h_da
HOCHSCHULE DARMSTADT
UNIVERSITY OF APPLIED SCIENCES

Hochschule Darmstadt

– Fachbereich Informatik–

Synergie von DLT und IOT: Anfoderungsanalyse und praktische Verprobung

Abschlussarbeit zur Erlangung des akademischen Grades

Master of Science (M.Sc.)

vorgelegt von

Sebastian Kanz

Matrikelnummer: 735176

Referent	:	Prof. Dr. Michael Braun
Korreferent	:	Prof. Dr. Martin Stiemerling

ERKLÄRUNG

Ich versichere hiermit, dass ich die vorliegende Arbeit selbstständig verfasst und keine anderen als die im Literaturverzeichnis angegebenen Quellen benutzt habe.

Alle Stellen, die wörtlich oder sinngemäß aus veröffentlichten oder noch nicht veröffentlichten Quellen entnommen sind, sind als solche kenntlich gemacht.

Die Zeichnungen oder Abbildungen in dieser Arbeit sind von mir selbst erstellt worden oder mit einem entsprechenden Quellennachweis versehen.

Diese Arbeit ist in gleicher oder ähnlicher Form noch bei keiner anderen Prüfungsbehörde eingereicht worden.

Darmstadt, 15. Oktober 2019

Sebastian Kanz

ABSTRACT

Short summary of the contents in English. Approximately one page...

BTW: A great guide by Kent Beck how to write good abstracts can be found here:

https://plg.uwaterloo.ca/~migod/research/beckOOPSLA.html

ZUSAMMENFASSUNG

Das Internet der Dinge (engl. IOT; Internet of Things) erhält immer mehr Einzug in das tägliche Leben. Smart-Home Lösungen, vernetzte und latenzempfindliche Connected-Cars oder die Vision einer Smart-City prägen die Forschungsarbeiten in den jeweiligen Bereichen. Ziel ist eine vollautomatische Machine-to-Machine (M2M) Abwicklung von Prozessen, um unseren Alltag zu automatisieren und zu vereinfachen. Dabei fallen eine Menge Daten an, die verarbeitet, übertragen und gespeichert werden müssen. Mit der Einführung des neuen Mobilfunk-Standards 5G und immer leistungsfähigeren Endgeräten sind Übertragung und Verarbeitung der Daten weitestgehend gesichert; bleibt die Frage offen, wo diese großen Datenmengen gespeichert und weiterprozessiert werden. Daneben erfreut sich das Thema Distributed-Ledger-Technology (DLT) immer größerer Beliebtheit: Es werden täglich neue Anwendungsfälle gefunden, die durch die verteilte Infrastruktur, der Trustless-Eigenschaft und der Dezentralität profitieren. Es bietet sich eine Untersuchung an, um zu überprüfen, inwieweit diese beiden noch recht jungen Technologien IOT und DLT Synergien besitzen und sich gegebenenfalls gegenseitig ergänzen können. Die vorliegende Masterarbeit evaluiert eine Auswahl etablierter DLTs anhand ihrer Tauglichkeit für den Einsatz im IOT-Umfeld mit Fokus auf den M2M-Bereich. Dazu wird zunächst ein IOT-Anwendungsfall erstellt, der stellvertretend für den M2M-Bereich für die weiteren Analysen verwendet wird. Anschließend werden konkrete Anforderungen aus verschiedenen Bereichen Infrastruktur, IT-Security, Performance und weiteren aufgestellt, die eine DLT erfüllen muss, um den Anforderungen des beispielhaften Anwendungsfalls gerecht zu werden. Die erstellten Kriterien werden auf eine Auswahl von DLT-Implementierungen angewandt, evaluiert und bewertet. Mit der am besten geeigneten DLT wird eine prototypische Implementierung des Anwendungsfalls vorgenommen, um die Ergebnisse aus der Anforderungsevaluierung zu überprüfen. Um den Use-Case möglichst realistisch zu simulieren werden Daten aus verschiedenen IOT-Sensoren an die DLT übermittelt und eine M2M-Komminkation zwischen IOT-Devices via DLT erstellt. Anschließende Load-Tests geben detaillierte Informationen über die Performance. Das Ergebnis ist eine strukturierte und nachvollziehbare Bewertung mehrerer, am Markt etablierter DLTs, inwieweit diese für DLT-sinnvolle IOT-Anwendungsfälle im M2M-Umfeld geeignet sind, sowie ein DLT-basierter Prototyp angelehnt an einen realen Use-Case, der beispielhaft als Nachweis der erarbeiteten Bewertung dient.

INHALTSVERZEICHNIS

ABBILDUNGSVERZEICHNIS

TABELLENVERZEICHNIS

LISTINGS

ABKÜRZUNGSVERZEICHNIS

DLT Distributed Ledger Technoligies

IOT Internet of Things

PoC Proof-of-Concept

API Application Programming Interface

OSI Open Systems Interconnection

DApp Distributed Application

BABOK Business Analysis Body of Knowledge

PMBOK Project Management Body of Knowledge

SWEBOK Software Engineering Body of Knowledge

SEBOK System Engineering Body of Knowledge

ISO International Organization for Standardization

IEEE Institute of Electrical and Electronics Engineers

IIBA International Institute of Business Analysis

PMI Project Management Institute

DoR Definition of Ready

DoD Definition of Done

UX User-Experience

SPoF Single-Point-of-Failure

EVM Ethereum Virtual Machine

W3C World Wide Web Consortium

DID Decentralized Identifier

VC Verifiable Credential

OSI Open Systems Interconnection

P2P Peer-to-Peer

Teil I

THESIS

1 EINLEITUNG

Das digitale Zahlungsmittel Bitcoin, ein für Internet of Things (IOT)-Anwendungen geschaffenes System wie IOTA oder ein komplexes System wie Ethereum, auf welchem sich umfangreiche Geschäftslogiken mittels Smart-Contracts umsetzen lassen: Die noch recht Junge Technologie hinter Blockchain-Lösungen - oder allgemeiner Distributed Ledger Technoligies (DLT)-Lösungen - greift auf grundlegende Ansätze zurück, die keineswegs neu sind:
Vor über 1500 Jahren suchten die Einwohner der Insel Yap im Westpazifik östlich der Philippinen eine Lösung für ihr Zahlungsmittel-Problem. Ihre Währung - Rai - waren großen runde Steintafeln, die teils mehrere Tonnen wiegen konnten und als alltägliches Zahlungsmittel ungeeignet waren. Die Dorfältesten merkten sich, wem welcher Rei-Stein gehörte. Fand eine Transaktion statt, so wurde diese den Dorfältesten mitgeteilt. Das Wissen, wem welcher Stein gehörte und wer von wem etwas gekauft hatte, war über mehrere Köpfe verteilt; die dezentrale Informationsspeicherung war geboren [6].
Schaut man in die jüngere Vergangenheit, so lässt sich die historische Entwicklung fortführen: Hash-Bäumeauch Merkle-Trees genannt, wurden 1979 von Ralph Merkle erfunden und sind heute eine der grundlegenden Bauteile einer Blockchain. 1991 beschrieben die Forscher Stuart Haber und W. Scott Stornetta die Idee hinter der Blockchain-Technologie [3]. 2008 veröffentlichte Satoshi Nakamoto das Whitepaper zu Bitcoin, dicht gefolgt von Ethereum, welches 2013 von Vitalik Buterin erstmals vorgestellt wurde.
Und wie sieht es heute aus? Mittlerweile ist ein ganzes Ökosystem an Blockchain- und DLT-Lösungen entstanden; die Website coinmarketcap.com listet beinahe 5000 Kryptowährungen mit einer Marktkapitalisierung von mehr als 200 Milliarden US-Dollar (Stand: 12/2019). Mittels sogenannter Smart-Contracts lassen sich komplexe Geschäftslogiken auf der Blockchain umsetzen und kombinieren, sodass ganze Anwendungen auf Blockchain-Basis erstellt werden können. Solche Distributed Application (DApp)s sind ein Schritt in die Zukunft hin zum Web 3.0 - auch Semantic Web genannt. Dabei handelt es sich um die nächste Evolutionsstufe des World-Wide-Webs, in dem Maschinen und Programme in der Lage sind, Anfragen und Interaktionen mit Menschen eine Semantik zuzuordnen. Das ist dann der Moment, indem die Kaffeemaschine den Kaffeekonsum des Benutzers analysiert und morgens den Kaffee selbstständig serviert. Produkte werden automatisch nachbestellt, da die Maschine weiß, wann der Kaffee aufgebraucht sein wird. Und wenn Besuch kommt, dann kann die Maschine den kalkulierten Verbrauch automatisch anpassen. Diese Zukunft wird möglich gemacht durch die Umsetzung des Web3.0 - unterstützt durch die Blockchain-Technologie und dezentrale Anwendungen.

1.1 MOTIVATION & PROBLEMSTELLUNG

Das Telekommunikationsunternehmen Cisco prognostiziert, dass bis zum Jahr 2030 mehr als 500 Milliarden mit dem Internet verbundene IOT-Geräte in verschiedenen Bereichen unseres alltäglichen Lebens Einzug erhalten haben werden [5]. Vernetzte Dinge unseres Alltags wie Kühlschranke, Kaffeemaschinen, aber auch die aus dem Business-Umfeld automatisierte Supply-Chain oder eine Smart-City sind nur einige wenige Beispiele dieses Geschäftsfeldes. Das Konzept von IOT ist nach wie vor sehr theoretisch, obwohl bereits einige Anwendungsfälle erarbeitet wurden. Um das große Potential von IOT vollumfänglich nutzbar zu machen und entsprechende Visionen umzusetzen, muss eine passende Backbone-Lösung für solche Anwendungsfälle bereitgestellt werden. Viele verschiedene Hersteller und Service-Provider benötigen eine einheitliche Plattform, auf der sie ihre IOT-Geräte, Services, Geschäftslogiken und Kunden miteinander vernetzen können sowie die Integration eines sicheren Bezahlsystems. Es stellt sich die Frage, ob und inwiefern die zwei innovativen Technologien DLT und IOT voneinander profitieren können und ob sich DLT als skalierende, performante und sichere Backbone-Technologie für IOT-Anwendungsfälle eignet.

1.2 ZIELSETZUNG & ZIELGRUPPE

Das Ziel dieser Arbeit ist die Untersuchung und prototypische Verprobung der folgenden These: 'DLT eignet sich als Backbone-Technologie für IOT und die technischen / nicht-funktionalen Anforderungen sind für alle Anwendungsfälle gleich'. Es wird gezeigt, inwieweit sich die Technologie DLT als Backbone-System für IOT-Anwendungsfälle eignet, welche Anforderungen dafür erfüllt sein müssen, und welche Implementierung für die Umsetzung in Frage kommt. Dazu im Verlauf der Arbeit das zu lösende Problem genauer spezifiziert und herausgearbeitet: Nachdem ein konkreter IOT-Anwendungsfall vorgestellt wurde und alle DLT-relevanten Anforderungen ermittelt und evaluiert sind, ergibt sich das konkrete Problem als die Umsetzung eines speziellen IOT-Anwendungsfalls mit einer konkreten DLT-Lösung. Der explizite Lösungsraum, also welche Anforderungen umgesetzt werden sollen, wird zuvor genau beschrieben. Das Problem gilt als gelöst, sobald die zuvor abgeleiteten Anforderungen mit dem Proof-of-Concept (PoC) erfüllt werden können. Abschließend wird die eingangs formulierte These diskutiert und evaluiert.
Diese Arbeit richtet sich an IT-Spezialisten aus dem Umfeld DLT und IOT, die sich über die Synergie beider Konzepte informieren, sowie Fachleuten aus der Industrie, die entsprechende IOT-Anwendungsfälle ausarbeiten möchten. Ein solides Grundverständnis für die grundlegenden Konzepte und Wordings wird an dieser Stelle vorausgesetzt; auf entsprechende Grundlagenliteratur wird gegebenenfalls verwiesen. In Kapitel 2 wird das benötigte Basiswissen vermittelt, welches für das Verständnis dieser Arbeit und der Zusammenhäng von Nöten ist.

1.3 METHODIK & VORGEHEN

In dieser Arbeit werden die Themenbereiche 'DLT' und 'IOT' vorgestellt, klassifiziert und in das Open Systems Interconnection (OSI) Referenzmodell eingeordnet. Die Synergie beider Bereiche wird herausgearbeitet und es wird dem Leser vorgestellt, wie diese Technologien voneinander profitieren können. Ein beispielhafter IOT-Anwendungsfall wird entwickelt und eine detaillierte Auflistung aller Anforderungen erarbeitet. Im nächsten Schritt werden die die ermittelten Anforderungen schrittweise auf eine Untermenge von fundamentalen Anforderungen reduziert, die relevant für IOT in Verbindung mit DLT sind. Mehrere, am Markt etablierte DLT Anwendungen werden anschließend vorgestellt und auf Basis dieser Anforderungsuntermenge evaluiert. Es wird geprüft, ob und inwieweit sie sich als Backbone-Lösung für den IOT-Anwendungsfall qualifizieren. Die vielversprechendste Lösung wird in einem PoC prototypisch umgesetzt, um die Anforderungsliste zu evaluieren. Es wird gezeigt, dass die gewählte DLT zielbringend als IOT Backbone-Lösung eingesetzt werden kann. Abschließend wird diskutiert, dass die nicht-funktionalen Anforderungen für DLT-geeignete IOT-Anwendungsfälle, unabhängig vom tatsächlichen Anwendungsfall selbst, stets die gleichen sind.

Diese Arbeit zeigt die Eignung von verschiedenen DLTs als Backbone-Lösung für IOT-Anwendungsfälle anhand eines beispielhaften PoCs. Es werden nur solche Bereiche von IOT betrachtet, die auch grundsätzlich für die Implementierung auf DLTs geeignet sind. Es gibt darüber hinaus weitere Bereiche, die sich nicht eignen, um auf DLTs umgesetzt zu werden und müssen auf einer anderen technologischen Basis implementiert werden. Des weiteren wird die in dieser Arbeit durchgeführte Analyse anhand eines PoC belegt. Aufgrund von Restriktionen wie der Zeitlimitierung und der praktischen Umsetzbarkeit könnten unter Umständen nicht alle fundamentalen Anforderungen gezeigt werden, die für einen IOT-DLT-Anwendungsfall erfüllt sein müssen.

1.4 ABGRENZUNG DIESER ARBEIT

todo

1.5 AUFBAU DIESER ARBEIT

todo

2 THEORETISCHE GRUNDLAGEN

Dieses Kapitel stellt dem Leser benötigtes Basiswissen zur Verfügung. Es wird ein grundlegendes Wissen zu den Themenbereichen DLT und IOT vermittelt und auf weitere Informationsquellen verwiesen, welche tiefergehende Informationen bereitstellen.

2.1 DLT

Die Begriffe Blockchain und Distributed Ledger Technology (kurz: DLT) werden häufig synonym verwendet, wobei es sich bei Blockchain um eine Unterklasse von DLT handelt. Beide bestehen aus einem dezentralen Netzwerk von Knoten, die Daten dezentral speichern, einen Konsens-Mechanismus zur Synchronisierung einsetzen und asymmetrische Verschlüsselungsverfahren zur Integrität und Absicherung des Systems nutzen. Hauptbestandteil eines Distributed Ledgers ist der Ledger selbst (deutsch: Kassenbuch), der eine Historie aller erfolgten Transaktionen speichert. Eine Transaktion beinhaltet einen Sender, einen Empfänger und eine Anzahl Einheiten, die versendet werden sollen. Je nach Implementierung können weitere Inhalte dazukommen.
Die Blockchain als Unterklasse der DLTs hat ihren Namen aufgrund der Beschaffenheit des Ledgers: Transaktionen werden in sogenannten Blöcken gruppiert und gespeichert. Jeder Block hat eine Referent auf den vorherigen Block, indem er dessen kryptografisch berechneten Hashwert speichert. Damit ergibt sich eine einfach verkettete Liste (Chain) von Blöcken (Block), die den gesamten Ledger repräsentiert. Ein Block wird von einem oder mehreren Knoten, die am Blockchain-Netzwerk teilnehmen, erzeugt und mittels eines Konsensalgorithmus durch alle beteiligten Knoten vor Manipulation geschützt. Eine Auflistung mehrere bekannter Konsensmechanismen ist der Tabelle **ABC** zu entnehmen.
ausführlicher

2.1.1 *Taxonomie von Blockchains*

Blockchains können klassifiziert werden, indem man die Sichtbarkeit von Informationen sowie die aktive Teilnahme von Netzwerkknoten am Konsensmechanismus betrachtet. Ist die Blockchain öffentlich erreichbar und können Transaktionen und Blöcke von beliebigen Knoten eingesehen werden, so handelt es sich um eine öffentliche (Public) Blockchain. Ist der Zugang zum Blockchain-Netzwerk beschränkt und können nur berechtigte Knoten Transaktionen und Blöcke einsehen, so spricht man von einer privaten (Private) Blockchain. Darüber hinaus lassen sich Blockchains über die Teilnahme

am Konsensalgorithmus, also der Entscheidung, welche Transaktionen in die Blockchain übernommen werden, klassifizieren: Ist jeder Netzwerkknoten, der die Blockchain erreichen und damit Informationen einsehen kann (unabhängig von public oder private) in der Lage, am Konsensverfahren teilzunehmen, so spricht man von einer beschränkungslose (Permissionless) Blockchain. Ist das Konsensverfahren auf priviligierte Knoten beschränkt, so handelt es sich um eine zugangsbeschränkte (Permissioned) Blockchain. Die vorgestellte Klassifizierung in Public, Private, Permissioned und Permissionless lassen sich darüber hinaus kombinieren, wodurch es vier mögliche Arten von Blockchains gibt. Tabelle **XYZ** stellt diese übersichtlich dar und listet dazu einige bekannte Beispiele auf. Public Permissionless: Ethereum, Bitcoin
Public Permissioned: EOS, Ripple
Private Permissionless: Holochain
Private Permissioned: Hyperledger Fabric
[12] Die bekanntesten Vertreter Bitcoin und Ethereum sind öffentliche und beschränkungslose Blockchains, an denen jeder teilnehmen kann und alle Daten öffentlich einsehbar sind. Im Unternehmensumfeld wird häufig auf private Blockchains gesetzt, da hier die Zugangskontrolle zu gegebenenfalls sensiblen Daten gewährleistet wird und keine Geschäftsgeheimnisse nach Außen dringen können. Kooperationen und Zusammenarbeiten zwischen Unternehmen werden oftmals als Permissioned Blockchains umgesetzt, da hier jeder Kooperationspartner einen oder mehrere am Konsensverfahren teilnehmende Knoten besitzt.

