HOCHSCHULE
MITTWEIDA
UNIVERSITY OF
APPLIED SCIENCES

BELEGARBEIT

Herr
Felix Hildebrandt

Educhain als universelle Blockchain für Studenten

Mittweida, 2021

Fakultät der Angewandten Computer
und Biowissenschaften

BELEGARBEIT

Educhain als universelle Blockchain für Studenten

Autor:
Mr. B.Sc.

Felix Hildebrandt

Kurs:
Architektur komplexer Softwaresysteme

Seminargruppe:
BC20w1-M

Kursleiter:
Prof. Dr.-Ing. Wilfried Schubert

Einreichung:
Mittweida, 09.07.2021

Bibliografische Beschreibung:

Hildebrandt, Felix:
Educhain als universelle Blockchain für Studenten - 2021

Mittweida, Hochschule Mittweida, Fakultät der Angewandten Computer und Biowissenschaften, Belegarbeit, 2021

Referat:

Es gibt viele revolutionäre Software- und Forschungsprojekte aus der Region, welche die Anwendungsbereiche Digitale Identitäten, IoT und Dokumentenverwaltung mit der Blockchain-Technologie verknüpfen. Es mangelt jedoch an einer Lösung, welche diese Projekte für den täglichen Lebensstil an Hochschulen und Universitäten vereint.

Educhain soll den Studenten mithilfe einer eigenen Blockchain ein Ökosystem bieten, in welchem Datennachweise, Identifikationen, Zugangsberechtigungen sowie Bezahlungen hochschul- und geräteübergreifend möglich sind.

Die Belegarbeit beschreibt die grundlegenden Architekturkonzepte welche bei der Entwicklung zum Einsatz kommen sollen. Als Prototyp dient eine Anwendung mit der Softwarebibliothek React, mit der sich Dokumente mit einem Zeitstempel versehen und Token aushändigen lassen.

