachelorarbeit_Torben-Schuhmacher.pdf (besucht am 31.08.2022).
- [51] Spring. *Spring Framework Overview*. 2022. URL: <https://docs.spring.io/spring-framework/docs/current/reference/html/overview.html#overview> (besucht am 31.08.2022).
- [52] Statista. *Börsenwert von Apple in den Jahren 2001 bis 2021*. 2022. URL: <https://de.statista.com/statistik/daten/studie/219902/umfrage/marktkapitalisierung-von-apples/>.
- [53] Statista. *Most Popular Backend Frameworks – 2012/2022*. 2022. URL: <https://statisticsanddata.org/data/most-popular-backend-frameworks-2012-2022/> (besucht am 31.08.2022).
- [54] statista. *Most Popular Frameworks*. 2022. URL: <https://www.statista.com/statistics/1124699/worldwide-developer-survey-most-used-frameworks-web/> (besucht am 31.08.2022).
- [55] Statista. *Worldwide spending on blockchain solutions from 2017 to 2024*. 2022. URL: <https://www.statista.com/statistics/800426/worldwide-blockchain-solutions-spending/> (besucht am 31.08.2022).
- [56] Sumologic. *Software Development*. 2022. URL: <https://www.sumologic.com/glossary/software-deployment/> (besucht am 31.08.2022).
- [57] Swagger.io. *Open-API*. 2022. URL: <https://swagger.io/specification/>.
- [58] Tailwind-CSS. *Tailwind-CSS*. 2022. URL: <https://tailwindcss.com/>.

LITERATUR

- [59] Don Tapscott und Alex Tapscott. *Die Blockchain-Revolution: Wie die Technologie hinter Bitcoin nicht nur das Finanzsystem, sondern die ganze Welt verändert.* Plassen Verlag, 2016.
- [60] Nicklas Urban. "When Blockchain?" In: (2022). (Besucht am 31.08.2022).
- [61] Alex White-Gomez. *What are Layer 2 (L2) Solutions in the Blockchain and Why Are They Important?* 2022. URL: <https://www.one37m.com/nft/what-are-layer-2-solutions-and-why-are-they-important>.
- [62] Wikipedia. *A survey on web application security.* URL: <https://de.wikipedia.org/wiki/Webanwendung>.
- [63] Wikipedia. *DTO.* 2022. URL: https://en.wikipedia.org/wiki/Data_transfer_object.
- [64] Wikipedia. *Routing.* 2022. URL: <https://en.wikipedia.org/wiki/Routing>.
- [65] Wikipedia. *SDK.* 2022. URL: https://en.wikipedia.org/wiki/Software_development_kit (besucht am 31.08.2022).
- [66] Wikipedia. *User Story.* 2022. URL: https://de.wikipedia.org/wiki/User_Story.

8 — Anhang

Codeauschnitt 8.1: Inhalt der Docker-Compose.yaml Datei

```
<?xml version="1.0" encoding="UTF-8"?>
<project xmlns="http://maven.apache.org/POM/4.0.0"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://maven.apache.org/POM/4.0.0
    https://maven.apache.org/xsd/maven-4.0.0.xsd">
<modelVersion>4.0.0</modelVersion>
<parent>
  <groupId>de.nl.smartcontracts</groupId>
  <artifactId>block-commander-root</artifactId>
  <version>1</version>
  <relativePath>../pom.xml</relativePath>
</parent>

<artifactId>backend-service</artifactId>
<version>1</version>
<name>backend-service</name>
<description>Demo project for Spring Boot</description>
<properties>
  <java.version>17</java.version>
</properties>
<dependencies>
  <dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter</artifactId>
  </dependency>
  <dependency>
    <groupId>org.springdoc</groupId>
    <artifactId>springdoc--ui</artifactId>
```

```
<version>1.6.9</version>
</dependency>
<dependency>
    <groupId>com.google.code.gson</groupId>
    <artifactId>gson</artifactId>
    <version>2.9.0</version>
    <scope>provided</scope>
</dependency>
<dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-data-mongodb</artifactId>
</dependency>
<dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-test</artifactId>
    <scope>test</scope>
</dependency>
<dependency>
    <groupId>org.json</groupId>
    <artifactId>json</artifactId>
    <version>20220320</version>
</dependency>

<dependency>
    <groupId>org.projectlombok</groupId>
    <artifactId>lombok</artifactId>
    <version>1.18.24</version>
    <scope>provided</scope>
</dependency>
<dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-web</artifactId>
</dependency>
<dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot</artifactId>
    <version>2.7.0</version>
</dependency>
<dependency>
    <groupId>io.github.classgraph</groupId>
```

```
<artifactId>classgraph</artifactId>
<version>4.8.147</version>
</dependency>

<dependency>
    <groupId>com.querydsl</groupId>
    <artifactId>querydsl-mongodb</artifactId>
    <version>5.0.0</version>
</dependency>
<dependency>
    <groupId>com.querydsl</groupId>
    <artifactId>querydsl-apt</artifactId>
    <version>5.0.0</version>
</dependency>
<dependency>
    <groupId>com.querydsl</groupId>
    <artifactId>querydsl-core</artifactId>
    <version>5.0.0</version>
</dependency>
</dependencies>

<build>
    <plugins>
        <plugin>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-maven-plugin</artifactId>
        </plugin>
        <plugin>
            <groupId>com.mysema.maven</groupId>
            <artifactId>apt-maven-plugin</artifactId>
            <version>1.1.3</version>
            <executions>
                <execution>
                    <goals>
                        <goal>process</goal>
                    </goals>
                    <configuration>
                        <outputDirectory>target/
generated-sources/java</outputDirectory>
                        <processor>
```

KAPITEL 8. ANHANG

```
        org.springframework
        .data.mongodb.repository.support
        .MongoAnnotationProcessor
    </processor>
</configuration>
</execution>
</executions>
</plugin>
</plugins>
</build>

</project>
```