2.1.2 *Hash-Funktionen*

todo

2.1.3 *Smart-Contracts*

Ein essentieller Bestandteil vieler Blockchains sind sogenannte Smart-Contracts. Dabei handelt es sich um Software-Code, der auf der Blockchain (önchain") auf allen Knoten ausgeführt wird und eine Zustandsänderung in Form von ausgehenden Transaktionen zur Folge hat. Smart-Contracts werden meist in einer abgeschirmten Laufzeit-Umgebung (engl.: Runtime-Environment) ausgeführt; im Falle von Ethereum spricht man von der Ethereum Virtual Machine (EVM). Um auf Daten und Informtionen außerhalb des Blockchain-Netzwerkes zugreifen zu können, können sich Smart-Contract sogenannter Oracle-Services bedienen. Dabei handelt es sich um Informationsanbieter, die zum Beispiel Wetterdaten, Lottozahlen oder Nahverkehrsinformationen an einen Smart-Contract senden, damit dieser, basierend auf den empfangenen Daten, seine Logik ausführen und die Daten verarbeiten kann. Der grundlegende Nachteil von Oracle-Services liegt in der Vertrauensfrage: Während eine Blockchain grundsätzlich mittels Konsensverfahrens die Vertrauensfrage beantwortet, obliegt dem Besitzer bzw. Betreiber eines Oracles

die Macht über die Richtigkeit der Daten. Dieser Problematik kann durch den Einsatz mehrerer, mittels eines Konsensverfahrens abgestimmter Oracle-Services entgegengewirkt werden.
Mit Smart-Contracts werden oftmals (allerdings nicht ausschließlich) Verträge wie Kauf- oder Mietverträge umgesetzt. Bei solchen Verträgen ist das jeweils geltende Recht eines Landes zu beachten: Die Form und der Aufbau sowie der abzubildende Inhalt eines Vertrages muss gewissen Normen entsprechen und alle benötigten Informationen enthalten, damit ein Vertrag rechtskräftig ist. Um nun ein solch komplexes Konstrukt mittels Computercode als Smart-Contract abzubilden, müssen einige Punkte beachtet werden, damit die Implementierung rechtlich bindend ist. Nach Deutschem Recht muss ein Vertrag **todo**

2.1.4 *Konsensalgorithmen*

Ein Blockchain-Netzwerk ist ein verteiltes System, welches aus vielen einzelnen Knoten besteht. Damit Transaktionen und Daten über das Gesamtsystem hinweg konsistent sind, bedarf es eines Mechanismus, welcher sicherstellt, dass alle Knoten die gleichen Daten halten. Solche Mechanismen werden durch Konsensalgorithmen beschrieben. Es wird ein Protokoll definiert, welches Regeln definiert, wie und welche Daten gespeichert und welche verworfen werden. Im Folgenden werden einige bekannte Konsensverfahren genannt und kurz vorgestellt.

PROOF-OF-WORK (POW)

PROOF-OF-STAKE (POS)

PRACTICAL BYZANTINE FAULT TOLERANCE (POW)

PROOF-OF-AUTHORITY (POA)

DELEGATED-PROOF-OF-STAKE (DPOS)

todo

2.1.5 *Dezentrale Identitäten*

Eine Identität zeichnet sich durch eine Menge von Informationen, Daten und Eigenschaften aus, die diese eindeutig identifizieren. Nur der Inhaber einer Identität kann mit dieser auch agieren, da er der einzige ist, der Zugriff auf die geschützten Informationen hat, die das Innehaben einer Identität bestätigen. Diese können zum Beispiel ein Passwort, eine Geburtsurkunde, ein Fingerabruck oder Ähnliches sein.
Eine dezentrale Identität ist eine neuartige Technologie, die es dem Inhaber einer solchen erlaubt, seine Identität digital, dezentral und sicher durch den Einsatz asymmetrischer Verschlüsselung selbst zu verwalten. Sie ist

kryptografisch verifizierbar und der Inhaber entscheidet selbst, welche Informationen er teilen möchte und welche nicht. Das World Wide Web Consortium (W3C) entwickelt aktuell (Stand: Dezember 2019) einen Industrie-Standard, der zur Verifizierung und Authentifizierung persönlicher Informationen vor Dritten im Web 3.0 eingesetzt werden soll, und sich derzeit in der Version 1.0 befindet [11]. Eine dezentrale Identität besteht aus einer Decentralized Identifier (DID), welche weltweit einzigartig ist, und einem dazugehörigen DID-Dokument, welches Informationen über den beschriebenen Gegenstand enthält. Die folgenden Beispiele zeigen eine DID und ein dazugehörige DID-Dokument (entnommen aus [11]).

Listing 2.1: Beispiel einer DID

```
did:example:123456789abcdefghi
```

Listing 2.2: Beispiel eines DID-Dokuments

```
{
  "@context": "https://www.w3.org/ns/did/v1",
  "id": "did:example:123456789abcdefghi",
  "authentication": [{

    "id": "did:example:123456789abcdefghi#keys-1",
    "type": "RsaVerificationKey2018",
    "controller": "did:example:123456789abcdefghi",
    "publicKeyPem": "-----BEGIN PUBLIC KEY...END PUBLIC KEY-----\r\n"
  }],
  "service": [{

    "id":"did:example:123456789abcdefghi#vcs",
    "type": "VerifiableCredentialService",
    "serviceEndpoint": "https://example.com/vc/"
  }]
}
```

Durch den Einsatz von Blockchain-Technologie können DIDs manipulationssicher, hochverfügbar und für jeden zugänglich gespeichert werden. Die DID besteht aus drei Teilen: Zunächst dem Schlüsselwort "did", welches beschreibt, dass es sich um eine DID handelt. Anschließend folgt die DID-Methode (im Beispiel: ëxample"), die definiert, wie die DID aufzulösen und weitere Informtionen zu dieser Identität zu finden sind. Der letzte Teil ist eine ID ("123456789abcdefghi"), die für jede Methode einzigartig ist und somit eindeutig ermittelt werden kann.
Darüber hinaus existieren sogenannte Verifiable Credential (VC), die von vertrauenwürdigen Instanzen einer DID ausgestellt werden können. Dabei handelt es sich um verifizierbare Berechtigungsnachweise. Der Aussteller bescheinigt dem Empfänger eine bestimmte Eigenschaft und stellt einen Service-Endpoint zur Verfügung, an dem ein Dritter diesen VC verifizieren kann. So kann zum Beispiel eine Universität mit ihrer eigenen DID "did:hda:123456789abcdefghiëine

Student "did:example:sebastiankanzëin VC ausstellen, welches dem Student bescheinigt, aktuell an der Universität eingeschrieben zu sein. Möchte sich der Student nun an der Universitätsbibliothek authentifizieren, so kann er dort das VC der Universität vorzeigen und bekommt Zugriff auf die Bibliotheksausleihe. Die Bibliothek kann das VC verifizieren, indem es unter der Methode "hdaünd ëxample"die Universität und den Student identifiziert und anschließend die Signaturen überprüft.

2.1.6 *Blockchain im Kontext des OSI-Referenzmodells*

Das OSI Modell gilt seit Mitte der 8oer-Jahre als Standard zur Einordnung von Netzwerkprotokollen. Es wurde von der International Organization for Standardization (ISO) entwickelt und besteht aus sieben Schichten [13]:

PHYSICAL LAYER

DATA LINK LAYER

NETWORK LAYER

TRANSPORT LAYER

SESSION LAYER

PRESENTATION LAYER

APPLICATION LAYER

Da es sich bei DLT - wie der Name bereits sagt - um ein verteiltes System handelt, welches über ein Netzwerk kommuniziert, liegt es nahe, eine Einordnung in das OSI Modell durchzuführen. Dazu werden die verschiedenen Bestandteile eines DLTs den einzelnen Schichten des OSI Modells zugeordnet. [2] Vorstellung Blockchain-Modell Zuordnung OSI-Modell zu Blockchain-Modell Definition Protokoll, Definition Kommunikationsprotokoll, Begründung warum Blockchain Kommunikationsprotokoll ist **todo**

Eine Blockchain ist ein verteiltes System und besteht aus einem Peer-to-Peer (P2P) Netzwerk der beteiligten Blockchain-Knoten.

Ein Kommunikationsprotokoll ist nach Beispiel HTTP

2.2 IOT

todo IOT - Industrial IOT, Consumer IOT

2.2.1 *Digitaler Zwilling*

todo

2.2.2 *Anwendungsbereiche im Kontext DLT*

todo IOT-Bereiche mit Anwendungsbeispielen - Echtzeitanwendungen,

3 VERWANDTE FORSCHUNGSARBEITEN

todo

4

ANWENDUNGSFALL: VERMIETUNG VON HAUSHALTSGERÄTEN NACH DEM PAY-AS-YOU-USE PRINZIP

Qualitativ sehr hochwertige Haushaltsgeräte und Geräte für den professionellen Einsatz im Gastronomie-Umfeld haben hohe Anschaffungskosten, die der Privatanwender oder der Inhaber eines kleinen Gewerbes oftmals nicht leisten kann. Ein professioneller Kaffeevollautomat, eine leistungsfähige Spülmaschine oder eine Waschmaschine, die für hohe Kapazitäten ausgelegt ist, können Anschaffungskosten im vier bis fünfstelligen Euro-Bereich haben[1]. Eine naheliegende Möglichkeit besteht hier bei der Nutzung von Anbietern, die Haushaltsgeräte für eine monatliche oder jährliche Gebühr vermieten. So gibt es beispielsweise Anbieter für Kaffeemaschinen wie Tchibo[2] oder Nespresso[3], die ihre Produkte direkt vermieten, oder Anbieter, die als Zwischenhändler fungieren und sich auf die Vermietung von Haushaltsgeräten verschiedener Hersteller spezialisiert haben. Dabei kommen klassische Miet- und Bezahlmodelle zum Einsatz, wobei es sich meistens um monatliche oder jährliche Mietgebühren handelt. Einen neuartigen Ansatz verfolgt das Unternehmen Winterhalter mit ihrem Pay-per-Wash[4] Ansatz. Hier bezahlt der Kunde keine monatliche Mietgebühr, sondern pro Waschgang; die Berechnung erfolgt also auf dem tatsächlichen Verbrauch des Kunden und nicht auf einer kalkulierten Pauschale.
Dieses Kapitel beschreibt einen IOT-Anwendungsfall, der die oben beschriebene Problematik aufgreift und das von der Firma Winterhalter eingeführte Pay-per-Wash Bezahlmodell einen Schritt weiterführt. Dabei interagieren verschiedene Stakeholder miteinander nach einem Pay-as-You-Use Prinzip auf einer einheitlichen Plattform.

4.1 BESCHREIBUNG

Kunden mieten Haushaltsgeräte (im Consumer-Bereich oder für den professionellen Einsatz) wie Kaffeemaschinen oder Waschmaschinen je nach Anieter zum Nulltarif von verschiedenen Herstellern, die ihre Geräte auf einer Plattform zur Miete anbieten. Der genaue Verbrauch (Anzahl Kaffees, Menge an gewaschener Wäsche, Wasserverbrauch, etc.) wird mittels integrierter Sensoren an den Geräten erfasst und auf der Plattform persistiert. Damit wird ein genaues, vom tatsächlichen Verbrauch abhängiges Abrechnungsmodelle umgesetzt: Kunden zahlen nur das, was sie auch wirklich verbrauchen. Regelmäßige Reinigungen seitens der Kunden werden erfasst

1 Quelle!
2 https://www.tchibo-coffeeservice.de/shop/kaffeevollautomaten/
3 https://www.nespresso.com/pro/de/de/kaffeemaschinen-buero
4 https://www.pay-per-wash.biz/ch_de/

und durch ein entsprechendes Rabattmodell verrechnet. Serviceleistungen wie Wartung und Reparatur durch entsprechende Dienstleister können über die zugrundeliegende Plattform geplant, gesteuert und abgerechnet werden. Die Lieferung von Geräten, Ersatzteilen und Konsumgütern wie Kaffee oder Waschmittel erfolgt durch Lieferanten. Die Bestellung und Abrechnung wird über die zugrundeliegende Plattform koordiniert.
Die folgende Abbildung 4.1 veranschaulicht den erläuterten Anwendungsfall.

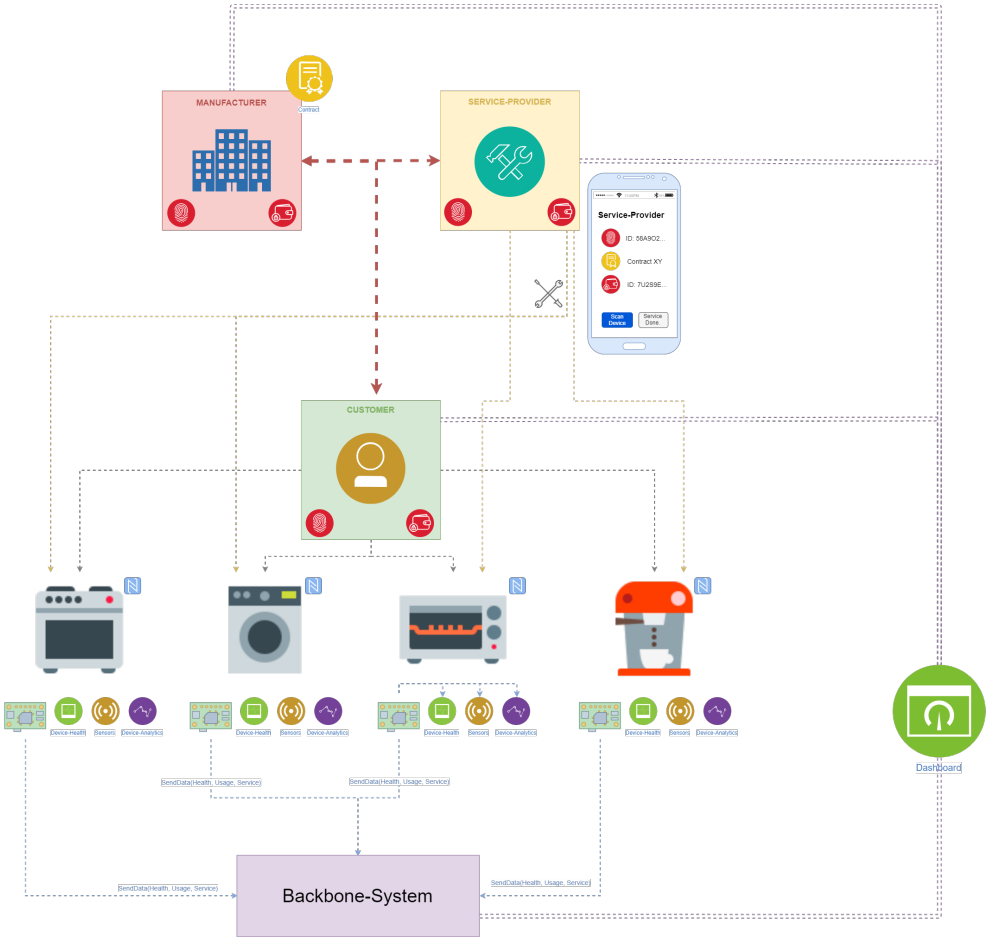

Abbildung 4.1: Grafische Veranschaulichung des Anwendungsfalls

Der vorgestellte Anwendungsfall beinhaltet das Zusammenspiel mehrerer Stakeholder:

HERSTELLER Der Hersteller der Geräte entwickelt und produziert die zu vermietenden Haushaltsgeräte und bietet diese zur Vermietung an Kunden auf der Plattform an. Er vertreibt Ersatzteile sowie Pflege- und Zusatzprodukte zu seinen Geräten, die Kunden und Service-Dienstleister erwerben können. Die vermieteten Geräte des Herstellers können Service-Dienstleister selbstständig und automatisiert für eine Reparatur oder eine Wartung beauftragen. Die Beauftragung und Abrechnung erfolgt über die Plattform. Für vermietete Geräte erhält der Hersteller nach ei-

nem Pay-as-You-Use Prinzip eine Bezahlung der Kunden entsprechend ihres Verbrauches.