Inhaltsverzeichnis

Inhaltsverzeichnis I

Abbildungsverzeichnis III

Tabellenverzeichnis IV

Abkürzungsverzeichnis VI

1 Projektbeschreibung 1

2 Problemanalyse im Studentenalltag 2

3 Vorgehen beim Erstellen der Architektur 4

4 Ausgangssituation 5

4.1 Fundamentale Entwicklungsbereiche 6

4.2 Ansprüche 7

5 Anforderungen und Use Cases 8

5.1 Anwendungsfälle 10

5.1.1 Prozesse der Benutzer und des Ökosystems 10

5.1.2 Interne und administrative Prozesse 14

5.2 Aktoren 15

5.2.1 Personen und Einrichtungen 15

5.2.2 Hardwareintegration 16

5.2.3 Softwarelösungen 16

6 Analysemodell 17

7 Architekturdokumentation 18

7.1 Dokumentation für Benutzer 18

7.2 Dokumentation für Entwickler 19

8 Architekturerstellung 21

8.1 Spezifikation der Einflussfaktoren 21

8.1.1 Produktspezifische Kenngrößen 22

8.1.2 Technische Kenngrößen 33

8.1.3 Organisatorische Kenngrößen 44

8.2 Plattformen 55

8.2.1 Software 55

8.2.2 Hardwarekomponenten 56

8.3 Entwurf und Dokumentation 57

8.3.1 Betrachtung des Projektkomplexes 57

8.3.2 Erster Entwurf 58

8.4 Szenario-basierte Bewertung 65

8.5 Phase 1 65

8.6 Phase 2 68

9 Architekturumsetzung 69

10 Exemplarische Umsetzung 70

10.1 Design Pattern 70

10.2 Prototyp 71

10.3 Installation 74

11 Zusammenfassung 75

Anlagen

Anlage 1: Anwendungsvorgehen 78

Anlage 2: Clone Factory Pattern 96

Quellenangaben 97

Hilfsmittel und Werkzeuge 99

Selbstständigkeitserklärung 101

Abbildungsverzeichnis

Abbildung 1 Vorgehen bei der Architekturerstellung 4

Abbildung 2 Komplexes Use Case Übersicht 8

Abbildung 3 Analysemodell 17

Abbildung 4 Ablaufdiagramm: Tür öffnen 64

Abbildung 5 Utility Tree 67

Abbildung 6 Interaktion im Prototyp 69

Abbildung 7 Clone Factory Pattern 70

Abbildung 8 Vorschau: Hauptmenü 71

Abbildung 9 Vorschau: Dateinachweis erstellen 71

Abbildung 10 Vorschau: Dateinachweis erhalten 72

Abbildung 11 Vorschau: Tokenerhalt 73

Tabellenverzeichnis

Tabelle 1 Produktspezifische Kenngröße: Funktionale Anforderungen 22

Tabelle 2 Produktspezifische Kenngröße: Benutzerschnittstellen..... 23

Tabelle 3 Produktspezifische Kenngröße: Leistung I 24

Tabelle 4 Produktspezifische Kenngröße: Leistung II 25

Tabelle 5 Produktspezifische Kenngröße: Zuverlässigkeit..... 26

Tabelle 6 Produktspezifische Kenngröße: Sicherheit I..... 27

Tabelle 7 Produktspezifische Kenngröße: Sicherheit II..... 28

Tabelle 8 Produktspezifische Kenngröße: Sicherheit III..... 29

Tabelle 9 Produktspezifische Kenngröße: Testbarkeit..... 30

Tabelle 10 Produktspezifische Kenngröße: Agilität..... 31

Tabelle 11 Produktspezifische Kenngröße: Portierbarkeit 32

Tabelle 12 Technische Kenngröße: Hardware I..... 33

Tabelle 13 Technische Kenngröße: Hardware II..... 34

Tabelle 14 Technische Kenngröße: Softwaretechnologie I..... 34

Tabelle 15 Technische Kenngröße: Softwaretechnologie II..... 35

Tabelle 16 Technische Kenngröße: Softwaretechnologie III..... 36

Tabelle 17 Technische Kenngröße: Softwaretechnologie IV..... 37

Tabelle 18 Technische Kenngröße: Softwaretechnologie V..... 38

Tabelle 19 Technische Kenngröße: Architekturtechnologie I..... 39

Tabelle 20 Technische Kenngröße: Architekturtechnologie II..... 40

Tabelle 21 Technische Kenngröße: Architekturtechnologie III 41

Tabelle 22 Technische Kenngröße: Standards I 42

Tabelle 23 Technische Kenngröße: Standards II 43

Tabelle 24 Organisatorische Kenngröße: Management I 44

Tabelle 25 Organisatorische Kenngröße: Management II 45

Tabelle 26 Organisatorische Kenngröße: Management III 46

Tabelle 27 Organisatorische Kenngröße: Mitarbeiter I 47

Tabelle 28 Organisatorische Kenngröße: Mitarbeiter II 48

Tabelle 29 Organisatorische Kenngröße: Prozess und Entwicklungsumgebung I 49

Tabelle 30 Organisatorische Kenngröße: Prozess und Entwicklungsumgebung II 50

Tabelle 31 Organisatorische Kenngröße: Prozess und Entwicklungsumgebung III 51

Tabelle 32 Organisatorische Kenngröße: Entwicklungszeitplan I 52

Tabelle 33 Organisatorische Kenngröße: Entwicklungszeitplan II 53

Tabelle 34 Organisatorische Kenngröße: Entwicklungsbudget 54

Abkürzungsverzeichnis

ABI	Application Binary Interface
BCCM	Blockchain Competence Center Mittweida
BWK	Backup Wallet Keyset
C	Programmen in der Programmiersprache C
dApp	dezentrale App
DE	Designentscheidung
DWK	Device Wallet Keyset
EDUC	Educhain Protocol-Token
EDUT	Educhain Euro-Token
FE	Finanzentscheidung
FT	Fungible Token
HTTPS	Hypertext Transfer Protocol Secure Web-Protokoll
IDE	integrierte Entwicklungsumgebung
ITK	Informations- und Kommunikationstechnik
JS	Programme in der Programmiersprache JavaScript
NFT	Non-Fungible Token
PoS	Proof of Stake
PoW	Proof of Work
SC	Smart Contract
SCA	Smart Contract Account
SOL	Programme in der Programmiersprache Solidity
TX	Transaktion
ZE	Zeitentscheidung

1 Projektbeschreibung

Im Fach E-Entrepreneurship & Digital Innovation Management des Masterstudiengangs Blockchain & Distributed Ledger Technologies an der Hochschule Mittweida wurde im Wintersemester 2020 die Startup-Idee Educhain präsentiert, welche sich mithilfe der Blockchain-Technologie hochschulübergreifend in den Alltag von Studenten integrieren soll.

Ziel des Projektes ist es, Ausweise von Studenten hochschul- und geräteübergreifend nutzen zu können, den Datenaustausch zwischen Studenten und der Professur sekundenschnell zu validieren, Spinde und Schlösser digital anzusteuern sowie die Bezahlung von Diensten an Hochschulen und Universitäten wie der Mensa sicher vom Smartphone aus durchzuführen.