LIEFERANT Der Lieferant ist für die Lieferung der Geräte und Zusatzprodukte zu den Kunden und Service-Dienstleistern zuständig. Er holt die Ware beim Hersteller ab und liefert diese aus; die benötigten Adressinformationen sind auf der Plattform hinterlegt. Die Bezahlung für die Auslieferung erfolgt über die Plattform und berechnet sich automatisch über die Distanz der Lieferstrecke und der Abmessung der Ware.

KUNDE Der Kunde mietet Geräte vom Hersteller. Die Bestellung und Abrechnung erfolgt über die Plattform nach einem Pay-as-You-Use Prinzip. Die regelmäßige Reinigung der Geräte wird auf der Plattform protokolliert. Bei Einhaltung der vorgeschriebenen Reinigungsintervalle erhält der Kunde eine vertraglich festgelegte Gutschrift, bei Nicht-Beachten eine entsprechende Gebühr. Darüber hinaus kann der Kunde auf Wunsch Konsumgüter wie Kaffee und Reinigungsmittel über die Plattform bestellen; dies geschieht vollautomatisch über das Gerät: Sobald die Menge des Produktes ein gewisses Limit unterschreitet, beauftragt das Gerät selbstständig den Kauf und die Anlieferung der Produkte über die Plattform.

SERVICE-DIENSTLEISTER Der Service-Dienstleister ist zuständig für die Wartung und Reparatur der Geräte und wird von den Geräten über die Plattform beauftragt.

Die Abbildung 4.2 zeigt den groben Ablauf beginnend mit der Mietanfrage eines Kunden bis zur monatlichen Bezahlung der Teilnehmer für ihre Leistungen.

4.2 TECHNISCHE LÖSUNGSSKIZZE

In dieser Arbeit wird der oben beschriebenen Anwendungsfall basierend auf einer DLT-Lösung prototypisch umgesetzt. Neben den beteiligten Stakeholdern besteht das Gesamtsystem aus einem Frontend, dass jedem Stakeholder die für ihn relevanten Funktionen zur Verfügung stellt und Informationen anzeigt. Das Backend des Systems besteht aus einer DLT-Lösung[5].

4.2.1 *Endgeräte*

Jedes Gerät besitzt eine eindeutige Identifikationsnummer und eine eindeutige Referenz zu dessen Eigentümer (Hersteller). Befindet sich ein Gerät in Vermietung, so existiert zwischen dem Mieter (Kunde) und dem Vermieter (Hersteller) ein Vertrag, der auf der Plattform persistiert wird. Das Gerät hat über das Internet Zugriff auf diese Plattform und damit auf den Vertrag,

5 Die konkrete Implementierung, die als Backend-System eingesetzt wird, wird im Laufe dieser Arbeit ermittelt.

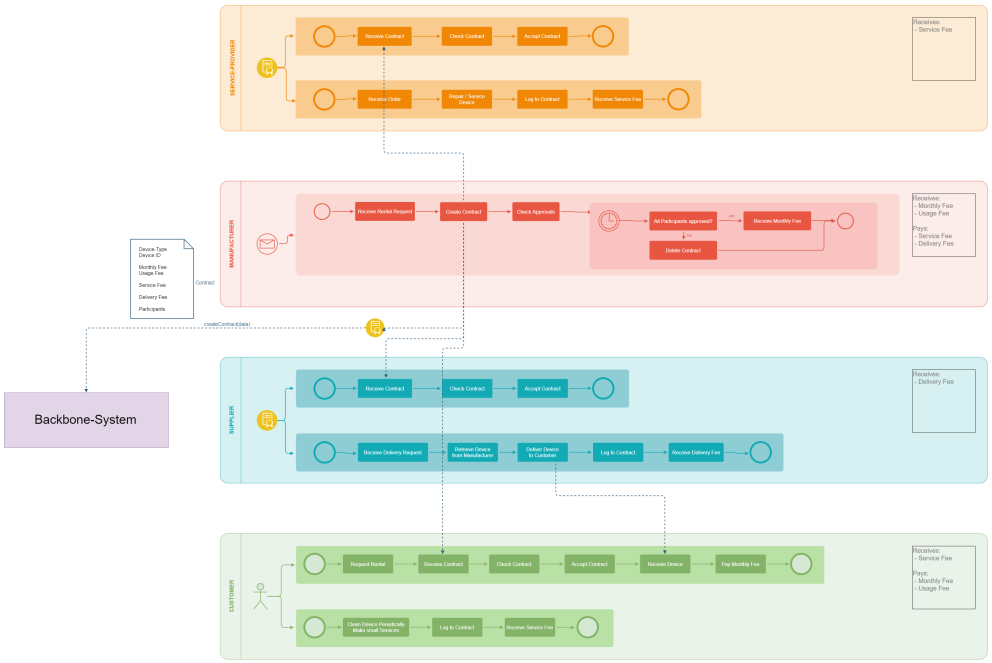

Abbildung 4.2: Grober Ablauf des Anwendungsfalls

in dem wichtige Informationen zu den Rahmenbedingungen wie der Dauer des Vertragsverhältnisses oder die Kosten einer verbrauchten Einheit. Die Sensoren zum Registrieren des Verbrauchs und des Gerätestatus befinden sich auf dem Gerät selbst. Diese melden alle gesammelten Daten an eine Sammelstelle am Gerät. Dort werden die Daten aufbereitet, gemäß Vertrag verarbeitet und gesammelt. In einem regelmäßigen Intervall meldet das Gerät alle relevanten Daten wie Nutzung, Reinigungs- und Wartungsarbeiten und sonstige Informationen an das Backend und den verknüpften Vertrag, der wiederum die entsprechenden Geld-Transfers in die Wege leitet (siehe unten). In Abbildung 4.3 wird ein Endgerät schematisch dargestellt.

4.2.2 *Verträge*

Verträge (Mietverträge, Service-Verträge, Lieferverträge) werden von allen beteiligten Parteien digital unterschrieben und im Backend gespeichert. Sie halten verschiedene Informationen, unter Anderem über die Vertragslaufzeit, die Kosten sowie die zu erbringenden Leistungen der Parteien. Der Vertrag beinhaltet Mechanismen zur Begutschriftung und zur Belastung der Konten aller Beteiligten; die folgende Auflistung nennt alle wichtigen Geld-Transfers:

- Nutzung durch den Kunden (Sender ist der Kunde, Empfänger ist der Hersteller)
- Reinigung durch den Kunden (Sender ist der Hersteller, Empfänger ist der Kunde)

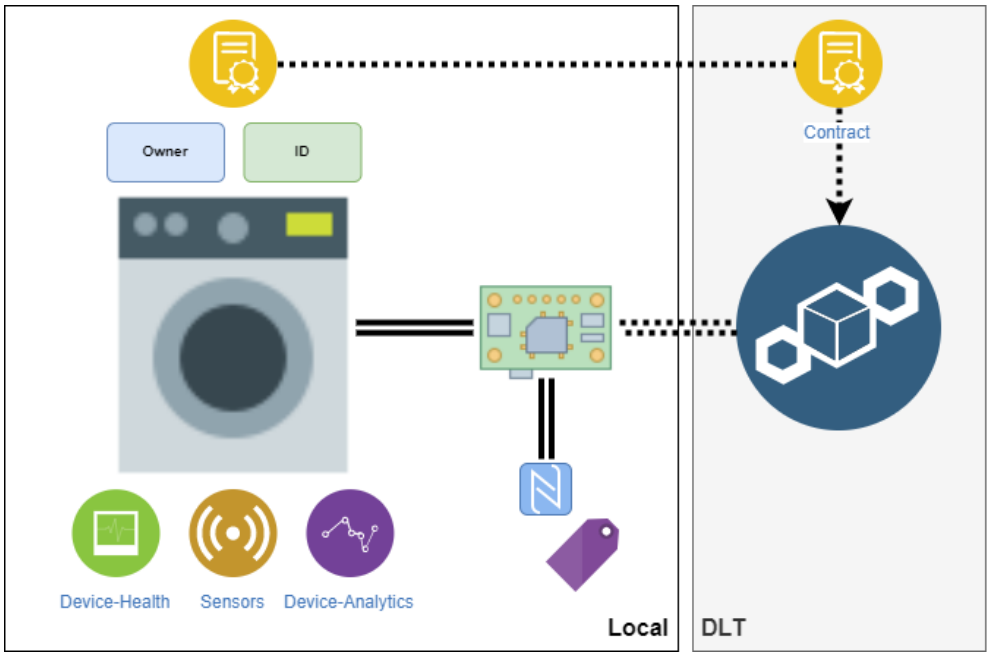

Abbildung 4.3: Aufbau und Bestandteile eines Endgeräts

- Wartung durch den Service-Dienstleister (Sender ist der Hersteller, Empfänger ist der der Service-Dienstleister)
- Lieferung durch den Lieferanten (Sender ist der Absender, Empfänger ist die Lieferant)

4.2.3 *Frontend*

Das Frontend stellt eine grafische Benutzerschnittstelle bereit, die je nach Stakeholder die relevanten Informationen und Funktionalitäten bereitstellt. Um an der Plattform teilnehmen zu können, muss ein Registrierungsprozess durchlaufen werden und die Rolle bestimmt werden (Hersteller, Kunde, ...). Jeder Rolle ist der Zugriff auf eine Ansicht gestattet:

HERSTELLER-ANSICHT Eine Übersicht über alle Geräte sowie deren Status, ob sie sich derzeit in Vermietung befinden, gibt dem Hersteller Aufschluss über die momentane Gesamtlage. Laufende Verträge können eingesehen und aktuelle Mietanfragen bearbeitet werden.

KUNDEN-ANSICHT Die gemieteten Geräte sowie die damit verbundenen, laufenden Verträge werden angezeigt. Es besteht Transparenz über Verbrauchs- und Statusinformationen, die die Geräte an die Plattform übermitteln. Laufende Kosten und aktueller Verbauch werden übersichtlich dargestellt. Es können Verträge gekündigt und neue Geräte gemietet werden.

SERVICE-DIENSTLEISTER-ANSICHT Eine Übersicht über alle aktuell laufenden Service-Verträge wird angezeigt. Alle Meldungen über Service-

Anfragen und Aufträge werden aufgelistet. Die Einnahmen durch Reparaturen und Services sowie den Kontostand werden detailliert dargestellt.

LIEFERANTEN-ANSICHT Eine Übersicht über alle aktuell laufenden Lieferverträge wird angezeigt. Die Einnahmen durch Auslieferungen sowie den Kontostand werden detailliert dargestellt.

4.2.4 *Backend*

Die konkrete DLT-Lösung wird im weiteren Verlauf dieser Arbeit evaluiert. Als dezentrale Plattform verwaltet und speichert sie alle Verträge sowie die Identitäten und Konten (Wallets) der oben aufgelisteten Stakeholder.

5 ANFORDERUNGEN

In diesem Kapitel werden die Anforderungen für den im vorherigen Kapitel aufgezeigten Anwendungsfall aufgestellt. Ziel ist es, die DLT-relevanten Anforderungen zu identifizieren, um dadurch im nächsten Kapitel eine geeignete DLT-Lösung für das Backend-System zu finden. Dazu werden zunächst einige Standards vorgestellt, wie Anforderungen klassifiziert und eingeordnet werden können. Diese Standards werden anschließend in einem angepassten Modell zusammengeführt und auf die konkreten Anforderungen angewandt. Die Klassifizierung dient als Werkzeug zur Einteilung der unübersichtlichen Gesamtmenge der Anforderungen und soll die weitere Identifikation DLT-relevanter Anforderungen vereinfachen. Als Zwischenergebnis existieren verschiedene Anforderungsklassen, die auf DLT-Relevanz geprüft werden. Durch schrittweises Ausschließen und Reduzieren der Anforderungsklassen werden jene Klassen identifiziert, die für die Umsetzung auf einer DLT-basierten Lösung entscheidend sind.

5.1 STANDARDS UND NORMEN

In diesem Abschnitt werden die verwendeten Standards und Normen aufgelistet und kurz beschrieben:

BABOK Das Business Analysis Body of Knowledge (BABOK) wird vom International Institute of Business Analysis (IIBA) herausgegeben und stellt einen Leitfaden für die Business-Analyse dar, genauere Informationen können [9] entnommen werden. Es unterteilt in sogenannte Knowledge-Areas und vermittelt Techniken und Kompetenzen im Umfeld der Business-Analyse. Anforderungen werden im BABOK nach Abstraktionsebene gruppiert: Die Business-Ebene, die Anforderungen abstrakt aus Sicht der gesamten Organisation betrachtet, die Stakeholder-Ebene, die die Anforderungen aus Sicht der verschiedenen Stakeholder beschreibt und die Solution-Ebene, die in funktionale und nicht-funktionale Anforderungen unterscheidet. Darüber hinaus gibt es eine Transition-Ebene, die temporäre Übergangsanforderungen zwischen dem Ausgangs- und dem Zielzustand des Gesamtsystems beschreibt.

PMBOK Das Project Management Body of Knowledge (PMBOK) wird vom Project Management Institute (PMI) herausgegeben und ist der State-of-the-Art Standard im Bereich Projektmanagement, siehe [10]. Das Werk teilt seine Sektionen ebenfalls wie das BABOK in sogenannte Knowledge-Areas ein und kennt die gleiche Gruppierung im Bereich Anforderungsmanagement. Neben der Einteilung in Business, Stakeholder, So-

lution und Transition Anforderungen kennt das PMBOK noch Quality und Project Anforderungen.

SWEBOK Das Software Engineering Body of Knowledge (SWEBOK) wurde von der Institute of Electrical and Electronics Engineers (IEEE) erstellt und stellt ein Standardwerk aus dem Bereich Software-Engineering dar, genauere Informationen können [1] entnommen werden. Anforderungen werden in System- und Software-Anforderungen unterteilt. Letztere werden untergliedert in Funktionale, Nicht-Funktionale, Produkt und Prozess Anforderungen.

SEBOK Das System Engineering Body of Knowledge (SEBOK) wurde unter Anderem von der IEEE erstellt und stellt ein Standardwerk aus dem Bereich System-Engineering dar, genauere Informationen können [4] entnommen werden. Anforderungen werden sehr detailliert unterteilt, unter Anderem in die Klassen Functional, Usability, Interface, Performance, Policies and Regulations, etc.

ISO29148 Die Norm 29148 der ISO beschreibt das Anforderungs-Engineering ISO als Teilbereich des Software-Engineering, genauere Informationen können [7] entnommen werden. Anforderungen werden ähnlich wie beim SEBOK untergliedert in Functional, Usability, Interface. Darüber hinaus kennt die Norm Human Factors und Process Anforderungen.

ISO25010 Die Norm 25010 der ISO beschreibt Qualitätskriterien eines Software-Produktes, siehe [8]. Die Kriterien Performance-Efficiency, Compatibility, Usability, Reliability, Security, Maintainability, Portability sowie deren Unterkriterien werden zur detaillierten Beschreibung von Nicht-Funktionalen Anforderungen eingesetzt.

Das Schaubild 5.1 zeigt die unterschiedlichen Normen und Standards grafisch auf und stellt die verschiedenen Anforderungsgruppierungen zueinander in Beziehung.

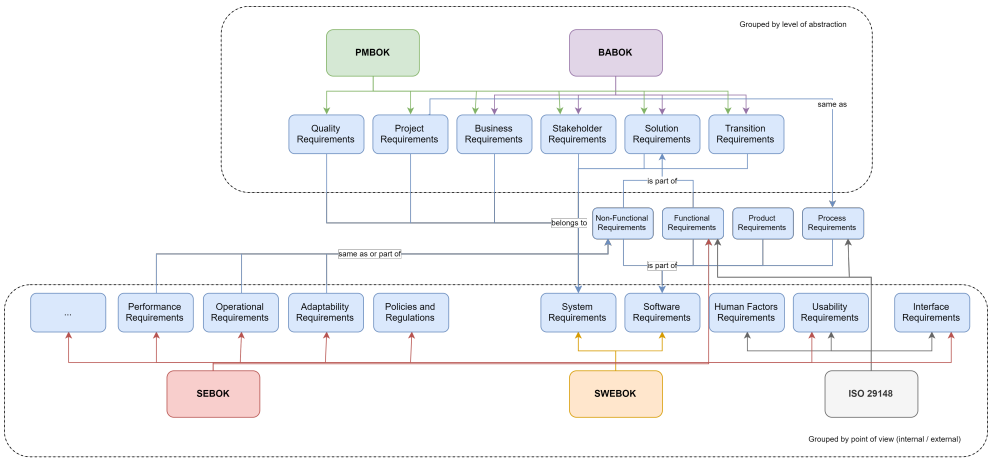

Abbildung 5.1: Einordnung der Begriffe und Zusammenhänge unterschiedlicher Normen und Standards

Grundsätzlich unterscheiden die vorgestellten Ansätze zur Anforderungsklassifizierung in zwei Sichtweisen: PMBOK und BABOK unterscheiden Anforderungen nach Abstraktionslevel während SWEBOK, SEBOK und ISO29148 primär nach der Perspektive der Stakeholder gruppieren. Das Schaubild 5.2 stellt diesen Zusammenhang grafisch dar.

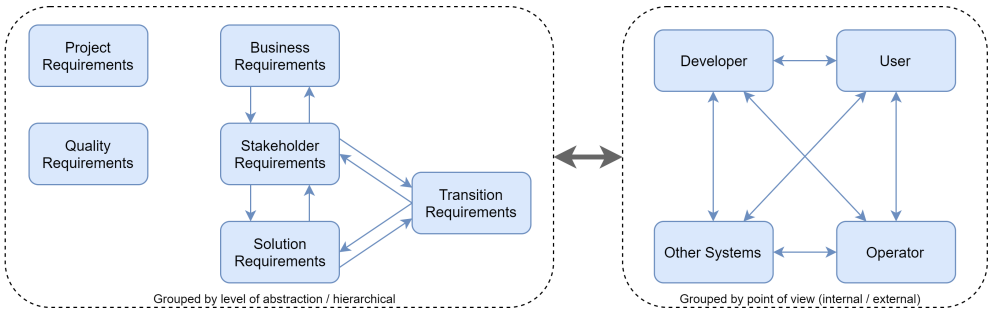

Abbildung 5.2: Anforderungen werden nach zwei verschiedenen Ansätzen gruppiert.

5.2 ABLEITUNG EINES KLASSIFIZIERUNGSMODELLS

Die Vorteile mehrerer der vorgstellten Ansätze zur Anforderungsklassifizierung können kombiniert werden, indem Anforderungen stufenweise in Unterklassen unterteilt werden. Dabei wird sich der Klassifizierung des [1], [10] bzw. [9] und [8] bedient und die folgende Einordnung erstellt:

SYSTEM ANFORDERUNGEN Anforderungen dieser Klasse beziehen sich auf das Gesamtsystem als solches.

SOFTWARE ANFORDERUNGEN Anforderungen dieser Klasse beziehen sich auf die Softwarekomponente des Gesamtsystems.

BUSINESS ANFORDERUNGEN Diese Anforderungen gehören zu der Klasse der System-Anforderungen und beschreiben Anforderungen, die sich an das Geschäftsmodell hinter dem Gesamtsystem richten. Der Scherpunkt liegt auf dem Mehrwert für die Organisation und dem damit verbundenen Nutzen des Gesamtsystems.
Leitfrage: „Welche Geschäftsfälle gibt es und wie werden diese abgedeckt? Welche Richtlinien und Vorgaben müssen beachtet werden? “

STAKEHOLDER ANFORDERUNGEN Diese Anforderungen gehören zu der Klasse der System-Anforderungen und beschreiben Anforderungen, die die Interessen der beteiligten Stakeholder widerspiegeln und sich keiner anderen Klasse zuordnen lassen.
Leitfrage: „Was muss das Gesamtsystem aus Sicht von [Stakeholder] können? “

TRANSITIONSANFORDERUNGEN Diese Anforderungen gehören zu der Klasse der System-Anforderungen und beschreiben den Übergang vom IST-Zustand des Systems in den SOLL-Zustand. Beispiele hierfür sind benötigte Anwenderschulungen oder Datenkonvertierungen.
Leitfrage: „Was muss gegeben sein, damit sich das Gesamtsystem von Zustand A in den Zustand B überführen lässt? “

PROJEKT ANFORDERUNGEN Diese Anforderungen gehören zu der Klasse der System-Anforderungen und beschreiben die Rahmenbedingungen an das Entwicklungsprojekt. Beispiele hierfür können die Projektsprache und Dokumentationsrichtlinien sein.
Leitfrage: „Welche Rahmenbedingungen sind dem Entwicklungsprojekt gegeben? “

QUALITÄTSANFORDERUNGEN Als Unterklasse der System-Anforderungen beschreiben die Qualitätsanforderungen die Qualitätsansprüche an das System und die Entwicklung und definieren Akzeptanzkriterien ähnlich zu Definition of Ready (DoR) bzw. Definition of Done (DoD).
Leitfrage: „Welche Qualitätsansprüche werden an das Gesamtsystem gestellt? “

NICHT-FUNKTIONALE ANFORDERUNGEN Diese Anforderungen werden gemäß ISO-Norm 25010 zur Software-Qualität definiert und sind Teil der Software-Anforderungen. Dazu zählen zum Beispiel die Performanz, die Kompatibilität und die Benutzbarkeit. Eine ausführliche Auflistung aller Klassen unter dem Sammelbegriff der nicht-funktionalen Anforderungen sowie die genauen Definitionen der Begriffe kann unter [8] eingesehen werden.
Leitfrage: „Wie gut muss die Software etwas können? “

FUNKTIONALE ANFORDERUNGEN Diese Unterklasse der Software-Anforderungen beschreibt, was das Software-System leisten muss und welche Aufgaben es erfüllen muss.
Leitfrage: „Was muss die Software können? “

PROZESS ANFORDERUNGEN Als Untergruppe der Software-Anforderungen bündelt diese Klasse alle Anforderungen, die den Prozess beschreiben, damit die Software so wird wie gefordert. Typischerweise sind Anforderungen an den Softwareentwicklungsprozess enthalten.
Leitfrage: „Was ist beim Entwickeln der Software zu beachten? “

Die Abbildung 5.3 fasst die Anforderungstypen zusammen und stellt sie hierarchisch strukturiert dar. Es wird deutlich, dass das entwickelte Modell beide Sichtweisen (vgl. Abbildung 5.2) aufgreift. Auf Seite der System-Anforderungen werden verschiedene Abstraktionslevel wie zum Beispiel die Business-Anforderungen und die Stakeholder-Anforderungen unterschieden. Stakeholder-Anforderungen wiederum spiegeln die Interessen der Beteiligten wider und betrachten die Anforderungen zusammen mit den Software-Anforderungen aus unterschiedlichen Perspektiven.

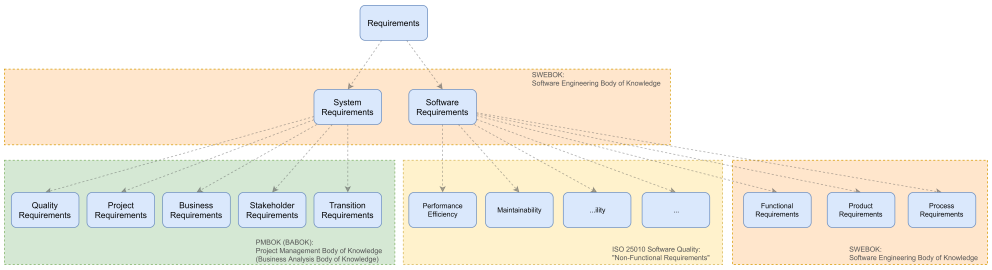

Abbildung 5.3: Anforderungsklassifizierung als kombiniertes Modell aus [1], [10] bzw. [9] und [8]

5.3 ANFORDERUNGSANALYSE

In diesem Abschnitt werden die Anforderungen zu dem in Kapitel 4 vorgestellten Anwendungsfall ermittelt. Dazu werden zunächst alle beteiligten Stakeholder identifiziert und kurz beschrieben. Anschließend werden aus der jeweiligen Perspektive heraus User-Stories gebildet und die daraus abgeleiteten Tasks aufgelistet. Eine detaillierte Beschreibung aller daraus resultierenden Anforderungen sowie die genaue Zuordnung zu den User-Stories sind im Anhang A zu finden.

Die Rollen Hersteller, Kunde, Service-Dienstleister und Lieferant wurden bereits unter 4.1 beschrieben. Die folgende Auflistung zeigt weitere Anforderungsrollen, die über die bereits genannten hinausgehen:

PLATTFORM-BETREIBER Der Plattform-Betreiber ist verantwortlich für den Betrieb der Plattform und ist hauptsächlich an einem stabilen System und einer einfachen Wartung der Software interessiert.

IT-SECURITY-BEAUFTRAGTER Für den IT-Security-Beauftragten stehen aller sicherheitsrelevanten Themen im Fokus. Dazu zählen insbesondere Verschlüsselung, Datensicherheit und Datenschutz sowie die Authentizität von Daten.

BUSINESS-DEVELOPER Der Business-Developer beschäftigt sich mit der Unternehmensentwicklung und hat Anforderungen an das Geschäftsmodell, die Wirtschaftlichkeit und die Zielerreichung des Gesamtsystems.

SYSTEM-ARCHITEKT Für den Software-Architekten stehen alle Fragen rund um die IT-Architektur der Plattform im Vordergrund. Dazu zählen Application Programming Interface (API)s, Modularisierung und der generelle Aufbau der Software.

Um konkrete Anforderungen zu erstellen, werden Userstories aus der Perspektive jedes Stakeholders erarbeitet. Userstories beschreiben eine Funktionalität des Systems, welche aus Sicht der jeweiligen Rolle benötigt wird, um ein bestimmtes Ziel zu erreichen oder einen bestimmten Zweck zu erfüllen. Somit haben alle Userstories die einheitliche Grundstruktur: „Als [Rolle] möchte ich [Funktion], um [Ziel / Zweck]. “Zu jeder User-Story werden

Tasks beschrieben, die die User-Stories in logische Teile untergliedern. Im letzten Schritt werden diese Tasks feingranular in Anforderungen unterteilt. Die Klassifizierung wird anschließend nach dem erarbeiteten Modell aus 5.2 durchgeführt.

Die erste Userstory (A1) fasst grundlegende Tätigkeiten über das Agieren auf der Plattform als Akteur (Hersteller, Kunde, Service-Dienstleister, Lieferant, Endgerät) zusammen. Dies beinhaltet sämtliche Interaktionen und Bedingungen, die für alle Akteure gleich sind. Die Userstory A1 beinhaltet vier Tasks:

- Die Zugangsberechtigung zur Plattform ist in fünf Anforderungen untergliedert, vier davon sind als *Funktionale Anforderungen* klassifiziert und eine als *Security Anforderung*.
- Die Kommunikation zwischen Rollen ist durch fünf Anforderungen definiert und beinhaltet eine *Funktionale Anforderung*, drei *Security Anforderungen* und eine *Performance-Efficiency Anforderung*.
- Die Vertragsgestaltung ist in in elf Tasks untergleidert, zwei davon sind *Security Anforderungen* und neun *Funktionale Anforderungen*.
- Grafische Oberflächen, die alle Akteure zur Interaktion mit der Plattform benötigen, sind in zwei *Funktionale Anforderungen* unterteilt.

Die Userstories M1 bis M3 sind aus Sicht eines Manufacturers beschrieben:

- Userstory M1 beschreibt die Vermietung von Endgeräten und beinhaltet zwei Tasks mit insgesamt sechs Anforderungen. Fünf der sechs Anforderungen sind *Funktionale Anforderungen*, eine wurde als *Portability Anforderung* klassifiziert.
- Userstory M2 beschreibt das Erzeugen von Verträgen und beinhaltet einen Task mit drei Anforderungen, welche alle der Klasse der *Funktionalen Anforderungen* zugeordnet wurden.
- Userstory M3 beschreibt die Abrechnung von Verträgen und beinhaltet zwei Tasks mit insgesamt drei Anforderungen, wobei eine Anforderung als *Funktionale Anforderung*, eine als *Performance-Efficiency Anforderung* und eine als *Security Anforderung* bestimmt wurden.

Die Userstories C1 bis C3 sind aus Sicht des Customers beschrieben:

- Userstory C1 beschreibt die Ansicht verfügbarer Geräte, also eine Oberfläche, die dem Customer bereitgestellt werden muss, auf der er alle zur Miete verfügbaren Geräte gelistet bekommt. Diese Userstory beinhaltet zwei Tasks mit insgesamt zwei *Funktionalen Anforderungen*.
- Userstory C2 beschreibt die Wartung und Reinigung der Geräte durch den Customer und beinhaltet vier Tasks mit insgesamt sechs Anforderungen. Dabei handelt es sich um vier *Funktionale Anforderungen*, eine *Security Anforderung* und eine *Performance-Efficiency Anforderung*.

- Userstory C3 beschreibt die Bedienbarkeit aus Nutzersicht und beinhaltet zwei Tasks mit insgesamt drei Anforderungen. Dabei handelt es sich um eine *Transition Anforderung* und zwei *Usability Anforderungen*.

Die Userstories SP1 und SP2 sind aus Sicht des Service-Providers beschrieben:

- Userstory SP1 beschreibt das Anbieten eigener Service-Dienstleistungen auf der Plattform mit einem Task und zwei *Funktionalen Anforderungen*.
- Userstory SP2 beschreibt das Abschließen von Service-Aufträgen nach getätigtem Service an den Geräten vor Ort und ist unterteilt in zwei Tasks mit je zwei *Funktionalen Anforderungen*.

Die Userstories SEC1 bis SEC3 sind aus Sicht des Security-Beauftragten beschrieben:

- Userstory SEC1 beschreibt die sichere Zahlungsabwicklung und ist in zwei Tasks mit insgesamt fünf Anforderungen unterteilt. Dabei handelt es sich um zwei *Funktionale Anforderungen* und drei *Security Anforderungen*.
- Userstory SEC2 beschreibt die sichere Kommunikation mittels signierter Nachrichten und untergliedert sich in zwei Tasks mit insgesamt zwei *Quality Anforderungen*.
- Userstory SEC3 beschreibt die Manipulationssicherheit und ist in zwei Tasks mit insgesamt drei *Security Anforderungen* unterteilt.

Die Userstories BD1 bis BD4 sind aus Sicht des Business-Developers beschrieben:

- Userstory BD1 beschreibt das Geschäftsmodell und beinhaltet zwei Tasks mit insgesamt sechs Anforderungen, wobei fünf davon als *Funktionale Anforderungen* und eine als *Business Anforderung* klassifiziert wurden.
- Userstory BD2 beschreibt den Zugang von Geschäftspartnern zu der Plattform und ist in zwei Tasks mit insgesamt drei Anforderungen untergliedert. Diese wurden als *Business Anforderung*, *Transition Anforderung* und *Stakeholder Anforderung*.
- Userstory BD3 beschreibt die Plattform als Hersteller-übergreifend und ist in drei Tasks mit insgesamt drei Anforderungen unterteilt. Dabei wurde eine Anforderung als *Business Anforderung* und zwei als *Compatibility Anforderungen* klassifiziert.
- Userstory BD4 beschreibt die Abrechnungsmodelle der Plattform und ist in einen Task mit zwei Anforderungen unterteilt. Dabei handelt es sich um eine *Business Anforderung* und einer *Maintainability Anforderung*.

Die Userstories SA1 und SA2 sind aus Sicht des System-Architekten beschrieben:

- Userstory SA1 beschreibt die einfache Einbindung der Plattform in bestehende Infrastruktur und ist in einen Task mit einer *Process Anforderung* unterteilt.
- Userstory SA2 beschreibt die Robustheit des Gesamtsystems bei Ausfällen und ist in zwei Tasks mit je zwei Anforderungen unterteilt. Dabei handelt es sich um zwei *Security Anforderungen* und zwei *Reliability Anforderungen*.

Die Userstory P1 ist aus Sicht des Plattform-Betreibers beschrieben:

- Userstory P1 beschreibt die automatisierte Bereitstellung der Software und ist in zwei Tasks mit je einer Anforderung untergliedert. Diese sind als *Maintainability Anforderung* und als *Portability Anforderung* klassifiziert.