Mit einem Netzwerk von teilnehmenden Hochschulen und Universitäten wird den Studierenden die Möglichkeit geboten, eine große Bandbreite von Services externer Universitäten nutzen zu können. Dieses Prinzip ähnelt der schon heute weit verbreiteten Lösung eduroam, welche den hochschulübergreifenden Internetzugang gewährleistet.

Im Sommersemester 2021 wird Educhain mit Hilfe der Kurse E-Entrepreneurship & Digital Innovation Management II sowie Architektur komplexer Softwaresysteme erstmalig ins Leben gerufen. Als Prototyp dient eine Web-Anwendung welche die Möglichkeit bietet Dokumente zu hashen und Token auszuhändigen. Passende Schnittstellen sollen genutzt werden, um die aus der Region stammenden Softwarelösungen ID-Ideal [5], ECHT! [6] und IN3 [7] einzubinden. Dadurch erweitert sich das Ökosystem um Identitätsmanagement, Dokumentenvalidierung sowie um die Interaktion mit digitalen Schlössern und Spinden.

2 Problemanalyse im Studentenalltag

Studenten an Hochschulen und Universitäten besitzen bisher einen Studentenausweis für das Bestätigen ihrer Identität in der analogen Welt. Außerdem nutzten sie eine eigene E-Mail-Adresse um Nachrichten auszutauschen oder sich bei Onlinediensten mit Studentenrabatten auszuweisen. Die Mailadresse wird von der Lehreinrichtung selbst ausgestellt. Zu diesen zwei Identitäts-Artefakten haben Studenten ebenso die Möglichkeit, eine hochschuleigene App oder Onlineplattform zu verwenden, um Notenabfragen, Stundenpläne, Einschreibungen in Kurse, Prüfungstermine oder zusätzlichen Informationen für Services der Mensa abzurufen. Zertifikate oder Nachweise sind nach wie vor nur analog von den entsprechenden Stellen der Fakultäten auszuhändigen.

Das aktuelle System bietet einige Nachteile bzw. Raum für Verbesserungsvorschläge um der heutigen Zeit und einer vollkommenen Digitalisierung gerecht zu werden.

Aktuell ist der Hochschulausweis immer mitzuführen, um sich auszuweisen, Türschlösser von Räumlichkeiten öffnen zu können oder in der Mensa zu bezahlen. Durch die Pandemie werden oft Listen zum Aufenthalt mithilfe des Studentenausweises benötigt, was das Mitführen eines Ausweises unumgänglich macht. Für ein einzelnes Dokument ist das zusätzliche Mitführen nicht tragisch, bei mehreren zusätzlichen Zertifikaten wie dem Impfausweis kann dies aber schnell anstrengend werden und bietet eine weitere Fehlerquelle für Sachverlust.

Nicht nur ist das Mitführen problematisch: Es kommt auch zu unzähligen Hortungen von Zertifikat-Kopien an Universitäten, Ämtern wie der BAföG-Stelle, dem Landratsamt oder Minijob-Zentren. Ebenso ist dies bei Unternehmen, bei denen sich Studenten ausweisen müssen, der Fall. Das gleiche Prinzip gilt für das Managen von Zugängen für Räumlichkeiten. Man selbst verliert schnell den Überblick, wer welche Daten von einem besitzt, verwendet oder ob diese stets aktuell sind. Das Schema verlagert die Hoheit über Daten schnell an die jeweilige Instanz, welche dann für die Sicherheit und die rechtmäßige Verwendung bürgt. Der Bürokratieaufwand ist immens hoch, beim Übermitteln von Dokumenten dauert die manuelle Prüfung und Zuordnung sehr lang- ganz zu schweigen vom Postweg, der manchmal nicht umgangen werden kann.

Diese Erkenntnis wird deutlich, wenn man bedenkt, dass Hochschulausweise nicht nur analog existieren, sondern auch lediglich an den eigenen Lehrstellen Verwendung finden. Dass die Ausweise nicht hochschulübergreifend ausgestellt werden können behindert auch die Verwaltung von Ausladsaufenthalten, bei denen in der benachbarten Universität eine Studentenidentität komplett neu angelegt werden muss. Regelmäßig kommt es vor, dass die komplette Einrichtung erst nach der Ankunft abgeschlossen werden kann. Ähnliches bezieht sich auf Studenten während Präsenzpausen oder während Homeoffice-Auflagen in

ihrer Heimatstadt. Sie können weder die Räumlichkeiten wie Bibliotheken benutzen oder mit Ihrem Ausweis dortige Studentenrabatte einlösen. Die Einschränkung bezieht sich nicht nur auf die Rabattierung, sondern auch auf die Bezahlung mit Ausweisen selbst. Trifft man sich mit Kommilitonen an einer benachbarten Universität oder deren verbündeten Einrichtungen, so kann man dort nicht mit eingezahltem Geld seines Ausweises bezahlen. Die separate Administration ist hier eine Barriere, hinter der sich auch Vergünstigungen für nicht ansässige Studierenden verriegeln. Oftmals wird beim Vorzeigen einer Studentenmitgliedschaft aus reiner Kulanz gehandelt, da Ausweisinformationen dort nicht validiert werden können.