Insgesamt wurden 19 Userstories beschrieben, die in 39 Tasks unterteilt wurden. Die 83 daraus resultierenden Anforderungen wurden klassifiziert, sodass sich 9 Level-1 *System Anforderungen* und 74 Level-1 *Software Anforderungen* ergaben. Für jede Level-2 Klasse als Unterklasse der *System Anforderungen* wurden Anforderungen ermittelt bis auf die Klasse der Projekt-Anforderungen. Im Rahmen dieser Arbeit wird ein PoC entwickelt, womit diese Arbeit aus Sicht der Anforderungserstellung als Entwicklungsprojekt angesehen werden kann. Rahmenbedingungen wie die zeitliche Begrenzung des Projektes oder dass es sich um einen PoC und nicht um ein ausgereiftes Produkt handelt könnten als Projekt-Anforderungen definiert werden. Da dies aber keine konkrete Auswirkung auf den Anwendungsfall als solchen hat, wird an dieser Stelle auf diese Klasse verzichtet. Bei der Entwicklung eines marktreifen Produktes ist diese Klasse allerdings zu beachten.
Auf Seiten der Software-Anforderungen wurden alle Level-2 Klassen bis auf die *Functionality-Suitability Anforderungen* abgedeckt. Dies liegt daran, dass diese Kategorie als einzige der ISO-25010 Klassen zu der Klasse der Funktionalen Anforderungen gehört wird und damit implizit enthalten ist.

5.4 ANFORDERUNGSEVALUIERUNG

Die Anforderungsevaluierung hat zum Ziel, die im vorherigen Abschnitt beschriebenen und klassifizierten Anforderungen schrittweise zu reduzieren, um die DLT-relevanten Anforderungen zu identifizieren. Dabei handelt es sich um Anforderungen, die relevant für eine technische Umsetzung auf Basis einer DLT-Lösung sind.

5.4.1 *System Anforderungen*

Die erste Level-2 Subklasse der *System Anforderungen* ist die Klasse der *Quality Anforderungen*. Diese Klasse definiert die Qualitätskriterien, die die Plattform erfüllen muss. Anforderungen dieser Klasse fungieren oft als Enabler für weitere Anforderungen anderer Klassen. Für den beschriebenen IOT-Anwendungsfall wurden zwei Anforderungen identifiziert, die dieser Klasse zuzuordnen (Anforderungen SEC2.1.1, SEC2.2.1) und Teil der Userstory SEC2 sind. Sie beschreiben, dass die Verwendung von HTTPS bzw. SSL/TLS ein vorgeschriebenes Qualitätskriterium ist. Darüber hinaus müssen Passwortregeln hinterlegbar sein, um die Sicherheit der auf der Plattform verwendeten Passwörter zu gewährleisten. Es wird deutlich, dass die Anforderungen dieser Klasse primär Rahmenbedingungen darstellen und keine direkten Auswirkungen auf die technische Basis haben. Diese Qualitätsansprüche, die mit den genannten Anforderungen einhergehen, haben keine Auswirkung auf eine mögliche Umsetzung der Plattform durch eine DLT-Lösung. Damit werden die Anforderungen im weiteren Verlauf nicht tiefergehend betrachtet.

Die zweite Level-2 Subklasse der *System Anforderungen* ist die Klasse der *Project Anforderungen*, die Rahmenbedingungen an das Projekt beschreiben. Die Entwicklung des PoC im Rahmen dieser Arbeit stellt ein solches Entwicklungsprojekt dar, ist allerdings für die Betrachtung in diesem Kontext hinsichtlich DLT-Relevanz nicht weiter zu berücksichtigen. Somit werden die *Project Anforderungen* nicht in die weitere Analayse miteinbezogen.

Die dritte Level-2 Subklasse der *System Anforderungen* ist die Klasse der *Business Anforderungen*, die die Geschäftsfälle und -anforderungen von einer abstrakteren Perspektive betrachten. Im Fokus stehen die Bedürfnisse und Rahmenbedingungen des Unternehmens, welches die Plattform beauftragt hat. Im Rahmen des IOT-Anwendungsfalls wurden vier Anforderungen dieser Klasse identifiziert (Anforderungen BD1.1.1, BD2.1.1, BD3.1.1 und BD4.2.1); betroffen sind die Userstories BD1 bis BD4. Die Anforderungen decken das Abrechnungsmodell Pay-As-You-Use ab und beschäftigen sich mit der aktuellen und zukünftigen geschäftlichen Ausrichtung der Plattform. Dies hat keine Auswirkungen auf die technische Basis, die die zugrundeliegende Plattform verwendet: Es handelt sich um keine Technologieentscheidende Anforderung. Anforderung BD4.2.1 beschreibt, dass ein Abrechnungsmodell (konkret: Pay-As-You-Use) in einem Vertrag abgebildet wird. Diese Anforderung ist DLT-relevant: Zum einen impliziert diese Anforderung, dass eine zugrundeliegende DLT-Plattform in der Lage ist, Verträge abzubilden. Im Umfeld von DLTs redet man Smart-Contracts: Code, der auf onchain ausgeführt wird und eine Vertragslogik widerspiegeln kann. Zum anderen muss gewährleistet sein, dass die Smart-Contract Implementierung mächtig genug ist, damit das Abrechnungsmodell Pay-As-You-Use dort codiert werden kann. Diese Anforderung muss in die weitere Analyse mitein-

bezogen werden.

Die vierte Level-2 Subklasse der *System Anforderungen* ist die Klasse der *Stakeholder Anforderungen*, welche Anforderungen speziell aus der Sicht einzelner Stakeholder beschreiben, die mit anderen Klassen noch nicht abgedeckt werden konnten. Im Kontext des Anwendungsfalls wurde eine Anforderung identifiziert (BD2.2.1), die dieser Klasse zuzuordnen ist und zur Userstory BD2 gehört. Es wird der Onboarding-Prozess eines Geschäftspartners beschrieben, indem dieser als Partner identifiziert werden muss. Dieser Prozess muss entsprechend der Anforderungen gestaltet werden, womit es sich um eine abstrakte Beschreibung dessen, wie ein Prozess auszusehen hat, handelt. Es werden keine technischen Details gefordert, wodurch keine Abhängigkeiten zu der technischen Umsetzung der Plattform entstehen. Damit ist diese Klasse für die weitere Analyse nicht relevant und kann vernachlässigt werden.

Die fünfte Level-2 Subklasse der *System Anforderungen* ist die Klasse der *Transition Anforderungen*, die sämtliche Anforderungen von einem IST-Zustand (zeitlich vor der Einführung der Plattform) in einen SOLL-Zustand (zeitlich nach Einführung der Plattform) kapseln. Zwei Anforderungen (Anforderungen C3.1.1 und BD2.1.2) wurden identifiziert, die dieser Klasse zuzuordnen und Teil der Userstories C3 und BD2 sind. Es handelt sich in dem vorliegenden Kontext um Anforderungen, die die schnelle Erlernbarkeit durch den Benutzer sowie die Schulung von Mitarbeitern zur Nutzung der Plattform beschreiben und somit die User-Experience (UX) in den Vordergrund stellen. Da es sich in diesem Fall um Aufbau, Verständlichkeit und Benutzbarkeit einer grafischen Oberfläche (Schnittstelle zum Benutzer) handelt, kann diese Klasse im weiteren Verlauf der Analyse vernachlässigt werden.

5.4.2 *Software Anforderungen*

Anforderungen dieser Level-1 Klasse stellen den Großteil aller Anforderungen dar. In einem realistischen Entwicklungsprojekt sind die individuellen Rahmenbedingungen, Qualitätskriterien, Business-Richtlinien und Integrationsrichtlinien des jeweiligen Unternehmens zu beachten. Die in dieser Arbeit aufgestellten System-Anforderungen decken nur die grundlegendsten Anforderungen dieser Kategorie ab. Die *Software Anforderungen*, die in diesem Abschnitt evaluiert werden, sind unabhängig von den *System Anforderungen* stets dieselben.

Die erste Level-2 Subklasse der *Software Anforderungen* sind die *Process Anforderungen*, die Anforderungen an den Entwicklungsprozess der Software stellen. Für den vorliegenden IOT-Anwendungsfall wurde eine Anforderungen (Anforderungen SA1.1.1) ermittelt, die zu der Userstory SA1 gehört. Während der Entwicklung ist darauf zu achten, dass die Modularität der

Softwarekomponenten gewahrt bleibt, damit diese später unabhängig voneinander bereitgestellt und gewartet werden können. Dies hat keine Auswirkung auf die Umsetzung auf Basis einer DLT-Lösung. Im Allgemeinen haben Anforderungen dieser Klasse einerseits eine technische Relevanz, da sie zum Beispiel einzusetzende Technologien für Schnittstellen beschränken oder die Art und Weise definieren, wie Software entwickelt werden soll. Andererseits kann jede Software durch den Einsatz einer geeigneten Middleware miteinander verbunden werden, sollten vorhandene Schnittstellen und Softwarekomponenten nicht Standard-konform oder kompatibel sein. Somit können die Anforderungen dieser Klasse ebenfalls vernachlässigt werden.

Die zweite Level-2 Subklasse der *Software Anforderungen* sind die *Funktionalen Anforderungen*, die beschreiben, welche Funktionalität die Plattform anbieten muss. Für den vorliegenden IOT-Anwendungsfall wurden 45 solcher Anforderungen ermittelt, die insgesamt zehn Userstories betreffen. Um die Übersichtlichkeit zu bewahren, wurde diese Klasse in verschiedene, logische Abschnitte gegliedert. Diese wurden nach der inhaltlichen Thematik der Anforderungen definiert und folgen keiner speziellen Klassifizierung.

GUI 12 der 45 Anforderungen dieser Klasse betreffen direkt oder indirekt die grafische Darstellung für den Benutzer. Es handelt sich hierbei lediglich um das Frontend, welches Daten aufbereitet darstellt und Schaltflächen zur Interaktion mit dem Backend bereitstellt. Somit werden die Anforderungen dieser Klasse für weiterführende Analysen nicht beachtet. Konkret handelt es sich um die Anforderungen A1.1.4, A1.3.4, A1.4.1, A1.4.2, M2.1.1, M2.1.2, M2.1.3, C1.1.1, C1.2.1, SP1.1.1, SP1.1.2 und SP2.2.1.

ENDGERÄT 14 der 45 Anforderungen beziehen sich auf Funktionalitäten, die das Endgerät bereitstellen muss, um auf der Plattform vermietet werden zu können. Dabei geht es primär um die Konnektivität zu der Plattform, die Funktionsweise der verbauten Sensoren sowie der Kommunikation mit dem Customer und dem Service-Provider. Die Engeräte fungieren in ihrer Rolle als Peripherie und können unterschiedlicher Art sein. Es kann sich um Haushaltsgeräte aller Art handeln - von der Kaffeemaschine bis zur Waschmaschine - lediglich die Schnittstellen nach Außen sind klar definiert. Es wird deutlich, dass die Anforderungen an die Geräte unabhängig von der technischen Umsetzung der Plattform sind und keinen Einfluss auf deren Umsetzung haben. An dieser Stelle sei an die Möglichkeit einer Middleware erinnert, die die Integration verschiedener Geräte übenehmen könnte, sofern die Geräte inkompatibel zur Plattform wären. Eine Möglichkeit wäre die Verwendung eines lokalen Gateways, dass die Daten der Geräte konvertiert und an die Plattform übermittelt. Standardmäßig sollten die Geräte die Kommunikation mit der Plattform bereits Hersteller-seitig gewährleisten. (Anforderungen M1.1.1, M1.2.4, M3.1.1, C2.1.1, C2.1.2, C2.3.1, C2.4.1, SP2.1.1, SP2.1.2 und SP2.2.2)

KOMMUNIKATION Drei der 45 Anforderungen beziehen sich auf die Kommunikation zwischen Akteuren auf der Plattform (Anforderungen A1.2.1, A1.3.1 und A1.3.3). Anforderung A1.3.1 beschreibt die Kommunikation über Verträge, die die Akteure der Plattform miteinander abschließen können. Im Kontext einer DLT-Lösung bedeutet das, dass die Plattform in der Lage sein muss, einen Vertrag abzubilden (siehe Business Anforderungen oben). Diese Anforderung hat eine DLT-Relevanz und muss in der weiteren Analyse betrachtet werden. Die übrig gebliebenen zwei Anforderungen beschreiben die allgemeinere Aspekte der Kommunikation auf der Plattform und können vernachlässigt werden.

FINANZEN Sieben der 45 Anforderungen (A1.3.11, SEC1.1.1, SEC1.1.3, SEC1.2.1, SEC1.2.2, BD1.1.2, BD1.1.3, BD1.2.1 und BD4.2.1) beschreiben den Geldfluss zwischen Akteuren aufgrund ihrer vertraglichen Vereinbarungen. Geldtransfers werden geloggt und vor Ausführung überprüft. Diese Anforderungen sind für eine Umsetzung auf einer DLT-Lösung nicht relevant, da sie lediglich die Richtung und Menge des Geldflusses sowie Rahmenbedingungen an Geldtransfers festlegen. Damit allerdings erbrachte Leistungen kostenpflichtig gemäß des Abrechnungsmodells abgerechnet werden können, muss die Plattform zum einen das Abrechnungsmodell implementieren und zum anderen ein digitales Zahlungsmittel bereitstellen. Ersteres kann auf einer DLT-Lösung mittels Smart-Contracts umgesetzt werden. Zweiteres bedarf einer internen Währung, um Leistungen in Echtzeit nach Verbrauch abzurechnen. Damit sind diese zwei Anforderungen relevant und müssen in der weiteren Analyse beachtet werden.

ROLLENMANAGEMENT Drei der 49 Anforderungen (A1.1.2, A1.1.3 und A1.1.5) definieren das Rollenmanagement und den Zugang zu der Plattform mittels Registrierung und Anmeldung. Letzteres stellt keine Relevanz dar, da es sich um eine standardmäßige Zugangsbeschränkung handelt und nicht abhängig von der Backend-Lösung ist. Im Gegensatz dazu werden Rollen in den Verträgen genutzt, um Berechtigungen der Akteure zu prüfen. Es handelt sich um Informationen, die von Verträgen auf der Plattform einsehbar sein müssen. Content- oder allgemeiner Informationsprovider, die Informationen auf einer DLT-Umgebung bereitstellen, nennt man Oracles (siehe Kapitel XYZ). Damit liegt hier eine Relevanz in Bezug auf die Umsetzung auf Basis einer DLT-Lösung vor und muss im weiteren Verlauf beachtet werden.

VERTRAGSKONSTRUKT Sechs der 49 Anforderungen (A1.3.2, A1.3.6, A1.3.7, A1.3.8, A1.3.10 und BD1.2.4) beschreiben das Vertragskonstrukt: Eigenschaften wie Individualität und Zugriffsregelung haben keinen Einfluss auf die technische Lösung, wohingegen die Komplexität und Editierbarkeit (drei Anforderungen) große Relevanz haben. Zum einen muss die Implementierung eines Vertrages mittels Smart-Contracts auch komplexe Konstrukte abbilden können. Zum anderen sind Smart-Contracts - sind sie einemal in der DLT gespeichert - unveränderbar und kön-

nen als solches nicht überarbeitet werden. Die zugrundeliegende DLT-Technologie muss also einen entsprechenden Mechanismus anbieten, um Smart-Contracts zu warten und zu aktualisieren. Darüber hinaus handelt es sich bei den Verträgen, die mittels Smart-Contracts abgebildet werden sollen, um rechtskräftige Miet- und Serviceverträge. Demnach müssen die Implementierungen rechtssicher und rechtskonform abgebildet werden können. Diese Punkte sind bei der Wahl der DLT-Lösung zu beachten.

Die dritte Level-2 Subklasse der *Software Anforderungen* unter dem Sammelbegriff der *Nicht-Funktionalen Anforderungen* heißt *Kompatibilität* (Compatibility). Für den vorliegenden IOT-Anwendungsfall wurden zwei solcher Anforderungen ermittelt (BD3.2.1, BD3.3.1,), welche die Userstory BD3 betreffen. Die Kompatibilität stellt die Fähigkeit des Gesamtsystems dar, Informationen mit anderen System auszutauschen und sich eine Umgebung mit anderen Systemen zu teilen. Es handelt sich hierbei um Anforderungen, die im Kontext des IOT-Anwendungsfalls keine Relevanz für die Wahl der technischen Basis haben. Sollten Systeme nicht kompatibel sein, so könnte im Zweifelsfall eine Middleware für die Übersetzung bzw. die Kompatibilität sorgen. Somit werden die Anforderungen dieser Klasse für die weitere Analyse nicht weiter beachtet.

Die vierte Level-2 Subklasse der *Software Anforderungen* unter dem Sammelbegriff der *Nicht-Funktionalen Anforderungen* heißt *Wartbarkeit* (Maintainability). Anforderungen dieser Klasse stellen die Fähigkeit des Gesamtsystems dar, effizient wartbar zu sein um z.B. die Funktionalität zu erweitern und zu verbessern. Konkret im Kontext des IOT-Anwendungsfalls beziehen sich die Anforderungen BD4.2.2 und P1.1.1 der zwei Userstories BD4 und P1 auf den Aufbau der Software: Einzelne Module können separat gewartet werden und die Testabdeckung ist hoch genug, damit die Wartung einzelner Module keine unerwünschten Nebeneffekt mit sich führt. Es handelt sich also um generische Anforderungen, die keinen Bezug auf die technische Umsetzung haben und daher keine DLT-Relevanz besitzen. Somit wird diese Anforderungsklasse im weiteren Verlauf nicht weiter berücksichtigt.

Die fünfte Level-2 Subklasse der *Software Anforderungen* unter dem Sammelbegriff der *Nicht-Funktionalen Anforderungen* heißt *Performance-Effizienz* (Performance-Efficiency) und beschreibt die Leistung des Gesamtsystems in Bezug auf die zur Verfügung stehenden Ressourcen. Die Anforderungen dieser Klasse (A1.2.2, M3.2.1 und C2.4.2) im konkreten Anwendungsfall betreffen drei Userstories (A1, M3 und C2) und beschreiben alle das zeitliche Verhalten von übermittelten Daten: Die Kommunikation zwischen Akteuren, Geräten und Verträgen auf der Plattform wird sofort übermittelt. Es werden keine Daten zurückgehalten, aggregiert oder zeitverzögert übermittelt. Diese Anforderungen treffen keine Aussage über die Verarbeitungsdauer der Daten und haben damit keine Relevanz in Bezug auf die Wahl der technischen Basis. Somit werden die Anforderungen dieser Klasse für die weitere Analyse ignoriert.

Die sechste Level-2 Subklasse der *Software Anforderungen* unter dem Sammelbegriff der *Nicht-Funktionalen Anforderungen* heißt *Portabilität* (Portability) und beschreibt die Fähigkeit des Gesamtsystems von einer Hardware bzw. Umgebung in einer andere migriert zu werden. In diesem Kontext existiert eine Anforderung (P1.2.1) der Userstory P1, welche die automatisierte Installation und Bereistellung der Software definiert. Es ist deutlich, dass es sich hierbei um keine Technologie-entscheidende Anforderung handelt. Demnach wird diese Anforderungsklasse in der weiteren Analyse nicht beachtet.

Die siebte Level-2 Subklasse der *Software Anforderungen* unter dem Sammelbegriff der *Nicht-Funktionalen Anforderungen* heißt *Ausfallsicherheit* (Reliability) und beschreibt wie gut ein System unter bestimmten Bedingungen die geforderten Funktionalitäten durchführen kann. Anfoderung SA2.2.2 beschreibt, dass die Plattform keinen Single-Point-of-Failure (SPoF) besitzen darf. Dies beschreibt eine generelle Eigenschaft und ist im Kontext eines dezentralen Systems wie es DLT ist nicht weiter von Relevanz. Daneben fordert Anforderung SA2.2.1, dass die Plattform in der Lage ist, bis zu 10.000 Endgeräte zu verarbeiten, ohne Einbußen in der Funktionalität oder der Geschwindigkeit der Verarbeitung. Diese Anfoderung muss bei der Wahl der zugrundeliegenden DLT-Lösung beachtet werden, da hierbei die Performanz und Verfügbarkeit des beeinträchtigt wird.

Die achte Level-2 Subklasse der *Software Anforderungen* unter dem Sammelbegriff der *Nicht-Funktionalen Anforderungen* heißt *Sicherheit* (Security) und befasst sich mit der Thematik rund um Software- und Datensicherheit. Diese Klasse beinhaltet 16 Anforderungen (A1.1.1, A1.2.3, A1.2.4, A1.2.5, A1.3.5, A1.3.9, M3.2.2, C2.2.1, SEC1.1.2, SEC1.2.1, SEC1.2.2, SEC3.1.1, SEC3.1.2, SEC3.2.1, SA2.1.1 und SA2.1.2) aus sechs Userstories (A1, M3, C2, SEC1, SEC3 und SA2). Anhand der hohen Anzahl ist die Wichtigkeit dieser Klasse zu sehen. Die vorliegenden Anforderungen lassen sich grob ob vier Subkategorien untergliedern: Verantwortlichkeit, Authentizität, Vertraulichkeit und Integrität. Verantwortlichkeit beinhaltet fünf Anforderungen, die unter Anderem die eindeutige Identifikation der Akteure sowie die Protokollierung von Aktivitäten und deren Zuordnung zu Accounts beschreiben. Während es sich bei dem Großteil um allgemeine Richtlinien handelt und dieser nicht von der konkreten technischen Umsetzung abhängig ist, so ist die generelle Identifikation von großer Relevanz für die Umsetzung auf Basis einer DLT-Lösung. Um einem Account (im Kontext von DLT eher Wallet) eine natürliche Person oder ein Unternehmen eindeutig zuordnen zu können, bedarf es einer informativen Instanz, die auf der Plattform agiert und diese Informationen bereitstellt. Eine mögliche Umsetzung im Kontext DLT wäre die Implementierung eines entsprechenden Oracles, welches Personen und Unternehmen einer Wallet zuordnet und umgekehrt.
Die nächste Subkategorie - Authentizität - beinhaltet eine Anforderung, die die Zugriffsbeschränkung zu der Wallets (Konten) beschreibt. Da Wallets auf einer DLT durch eine Kombination aus einem öffentlichen und einem privaten Schlüssel sind und der private der PIN-Nummer eines Kontos ent-

spricht, besteht eine implizite Zugriffsbeschränkung; es liegt hierbei keine DLT-Relevanz vor.
Der Inhalt von Nachrichten zwischen Akteuren sowie der Vertragsinhalt abgeschlossener Verträge unterliegt der Vertraulichkeit, darf also nur von beteiligten bzw. berechtigten Akteuren eingesehen werden. Da Transaktionen auf DLT-Lösungen generell öffentlich einsehbar sind liegt hier eine DLT-Relevanz vor: Es muss dafür gesorgt werden, dass der Inhalt von Transaktionen und Smart-Contracts vertraulich behandelt werden kann.
Die Subkategorie Integrität hat den größten Anteil: Insgesamt acht Anforderungen wurden für den vorliegenden Anwendungsfall ermittelt, wobei Dateninkonsistenzen, Datenverlust und Manipulationssicherheit im Fokus stehen. Diese Eigenschaften entsprechen in etwa dem, was einen Distributed Ledger ausmacht, und sind deshalb zunächst einmal nicht weiter relevant in Bezug auf die Umsetzung mittels eines DLT. Die Abbildung eines Vertrags muss allerdings so gestaltet werden, dass der Vertragsgegenstand nicht manipuluiert werden kann. Dies ist bei der Implementierung des Vertrags zu beachten und hat demnach eine Auswirkung auf die DLT-Relevanz und muss in der weiteren Analyse beachtet werden.

Die neunte Level-2 Subklasse der *Software Anforderungen* unter dem Sammelbegriff der *Nicht-Funktionalen Anforderungen* heißt *Benutzbarkeit* (Usability) und befasst sich mit dem Thema UX. Diese Klasse beinhaltet zwei Anforderungen (C3.1.2 und C3.2.1) der Userstory C3 und besitzt keine DLT-Relevanz, da es um die Schnittstelle zum Endbenutzer geht und nicht um die technologische Basis der Plattform. Somit wird diese Klasse im weiteren Vorgehen nicht beachtet.

In den vorherigen Abschnitten wurden alle DLT-relevanten Anforderungen der verschiedenen Klassen identifiziert. Die nachfolgende Tabelle fasst diese zusammen.

Im Laufe der Untersuchung auf DLT-Relevanz wurde deutlich, dass die Einteilung nach Anforderungsklassen nur einen geringen Beitrag zur Identifikation der relevanten Anforderungen beitragen konnte. Selbst die Einteilung in verschiedene Themenbereiche konnte nur einen geringen Mehrwert liefern, indem die Übersichtlichkeit - trotz der hohen Anzahl an Anforderungen - gewahrt werden konnte. Gründe und Auswirkungen dafür werden später im Kapitel XYZ aufgezeigt und diskutiert.

DLT-relevante Anforderungen übersetzen in DLT-Sprache. Was muss ein DLT können, um in Frage zu kommen?

ID	Inhalt	Level-1	Level-2
A1.1.1	Jeder Akteur auf der Plattform kann eindeutig identifiziert werden.	Software	Security
A1.1.3	Ein Akteur agiert immer mit einer bestimmten Rolle auf der Plattform: Manufacturer, Customer, Supplier, Service-Provider oder Gerät. Ein Akteur kann mehrere Rollen haben.	Software	Functional
A1.1.5	Ein Akteur hat eine (mehrere) verifizierte Rolle(n).	Software	Functional
A1.2.5	Akteure können nur den Inhalt ihrer eigenen Nachrichten einsehen.	Software	Security
A1.3.1	Akteure schließen Verträge über die Plattform ab.	Software	Functional
A1.3.2	Verträge sind rechtlich bindend.	Software	Functional
A1.3.5	Der Vertragsgegenstand kann nicht durch Dritte manipuliert werden.	Software	Security
A1.3.7	Akteure können komplexe Vertragskonstrukte umsetzen. Verträge haben einen Status. Diese können Äktiv", În Erzeugungöder Ïnaktivßein.	Software	Functional
A1.3.8	Akteure können ihre Verträge im Nachhinein ändern.	Software	Functional
A1.3.9	Akteure können nur ihre eigenen Verträge ändern.	Software	Security
A1.3.10	Eine Vertragsänderung bedarf der Zustimmung aller beteiligten Akteure.	Software	Functional
A1.3.11	Erbrachte Leistungen werden kostenpflichtig verrechnet.	Software	Functional
BD4.2.1	Ein Abrechnungsmodell wird in einem Vertrag abgebildet.	System	Business
SA2.2.1	Die Plattform ist in der Lage, die Kommunikation und Datenverarbeitung bei bis zu 100.000 Endgeräten durchzuführen.	Software	Reliability

Tabelle 5.1: DLT-relevante Anforderungen

6 AUSWAHL RELEVANTER DLTS

Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Ut purus elit, vestibulum ut, placerat ac, adipiscing vitae, felis. Curabitur dictum gravida mauris. Nam arcu libero, nonummy eget, consectetuer id, vulputate a, magna. Donec vehicula augue eu neque. Pellentesque habitant morbi tristique senectus et netus et malesuada fames ac turpis egestas. Mauris ut leo. Cras viverra metus rhoncus sem. Nulla et lectus vestibulum urna fringilla ultrices. Phasellus eu tellus sit amet tortor gravida placerat. Integer sapien est, iaculis in, pretium quis, viverra ac, nunc. Praesent eget sem vel leo ultrices bibendum. Aenean faucibus. Morbi dolor nulla, malesuada eu, pulvinar at, mollis ac, nulla. Curabitur auctor semper nulla. Donec varius orci eget risus. Duis nibh mi, congue eu, accumsan eleifend, sagittis quis, diam. Duis eget orci sit amet orci dignissim rutrum.

6.1 VORGEHEN

Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Ut purus elit, vestibulum ut, placerat ac, adipiscing vitae, felis. Curabitur dictum gravida mauris. Nam arcu libero, nonummy eget, consectetuer id, vulputate a, magna. Donec vehicula augue eu neque. Pellentesque habitant morbi tristique senectus et netus et malesuada fames ac turpis egestas. Mauris ut leo. Cras viverra metus rhoncus sem. Nulla et lectus vestibulum urna fringilla ultrices. Phasellus eu tellus sit amet tortor gravida placerat. Integer sapien est, iaculis in, pretium quis, viverra ac, nunc. Praesent eget sem vel leo ultrices bibendum. Aenean faucibus. Morbi dolor nulla, malesuada eu, pulvinar at, mollis ac, nulla. Curabitur auctor semper nulla. Donec varius orci eget risus. Duis nibh mi, congue eu, accumsan eleifend, sagittis quis, diam. Duis eget orci sit amet orci dignissim rutrum.

6.2 MARKTÜBERSICHT DLTS

Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Ut purus elit, vestibulum ut, placerat ac, adipiscing vitae, felis. Curabitur dictum gravida mauris. Nam arcu libero, nonummy eget, consectetuer id, vulputate a, magna. Donec vehicula augue eu neque. Pellentesque habitant morbi tristique senectus et netus et malesuada fames ac turpis egestas. Mauris ut leo. Cras viverra metus rhoncus sem. Nulla et lectus vestibulum urna fringilla ultrices. Phasellus eu tellus sit amet tortor gravida placerat. Integer sapien est, iaculis in, pretium quis, viverra ac, nunc. Praesent eget sem vel leo ultrices bibendum. Aenean faucibus. Morbi dolor nulla, malesuada eu, pulvinar at, mollis ac, nulla. Curabitur auctor semper nulla. Donec varius orci eget risus.

Duis nibh mi, congue eu, accumsan eleifend, sagittis quis, diam. Duis eget orci sit amet orci dignissim rutrum.

6.3 ANFORDERUNGSERFÜLLUNG

Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Ut purus elit, vestibulum ut, placerat ac, adipiscing vitae, felis. Curabitur dictum gravida mauris. Nam arcu libero, nonummy eget, consectetuer id, vulputate a, magna. Donec vehicula augue eu neque. Pellentesque habitant morbi tristique senectus et netus et malesuada fames ac turpis egestas. Mauris ut leo. Cras viverra metus rhoncus sem. Nulla et lectus vestibulum urna fringilla ultrices. Phasellus eu tellus sit amet tortor gravida placerat. Integer sapien est, iaculis in, pretium quis, viverra ac, nunc. Praesent eget sem vel leo ultrices bibendum. Aenean faucibus. Morbi dolor nulla, malesuada eu, pulvinar at, mollis ac, nulla. Curabitur auctor semper nulla. Donec varius orci eget risus. Duis nibh mi, congue eu, accumsan eleifend, sagittis quis, diam. Duis eget orci sit amet orci dignissim rutrum.

6.4 BEWERTUNG, RANKING & AUSWAHL

Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Ut purus elit, vestibulum ut, placerat ac, adipiscing vitae, felis. Curabitur dictum gravida mauris. Nam arcu libero, nonummy eget, consectetuer id, vulputate a, magna. Donec vehicula augue eu neque. Pellentesque habitant morbi tristique senectus et netus et malesuada fames ac turpis egestas. Mauris ut leo. Cras viverra metus rhoncus sem. Nulla et lectus vestibulum urna fringilla ultrices. Phasellus eu tellus sit amet tortor gravida placerat. Integer sapien est, iaculis in, pretium quis, viverra ac, nunc. Praesent eget sem vel leo ultrices bibendum. Aenean faucibus. Morbi dolor nulla, malesuada eu, pulvinar at, mollis ac, nulla. Curabitur auctor semper nulla. Donec varius orci eget risus. Duis nibh mi, congue eu, accumsan eleifend, sagittis quis, diam. Duis eget orci sit amet orci dignissim rutrum.

7 UMSETZUNG

Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Ut purus elit, vestibulum ut, placerat ac, adipiscing vitae, felis. Curabitur dictum gravida mauris. Nam arcu libero, nonummy eget, consectetuer id, vulputate a, magna. Donec vehicula augue eu neque. Pellentesque habitant morbi tristique senectus et netus et malesuada fames ac turpis egestas. Mauris ut leo. Cras viverra metus rhoncus sem. Nulla et lectus vestibulum urna fringilla ultrices. Phasellus eu tellus sit amet tortor gravida placerat. Integer sapien est, iaculis in, pretium quis, viverra ac, nunc. Praesent eget sem vel leo ultrices bibendum. Aenean faucibus. Morbi dolor nulla, malesuada eu, pulvinar at, mollis ac, nulla. Curabitur auctor semper nulla. Donec varius orci eget risus. Duis nibh mi, congue eu, accumsan eleifend, sagittis quis, diam. Duis eget orci sit amet orci dignissim rutrum.

7.1 AUSWAHL DER ANWENDUNGSANFORDERUNGEN

Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Ut purus elit, vestibulum ut, placerat ac, adipiscing vitae, felis. Curabitur dictum gravida mauris. Nam arcu libero, nonummy eget, consectetuer id, vulputate a, magna. Donec vehicula augue eu neque. Pellentesque habitant morbi tristique senectus et netus et malesuada fames ac turpis egestas. Mauris ut leo. Cras viverra metus rhoncus sem. Nulla et lectus vestibulum urna fringilla ultrices. Phasellus eu tellus sit amet tortor gravida placerat. Integer sapien est, iaculis in, pretium quis, viverra ac, nunc. Praesent eget sem vel leo ultrices bibendum. Aenean faucibus. Morbi dolor nulla, malesuada eu, pulvinar at, mollis ac, nulla. Curabitur auctor semper nulla. Donec varius orci eget risus. Duis nibh mi, congue eu, accumsan eleifend, sagittis quis, diam. Duis eget orci sit amet orci dignissim rutrum.