Wenn man sich Gedanken zum aktuellen Bezahlungssystem mach, so ist dieses sehr divergent. In der lokalen Mensa kann mit dem Studentenausweis bezahlt werden, in verbündeten Klubs, an Spinden, Automaten oder bei Hochschulaktivitäten ist dies jedoch nicht der Fall. Beiträge wiederum können ausschließlich mit einer Überweisung getätigt werden. Obwohl Studenten evtl. noch über den Studentenausweis liquide wären, müssen sie immer Bargeld mit sich tragen oder eine Überweisung vom Rechner oder der Bank ausführen.

Wie schon angesprochen, sind Spinde und Vermietungsmöglichkeiten generell eine gute Lösung um Flexibilität im Alltag zu gewährleisten. Vormittags Meetings, nachmittags Präsentationen und trotzdem müssen Technik und Unterlagen nicht permanent mitgetragen werden, sondern können verwahrt werden. Der Nachteil hier ist, dass die potentiell mögliche Flexibilität durch das Bezahlschema gehindert wird. Spinde sind meist dauerhaft oder für einen ganzen Tag zu mieten und bar im Sekretariat zu bezahlen. Es ist also zwangsweise Personal notwendig und zusätzlich bezahlt man mehr, als man eventuell benötigt. Hier ist Vorausplanen essenziell. Eine variable Verlängerung oder Verkürzung um kleine Zeiträume ist aktuell noch nicht möglich.