7.2 POC

Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Ut purus elit, vestibulum ut, placerat ac, adipiscing vitae, felis. Curabitur dictum gravida mauris. Nam arcu libero, nonummy eget, consectetuer id, vulputate a, magna. Donec vehicula augue eu neque. Pellentesque habitant morbi tristique senectus et netus et malesuada fames ac turpis egestas. Mauris ut leo. Cras viverra metus rhoncus sem. Nulla et lectus vestibulum urna fringilla ultrices. Phasellus eu tellus sit amet tortor gravida placerat. Integer sapien est, iaculis in, pretium quis, viverra ac, nunc. Praesent eget sem vel leo ultrices bibendum. Aenean faucibus. Morbi dolor nulla, malesuada eu, pulvinar at, mollis ac, nulla. Curabitur auctor semper nulla. Donec varius orci eget risus.

Duis nibh mi, congue eu, accumsan eleifend, sagittis quis, diam. Duis eget orci sit amet orci dignissim rutrum.

7.2.1 *Implemetierung*

Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Ut purus elit, vestibulum ut, placerat ac, adipiscing vitae, felis. Curabitur dictum gravida mauris. Nam arcu libero, nonummy eget, consectetuer id, vulputate a, magna. Donec vehicula augue eu neque. Pellentesque habitant morbi tristique senectus et netus et malesuada fames ac turpis egestas. Mauris ut leo. Cras viverra metus rhoncus sem. Nulla et lectus vestibulum urna fringilla ultrices. Phasellus eu tellus sit amet tortor gravida placerat. Integer sapien est, iaculis in, pretium quis, viverra ac, nunc. Praesent eget sem vel leo ultrices bibendum. Aenean faucibus. Morbi dolor nulla, malesuada eu, pulvinar at, mollis ac, nulla. Curabitur auctor semper nulla. Donec varius orci eget risus. Duis nibh mi, congue eu, accumsan eleifend, sagittis quis, diam. Duis eget orci sit amet orci dignissim rutrum.

7.3 TESTAUFBAU

Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Ut purus elit, vestibulum ut, placerat ac, adipiscing vitae, felis. Curabitur dictum gravida mauris. Nam arcu libero, nonummy eget, consectetuer id, vulputate a, magna. Donec vehicula augue eu neque. Pellentesque habitant morbi tristique senectus et netus et malesuada fames ac turpis egestas. Mauris ut leo. Cras viverra metus rhoncus sem. Nulla et lectus vestibulum urna fringilla ultrices. Phasellus eu tellus sit amet tortor gravida placerat. Integer sapien est, iaculis in, pretium quis, viverra ac, nunc. Praesent eget sem vel leo ultrices bibendum. Aenean faucibus. Morbi dolor nulla, malesuada eu, pulvinar at, mollis ac, nulla. Curabitur auctor semper nulla. Donec varius orci eget risus. Duis nibh mi, congue eu, accumsan eleifend, sagittis quis, diam. Duis eget orci sit amet orci dignissim rutrum.

8

ERGEBNISSE & FAZIT

Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Ut purus elit, vestibulum ut, placerat ac, adipiscing vitae, felis. Curabitur dictum gravida mauris. Nam arcu libero, nonummy eget, consectetuer id, vulputate a, magna. Donec vehicula augue eu neque. Pellentesque habitant morbi tristique senectus et netus et malesuada fames ac turpis egestas. Mauris ut leo. Cras viverra metus rhoncus sem. Nulla et lectus vestibulum urna fringilla ultrices. Phasellus eu tellus sit amet tortor gravida placerat. Integer sapien est, iaculis in, pretium quis, viverra ac, nunc. Praesent eget sem vel leo ultrices bibendum. Aenean faucibus. Morbi dolor nulla, malesuada eu, pulvinar at, mollis ac, nulla. Curabitur auctor semper nulla. Donec varius orci eget risus. Duis nibh mi, congue eu, accumsan eleifend, sagittis quis, diam. Duis eget orci sit amet orci dignissim rutrum.

9 DISKUSSION

Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Ut purus elit, vestibulum ut, placerat ac, adipiscing vitae, felis. Curabitur dictum gravida mauris. Nam arcu libero, nonummy eget, consectetuer id, vulputate a, magna. Donec vehicula augue eu neque. Pellentesque habitant morbi tristique senectus et netus et malesuada fames ac turpis egestas. Mauris ut leo. Cras viverra metus rhoncus sem. Nulla et lectus vestibulum urna fringilla ultrices. Phasellus eu tellus sit amet tortor gravida placerat. Integer sapien est, iaculis in, pretium quis, viverra ac, nunc. Praesent eget sem vel leo ultrices bibendum. Aenean faucibus. Morbi dolor nulla, malesuada eu, pulvinar at, mollis ac, nulla. Curabitur auctor semper nulla. Donec varius orci eget risus. Duis nibh mi, congue eu, accumsan eleifend, sagittis quis, diam. Duis eget orci sit amet orci dignissim rutrum.

9.1 WIEDERAUFNAHME THESE TEIL 1: EIGNUNG ALS IOT-BACKBONE?

Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Ut purus elit, vestibulum ut, placerat ac, adipiscing vitae, felis. Curabitur dictum gravida mauris. Nam arcu libero, nonummy eget, consectetuer id, vulputate a, magna. Donec vehicula augue eu neque. Pellentesque habitant morbi tristique senectus et netus et malesuada fames ac turpis egestas. Mauris ut leo. Cras viverra metus rhoncus sem. Nulla et lectus vestibulum urna fringilla ultrices. Phasellus eu tellus sit amet tortor gravida placerat. Integer sapien est, iaculis in, pretium quis, viverra ac, nunc. Praesent eget sem vel leo ultrices bibendum. Aenean faucibus. Morbi dolor nulla, malesuada eu, pulvinar at, mollis ac, nulla. Curabitur auctor semper nulla. Donec varius orci eget risus. Duis nibh mi, congue eu, accumsan eleifend, sagittis quis, diam. Duis eget orci sit amet orci dignissim rutrum.

9.2 WIEDERAUFNAHME THESE TEIL 2: TECHNISCHE ANFORDERUNGEN IMMER GLEICH?

Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Ut purus elit, vestibulum ut, placerat ac, adipiscing vitae, felis. Curabitur dictum gravida mauris. Nam arcu libero, nonummy eget, consectetuer id, vulputate a, magna. Donec vehicula augue eu neque. Pellentesque habitant morbi tristique senectus et netus et malesuada fames ac turpis egestas. Mauris ut leo. Cras viverra metus rhoncus sem. Nulla et lectus vestibulum urna fringilla ultrices. Phasellus eu tellus sit amet tortor gravida placerat. Integer sapien est, iaculis in, pretium quis, viverra ac, nunc. Praesent eget sem vel leo ultrices bibendum. Aenean faucibus. Morbi dolor nulla, malesuada eu, pulvinar at,

mollis ac, nulla. Curabitur auctor semper nulla. Donec varius orci eget risus. Duis nibh mi, congue eu, accumsan eleifend, sagittis quis, diam. Duis eget orci sit amet orci dignissim rutrum.

10

AUSBLICK

Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Ut purus elit, vestibulum ut, placerat ac, adipiscing vitae, felis. Curabitur dictum gravida mauris. Nam arcu libero, nonummy eget, consectetuer id, vulputate a, magna. Donec vehicula augue eu neque. Pellentesque habitant morbi tristique senectus et netus et malesuada fames ac turpis egestas. Mauris ut leo. Cras viverra metus rhoncus sem. Nulla et lectus vestibulum urna fringilla ultrices. Phasellus eu tellus sit amet tortor gravida placerat. Integer sapien est, iaculis in, pretium quis, viverra ac, nunc. Praesent eget sem vel leo ultrices bibendum. Aenean faucibus. Morbi dolor nulla, malesuada eu, pulvinar at, mollis ac, nulla. Curabitur auctor semper nulla. Donec varius orci eget risus. Duis nibh mi, congue eu, accumsan eleifend, sagittis quis, diam. Duis eget orci sit amet orci dignissim rutrum.

Teil II

APPENDIX

A

APPENDIX: ANFORDERUNGEN

Story	Description	Task	Description	Requirement	Description	Class Level-1	Class Level-2	Non-Functional	Non-Functional Subcategory
Story A1 "Agieren auf Plattform"	"Als Akteur möchte ich in meiner Rolle als [X] auf der Plattform agieren."	Task A1.1	Akteuere können sich an der Plattform registrieren und gemäß ihrer Rolle miteinander agieren.	Requirement A1.1.1	Jeder Akteur auf der Plattform kann eindeutig identifiziert werden.	Software	NonFunctional	Security	Accountability
				Requirement A1.1.2	Ein Akteur registriert sich und meldet sich auf der Plattform an, bevor er dort agieren kann.	Software	Functional		
				Requirement A1.1.3	Ein Akteur agiert immer mit einer bestimmten Rolle auf der Plattform: Manufacturer, Customer, Supplier, Service-Provider oder Gerät. Ein Akteur kann mehrere Rollen haben.	Software	Functional		
				Requirement A1.1.4	Es existiert eine Oberfläche, auf die jeder Akteur Zugriff hat. Dort kann er sich registrieren und anmelden.	Software	Functional		
				Requirement A1.1.5	Ein Akteur hat eine (mehrere) verifizierte Rolle(n).	Software	Functional		
		Task A1.2	Akteure können über die Plattform miteinander kommunizieren.	Requirement A1.2.1	Akteure kommunizieren über die Plattform.	Software	Functional		
				Requirement A1.2.2	Die Kommunikation der beteiligten Akteure wird sofort übermittelt.	Software	NonFunctional	PerformanceEfficiency	Time behavior
				Requirement A1.2.3	Die Kommunitkation zwischen den Akteuren ist nachvollziehbar und eindeutig zuordenbar.	Software	NonFunctional	Security	Accountability
				Requirement A1.2.4	Die Kommunitkation zwischen den Akteuren kann nicht gelöscht oder manipuliert werden.	Software	NonFunctional	Security	Integrity
				Requirement A1.2.5	Akteure können nur den Inhalt ihrer eigenen Nachrichten einsehen.	Software	NonFunctional	Security	Confidentiality
		Task A1.3	Akteure können über die Plattform Verträge miteinander abschließen.	Requirement A1.3.1	Akteure schließen Verträge über die Plattform ab.	Software	Functional		
				Requirement A1.3.2	Verträge sind rechtlich bindend.	Software	Functional		
				Requirement A1.3.3	Akteure können Verträge ablehnen oder annehmen.	Software	Functional		
				Requirement A1.3.4	Es existiert eine Oberfläche, auf die jeder Akteur Zugriff hat. Dort kann er Vertragsanfragen erstellen. Auf der Empfängerseite muss eine Oberfläche existieren, die diese Anfragen anzeigt.	Software	Functional		
				Requirement A1.3.5	Der Vertragsgegenstand kann nicht durch Dritte manipuliert werden.	Software	NonFunctional	Security	Integrity
				Requirement A1.3.6	Akteure haben Zugriff auf alle Vertragsinformationen.	Software	Functional		
				Requirement A1.3.7	Akteure können komplexe Vertragskonstrukte umsetzen. Verträge haben einen Status. Diese können "Aktiv", "in Erzeugung" oder "Inaktiv" sein.	Software	Functional		
				Requirement A1.3.8	Akteure können ihre Verträge im Nachhinein ändern.	Software	Functional		
				Requirement A1.3.9	Akteure können nur ihre eigenen Verträge ändern.	Software	NonFunctional	Security	Confidentiality
				Requirement A1.3.10	Eine Vertragsänderung bedarf der Zustimmung aller beteiligten Akteure.	Software	Functional		
				Requirement A1.3.11	Erbrachte Leistungen werden kostenpflichtig verrechnet.	Software	Functional		
		Task A1.4	Akteure benötigen grafische Oberflächen zum Agieren auf der Plattform.	Requirement A1.4.1	Es existiert eine Oberfläche, auf die der Akteur Zugriff hat. Dort hat er eine Übersicht über alle seiner Verträge sowie aggregierte Informationen wie Anzahl aller Verträge, Kontostand, etc. Es werden ebenfalls angebotene Verträge angezeigt, die angenommen oder abgelehnt werden können.	Software	Functional		
				Requirement A1.4.2	Es existiert eine Oberfläche, auf die der Akteur Zugriff hat. Dort hat er eine detaillierte Übersicht über einen seiner Verträge und kann sich Detailinformationen dazu ansehen. Außerdem sieht er eine Übersicht über alle Nachrichten, die mit diesem Vertrag in Verbindung stehen und den Vertrag bearbeiten.	Software	Functional		
Story M1 "Geräte vermieten"	"Als Manufacturer möchte ich meine Geräte über die Plattform vermieten können, um meinen Umsatz zu steigern."	Task M1.1	Die Geräte können auf der Plattform vermietet werden.	Requirement M1.1.1	Ein Gerät ist Eigentum eines Manufacturers.	Software	Functional		
				Requirement M1.1.1	Ein Manufacturer kann beliebig viele Geräte besitzen und über die Plattform vermieten.	Software	Functional		
		Task M1.2	Die Geräte benötigen Sensoren, um Fehler und Defekte zu detektieren.	Requirement M1.2.1	Geräte können Fehlerzustände detektieren. Tritt ein Fehler auf, wird dieser dem Customer über ein Display angezeigt.	Software	Functional		
				Requirement M1.2.2	Geräte können Defekte detektieren. Tritt ein Fehler auf, wird dieser über die Plattform an den Service-Provider (Service-Vertrag) gemeldet.	Software	Functional		
				Requirement M1.2.3	Ein Defekt (Defektes Mahlwerk, undichte Anschlüsse, etc.) hindert das Gerät am Durchführen seiner Tätigkeit und muss durch einen Service-Provider behoben werden.	Software	Functional		
				Requirement M1.2.4	Ein Fehlerzustand (Leerer Wasserbehälter, geöffnete Abdeckung, etc.) hindert das Gerät am Durchführen seiner Tätigkeit und kann meistens durch den Customer behoben werden.	Software	Functional		
Story M2 "Verträge erzeugen"	"Als Manufacturer möchte ich in der Lage sein, Verträge anzulegen, um meine Geräte über die Plattform vermieten zu können."	Task M2.1	Ein Manufacturer kann Verträge erzeugen.	Requirement M2.1.1	Es existiert eine Oberfläche, auf die der Manufacturer Zugriff hat. Dort kann er Mietverträge erzeugen und als Antwort auf seine Mietanfrage an den Customer senden.	Software	Functional		
				Requirement M2.1.2	Es existiert eine Oberfläche, auf die der Manufacturer Zugriff hat. Dort kann er alle Service-Provider und deren Dienstleistungen einsehen, Service-Verträge erzeugen und an einen Service-Provider senden.	Software	Functional		
				Requirement M2.1.3	Es existiert eine Oberfläche, auf die der Manufacturer Zugriff hat. Dort kann er alle Supplier einsehen, Lieferverträge erzeugen und an den Supplier senden.	Software	Functional		
Story M3 "Verträge abrechnen"	"Als Manufacturer benötige ich eine korrekte, nutzungsabhängige und automatische Abrechnung der vermieteten Geräte, die regelmäßig aktualisiert wird, sowie der erbrachten Dienstleistungen, um den Umsatz aufrecht zu erhalten."	Task M3.1	Die Geräte müssen den Verbrauch detektieren.	Requirement M3.1.1	Geräte detektieren den Verbrauch (Kaffeemaschine: Anzahl Kaffees) und senden diesen an die Plattform.	Software	Functional		
		Task M3.2	Die Geräte müssen die Verbrauchsdaten an die Platform melden.	Requirement M3.2.1	Geräte senden den Verbrauch nach Fertigstellung des Produktes sofort an die Plattform.	Software	NonFunctional	PerformanceEfficiency	Time behavior
				Requirement M3.2.2	Die Verbrauchsdaten der Geräte können nicht manipuliert werden.	Software	NonFunctional	Security	Integrity
Story C1 "Ansicht verfügbarer Geräte"	"Als Customer möchte ich verfügbare Haushaltsgeräte angezeigt bekommen, um das passende Gerät mieten zu können."	Task C1.1	Es muss eine Auflistung aller verfügbarer (mietbarer) Geräte existieren.	Requirement C1.1.1	Es existiert eine Oberfläche, auf die der Customer Zugriff hat. Dort hat er die Möglichkeit, alle verfügbaren Geräte aufzulisten.	Software	Functional		
		Task C1.2	Es müssen alle für den Customer relevaten Informationen über das Gerät vorhanden und einsehbar sein.	Requirement C1.2.1	Der Customer hat Zugriff auf eine detaillierte Beschreibung des Geräts.	Software	Functional		
		Task C2.1	Das Gerät muss die Reinigung / Wartung durch den Customer detektieren können.	Requirement C2.1.1	Geräte detektieren eine Reinigung (Produktbehälter leeren, etc.).	Software	Functional		
				Requirement C2.1.2	Geräte detektieren eine Wartung (Entkalken, etc.).	Software	Functional		

Story C2 "Geräte warten"	"Als Customer möchte ich die gemieteten Geräte reinigen und warten können, um dafür vom Hersteller eine Gutschrift auf mein Vertragskonto zu erhalten."	Task C2.2	Es muss sichergestellt werden, dass eine Reinigung / Reparatur dem Gerät bzw. dessen Sensoren nicht vorgespielt werden kann.	Requirement C2.2.1	Detektierte Reinigungen / Wartungen können nicht manipuliert oder dem Gerät vorgespielt werden.	Software	NonFunctional	Security	Integrity
		Task C2.3	Der Customer muss vom Gerät eindeutig identifiziert werden können.	Requirement C2.3.1	Das Gerät kann den Customer indetifizieren.	Software	Functional		
		Task C2.4	Die Reinigung / Wartung muss an die Plattform übertragen werden und gemäß des Vertrages abgerechnet werden.	Requirement C2.4.1	Das Gerät sendet detektierte Reinigungen / Wartungen an die Plattform.	Software	Functional		
				Requirement C2.4.2	Das Gerät sendet detektierte Reinigungen / Wartungen nach Abschluss direkt an die Plattform.	Software	NonFunctional	PerformanceEfficiency	Time behavior
Story C3 "Einfache Bedienbarkeit der Plattform"	"Als Customer möchte ich eine intuitve, einfach zu bedienende Oberfläche, um mich gut auf der Plattform zurechtzufinden."	Task C3.1	Ein Customer muss zunächst mit der Plattform bekannt gemacht werden.	Requirement C3.1.1	Ein Customer erhält eine initiale Einführung über die Plattform bei der ersten Anmeldung.	System	Transition		
				Requirement C3.1.2	Die Funktionen der Plattform sind für den Customer schnell erlernbar und leicht verständlich.	Software	NonFunctional	Usability	Learnability
		Task C3.2	Die Plattform muss optisch ansprechend sein.	Requirement C3.2.1	Die Plattform ist für den Customer optisch ansprechend.	Software	NonFunctional	Usability	User interface aesthetics
Story SP1 "Dienstleitungen bereitstellen"	"Als Service-Provider möchte ich meine Angebotspalette auf der Plattform anbieten können."	Task SP1.1	Ein Service-Provider muss seine angebotenen Dienstleitungen auf der Plattform eingeben können.	Requirement SP1.1.1	Es existiert eine Oberfläche, auf die der Service-Provider Zugriff hat. Dort kann er alle Dienstleitungen, die er anbietet, sowie Detailinformationen, wie zum Beispiel Kosten der Dienstleitung, eintragen und damit auf der Plattform verfügbar machen.	Software	Functional		
				Requirement SP1.1.2	Es existiert eine Oberfläche, auf die der Service-Provider Zugriff hat. Dort kann er bereits angebotene Dienstleitungen editieren oder löschen.	Software	Functional		
Story SP2 "Service-Aufträge abschließen"	"Als Service-Provider möchte ich nach der Durchführung der Wartung diese mit meinem Smartphone am Gerät bestätigen, um den Service-Auftrag abzuschließen."	Task SP2.1	Der Service-Provider kann mit dem Gerät per Smartphone kontaktlos kommunizieren.	Requirement SP2.1.1	Gerät und Service-Provider können miteinander kommunizieren.	Software	Functional		
				Requirement SP2.1.2	Ein Gerät kann den Service-Provider eindeutig identifizieren.	Software	Functional		
		Task SP2.2	Es wird eine App benötigt, mit der der Service-Provider seine Identität und die durchgeführte Wartung am Gerät bestätigen kann.	Requirement SP2.2.1	Es existiert eine Smartphone-Oberfläche, auf die der Service-Provider Zugriff hat. Darüber kann er Wartungen abschließen.	Software	Functional		
				Requirement SP2.2.2	Ein Service-Provider kann einen Service-Auftrag starten und beenden, wenn er sich in der Nähe des Gerät befindet.	Software	Functional		
Story SEC1 "Sichere Zahlungsabwicklung"	"Als IT-Security-Beauftragter möchte ich sichergestellt wissen, dass die Zahlungsabwicklung auf der Plattform sicher und voll funktionsfähig ist."	Task SEC1.1	Es muss sichergestellt werden, dass Geldtransfers vom Sender an den Empfänger durchgeführt werden.	Requirement SEC1.1.1	Geldtransfers werden auf der Plattform geloggt.	Software	Functional		
				Requirement SEC1.1.2	Bei Nicht-Ausführung von Geld- und Nachrichtentransfers werden die Parteien benachrichtigt.	Software	NonFunctional	Security	Integrity
				Requirement SEC1.1.3	Die Plattform prüft Geldtransfers vor Ausführung.	Software	Functional		
		Task SEC1.2	Es muss sichergestellt werden, dass Sender und Empfänger des Geldes eindeutig identifizierbar sind.	Requirement SEC1.2.1	Jeder Akteuer besitzt eine (mehrere) eindeutige Kontonummer(n).	Software	NonFunctional	Security	Accountability
				Requirement SEC1.2.2	Konten sind zugriffsgeschützt.	Software	NonFunctional	Security	Authenticity
Story SEC2 "Sichere Kommunikation und signierte Nachrichten"	"Als IT-Security-Beauftragter möchte ich eine verschlüsselte Kommunikation mit der Plattform, damit meine Daten nicht in die Hände von Dritten gelangen."	Task SEC2.1	Die Kommunikation zwischen Akteuren und der Plattform muss verschlüsselt werden.	Requirement SEC2.1.1	Sämtliche Verbindungen sind per SSL/TLS zu verschlüsselt.	System	Quality		
		Task SEC2.2	Aktuelle Sicherheitsstandards müssen verwendet werden.	Requirement SEC2.2.1	Es können Passwortregeln hinterlegt werden. Die Einhaltung dieser Regeln wird überprüft.	System	Quality		
Story SEC3 "Manipulationssicherheit"	"Als IT-Security-Beauftragter möchte ich eine manipulationssichere Plattform, um die Integrität und Echtheit der Daten zu gewährleisten."	Task SEC3.1	Es müssen Vorkehrungen gegen Manipulationen getroffen werden.	Requirement SEC3.1.1	Der Zugang zu den Backend-Systemen wird protokolliert und nur Berechtigten gestattet.	Software	NonFunctional	Security	Accountability
				Requirement SEC3.1.2	Manipulationen werden durch den Einsatz kryptographischer Methoden verhindert.	Software	NonFunctional	Security	Integrity
		Task SEC3.2	Es ist nachvollziehbar, wer wann auf die Plattform zugegriffen hat.	Requirement SEC3.2.1	Alle Aktivitäten auf der Plattform werden geloggt.	Software	NonFunctional	Security	Accountability
Story BD1 "Geschäftsmodell"	"Als Business-Developer möchte ich mit dem Geschäftsmodell Pay-As-You-Use einen größeren Kundenbereich ansprechen und den Umsatz steigern."	Task BD1.1	Das Pay-As-You-Use Abrechnungsmodell muss in einem Vertrag abgebildet werden können.	Requirement BD1.1.1	Vermietete Geräte werden nach dem Pay-As-You-Use Prinzip abgerechnet.	System	Business		
		Task BD1.2	Es ist vertraglich festgelegt, welcher Akteur für welche Dienstleistung wie viel Geld bezahlt bzw. erhält.	Requirement BD1.2.1	Customer bezahlen jede verbrauchte Einheit (z.B. pro Tasse Kaffee) des gemieteten Geräts an den Manufacturer. Die genauen Kosten sind vom Gerät abhängig und werden durch den Manufacturer festgelegt.	Software	Functional		
				Requirement BD1.2.2	Manufacturer bezahlen Customer für jede durchgeführte Wartung des vermieteten Geräts. Die Höhe der Zahlung ist im Mietvertrag geregelt.	Software	Functional		
				Requirement BD1.2.3	Manufacturer bezahlen Service-Provider für jede erbrachte Service-Leistung. Die Höhe der Zahlung ist im Service-Vertrag geregelt.	Software	Functional		
				Requirement BD1.2.4	Verträge sind individuell gestaltbar.	Software	Functional		
				Requirement BD1.2.5	Manufacturer bezahlen Supplier für jede erbrachte Lieferung. Die Höhe der Zahlung ist im Liefervertrag geregelt.	Software	Functional		
Story BD2 "Plattform für Partner"	"Als Business-Developer möchte ich eine Plattform, die für Partner wie Service-Provider oder Supplier leicht zugänglich ist."	Task BD2.1	Partner agieren auf der Plattform.	Requirement BD2.1.1	Auf der Plattform agieren Geschäftspartner (Service-Provider, Supplier, ect.).	System	Business		
				Requirement BD2.1.2	Die Nutzung der Plattform ist intuitiv und schnell erlernbar. Umfangreiche Mitarbeiterschulungen zur Benutzung der Plattform sind nicht notwendig.	System	Transition		
		Task BD2.2	Partner müssen als solche identifiziert sein.	Requirement BD2.2.1	Der Prozess zur Prüfung, dass es sich z.B. bei einem Service-Provider auch tatsächlich um einen solchen handelt, ist benutzerfreundlich und so schnell und einfach wie möglich umsetzbar.	System	Stakeholder		
Story BD3 "Hersteller-übergreifende Plattform"	"Als Business-Developer möchte ich in Zukunft eine Plattform schaffen, auf der Hersteller unterschiedlicher Branchen ihre Produkte nach dem Pay-As-You-Use Prinzip vermieten können, um dem Kunden eine breitere Produktpalette zu bieten."	Task BD3.1	Die Plattform muss zukünftig weitere Hersteller zulassen, um ein ganzes Ökosystem von Geräten aller Art dem Customer zugänglich zu machen.	Requirement BD3.1.1	Es können in Zukunft weitere Hersteller auf der Plattform ihre Geräte zur Miete anbieten.	System	Business		
		Task BD3.2	Ein Gerät muss auf der Plattform generisch repräsentiert werden und darf nicht von einem bestimmten Produkttyp oder Hersteller abhängig sein.	Requirement BD3.2.1	Die Plattform kann Geräte unterschiedlicher Art anbinden und abrechnen.	Software	NonFunctional	Compatibility	Interoperability
		Task BD3.3	Ein Vertrag muss auf der Plattform generisch repräsentiert werden und darf nicht von einem bestimmten Produkttyp oder Hersteller abhängig sein.	Requirement BD3.3.1	Die Plattform kann unterschiedliche Vertragsarten abbilden und abrechnen.	Software	NonFunctional	Compatibility	Interoperability
Story BD4 "Vertragsgestaltung"	"Als Business-Developer möchte ich die Plattform dazu nutzen, künftig andere Vertragsarten umzusetzen, um weitere Geschäftsfelder und Kunden zu gewinnen."	Task BD4.1	Die Einbindung von Verträgen sowie deren Struktur muss möglichst modular und unabhängig geschehen, damit später andere Verträge leicht integriert werden können.	Requirement BD4.2.1	Ein Abrechnungsmodell wird in einem Vertrag abgebildet.	System	Business		
				Requirement BD4.2.2	Einzelne Bestandteile der Plattform beeinflussen sich gegenseitig nicht und können leicht ausgetauscht werden.	Software	NonFunctional	Maintainability	Modularity
Story SA1 "Modularer Aufbau"	"Als System-Architekt möchte ich eine modular aufgebaute Plattform, damit diese später möglichst einfach erweitert	Task SA1.1	Funktionalitäten werden in Modulen gekapselt und als Service bereitgestellt.	Requirement SA1.1.1	Ein Software-Modul wird unabhängig von anderen Modulen bereitgestellt und gewartet.	Software	Process		
Story SA2	"Als System-Architekt möchte ich SPOFs und Datenverlust vermeiden,	Task SA2.1	Die Plattform kann im Falle eines Crashes alle Daten konsistent halten.	Requirement SA2.1.1	Bei Ausfall einzelner Komponenten gehen keine Daten verloren.	Software	NonFunctional	Security	Integrity
				Requirement SA2.1.2	Bei Ausfall einzelner Komponenten entstehen keine Dateninkonistenzen.	Software	NonFunctional	Security	Integrity

"Redundanz"	damit das System auch im Fehlerfall weiter funktionsfähig ist."	Task SA2.2	Die Plattform darf keinen SPOF haben, damit im Falle eines Crashes das Gesamtsystem weiterhin lauffähig ist.	Requirement SA2.2.1	Die Plattform ist in der Lage, die Kommunikation und Datenverarbeitung bei bis zu 10.000 Endgeräten durchzuführen.	Software	NonFunctional	Reliability	Availability
				Requirement SA2.2.2	Die Plattform ist nicht von einer einzelnen Komponente (SPOF) abhängig.	Software	NonFunctional	Reliability	Availability
Story P1 "Deployment & Testing"	"Als Betreiber der Plattform möchte ich eine automatisierte und korrekte Bereitstellung der Plattform, um meinen Aufwand zu reduzieren."	Task P1.1	Es wird nur getestete Software bereitgestellt.	Requirement P1.1.1	Für alle Software-Komponenten existieren hinreichende Tests, die Test-Abdeckung beträgt 80%.	Software	NonFunctional	Maintainability	Testability
		Task P1.2	Die Bereitstellung ist automatisiert.	Requirement P1.2.1	Die Software wird automatisiert durch Skripte installiert und bereitgestellt.	Software	NonFunctional	Portability	Installability

LITERATUR

[1] Alain Abran und James W. Moore, Hrsg. *Guide to the Software Engineering Body of Knowledge: 2004 Version SWEBOK*. Los Alamitos, CA: IEEE Computer Society Press, 2005. ISBN: 0-7695-2330-7. URL: http://www2.computer.org/portal/web/swebok/2004guide.

[2] Mohammed Alani. *Guide to OSI and TCP/IP Models*. Jan. 2014. ISBN: 978-3-319-05152-9. DOI: 10.1007/978-3-319-05152-9.

[3] Binance. *Die Geschichte der Blockchain*. 2019. URL: https://www.binance.vision/de/blockchain/history-of-blockchain.

[4] R.J. Cloutier (Editor in Chief). *The Guide to the Systems Engineering Body of Knowledge (SEBoK)*. v.2.0. BKCASE. The Trustees of the Stevens Institute of Technology, International Council on Systems Engineering, Institute of Electrical und Electronics Engineers Computer Society, 2019.

[5] Cisco. *Internet of Things*. 2016. URL: https://www.cisco.com/c/dam/en/us/products/collateral/se/internet-of-things/at-a-glance-c45-731471.pdf.

[6] Forbes. *Blockchain's Secret 1,000 Year History*. 2018. URL: https://www.forbes.com/sites/oliversmith/2018/03/23/blockchains-secret-1000-year-history/#6871020e18d2.

[7] IEEE. "Systems and software engineering – Life cycle processes –Requirements engineering". In: *ISO/IEC/IEEE 29148:2011(E)* (2011).

[8] ISO/IEC. *ISO/IEC 25010 System and software quality models*. Techn. Ber. 2010.

[9] Iiba. *Babok: A Guide to the Business Analysis Body of Knowledge*. Bd. 3. International Institute of Business Analysis, 2015. ISBN: 9781927584026. URL: https://books.google.de/books?id=ogxTrgEACAAJ.

[10] Project Management Institute. *A Guide to the Project Management Body of Knowledge(PMBOK Guide)*. 4th. PMI global standard. Project Management Institute, 2010. ISBN: 9781933890661.

[11] W3C. *Decentralized Identifier*. 2019. URL: https://www.w3.org/TR/did-core/.

[12] Z. Zheng, S. Xie, H. Dai, X. Chen und H. Wang. "An Overview of Blockchain Technology: Architecture, Consensus, and Future Trends". In: *2017 IEEE International Congress on Big Data (BigData Congress)*. 2017, S. 557–564. DOI: 10.1109/BigDataCongress.2017.85.

[13] H. Zimmermann. "OSI Reference Model - The ISO Model of Architecture for Open Systems Interconnection". In: *IEEE Transactions on Communications* 28.4 (1980), S. 425–432. ISSN: 1558-0857. DOI: 10.1109/TCOM.1980.1094702.