3 Vorgehen beim Erstellen der Architektur

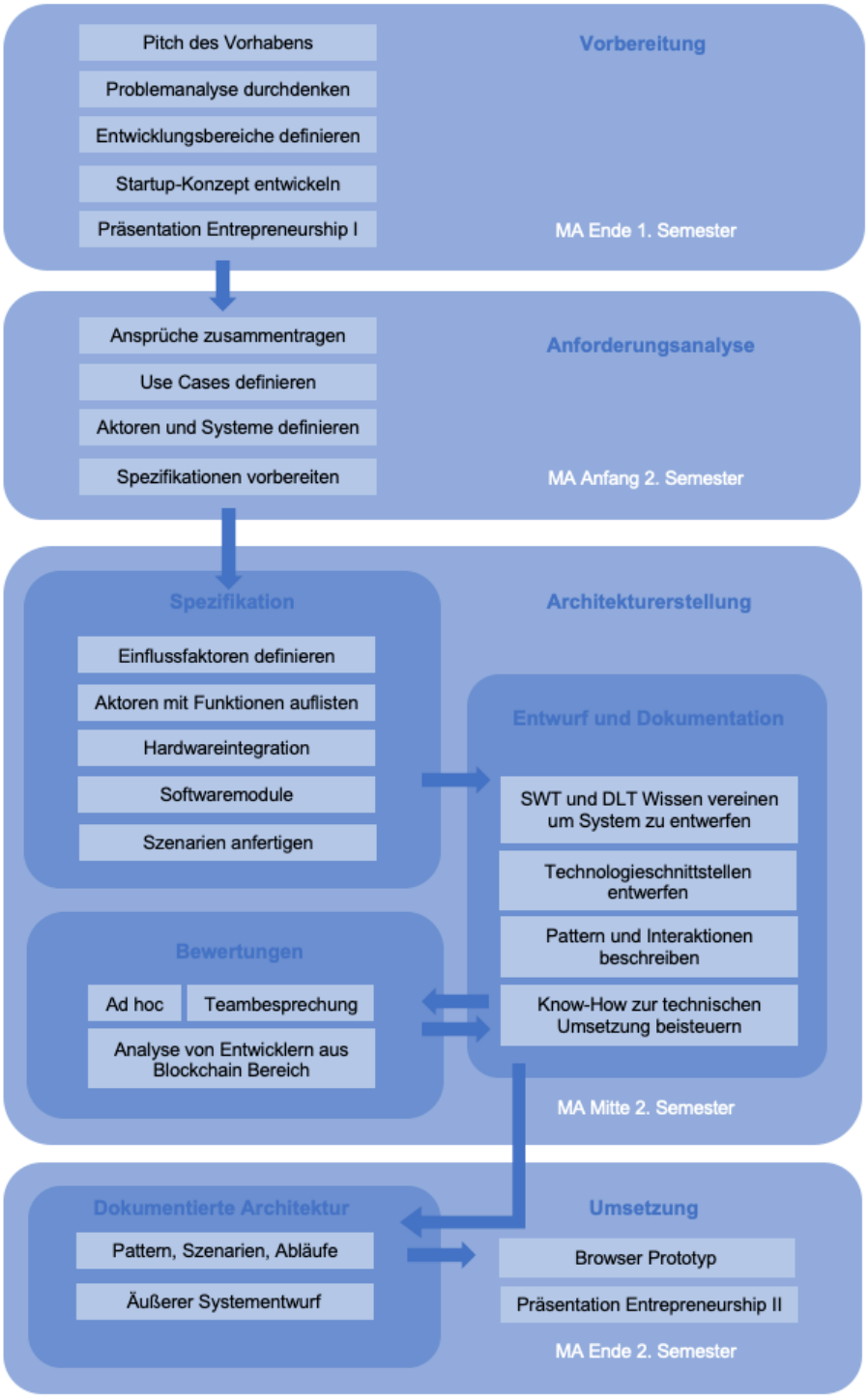

Abbildung 1 Vorgehen bei der Architekturerstellung

4 Ausgangssituation

Angesichts der aufgezeigten Probleme aus Kapitel 2 soll ein Ökosystem rund um die Benutzung von digitalen Identitäten entwickelt werden, welches die aktuelle Forschung und Blockchain-Technologien aus der Region vereint, um den Alltag von Studenten und der Professur grundlegend zu vereinfachen. Die Hochschulen und Universitäten werden hierbei als Kunden gesehen, die sich mit Kapital zur Besicherung am Netzwerk und der Serverarchitektur des Ökosystems beteiligen. Die Zeitplanung wird nicht von einer Deadline geprägt, da sich noch sehr viele Technologien und Standards in der Forschung befinden und das Ökosystem schrittweise an Funktionalität gewinnen soll. Es ist jedoch nötig, das fundamentale Blockchain-Netzwerk samt Nutzer-Accounts so früh wie möglich der Öffentlichkeit anzubieten, sodass Studenten weltweit eigene Features aufsetzen können. Die Software soll für browser- und Mobilgeräte gedacht sein- jedoch müssen auch Mikrocontroller Informationen von der Blockchain abrufen können, die durch Interaktionen mit den Geräten der Endnutzer entstehen. Die Kosten für das Bereitstellen der Software und Hardwarekomponenten sollten recht niedrig angesetzt werden, da Hochschulen und Universitäten primär keinen Gewinn erwirtschaften und sich die Bereitstellung von digitalen Ausweisen auf die Semestergebühren auswirken könnte.

Educhain besteht anfangs nur aus zwei Personen. Ein Projektleiter, welcher für die technischen Umsetzung verantwortlich ist, sowie einen Controller, der als Ansprechpartner für Kunden dient und die Projektlandschaften plant. Zusätzlich sollen 2 weitere Studenten ins Team des Projektleiters aufgenommen werden, welche Zuarbeiten und Recherche zur Implementation auf mobilen Endgeräten und Mikrocontrollern leisten. Insgesamt wurden vier konkrete Entwicklungsbereiche aufgestellt, die das Ökosystem umfassen soll.

4.1 Fundamentale Entwicklungsbereiche

Identifikation: Studenten erhalten eine digitale hochschulübergreifende Identität in Form eines Accounts, welcher auf allen Geräten nutzbar ist und Ausweise, Zertifikate, Zeugnisse oder Zugänge beinhalten kann.

Nachweise: Studenten erhalten sekundenschnell Informationen zu deren validierten Notenfreigaben, eingereichten Dokumenten, geänderten Zugängen und können diesbezügliche Nachweise mit nötigen Institutionen wie dem BAföG-Amt teilen.

Zugänge: Mittels der Nachweise ist es möglich, vom Mobilgerät digitale Schlösser aus dem IoT-Bereich zu verwenden um Zugang zu Räumlichkeiten zu erhalten. Spinde sollen ebenso verknüpft werden um Gegenstände mit variabler Dauer automatisiert einlagern zu können.

Bezahlung: Durch die Blockchain wird nicht nur Kapital für die Besicherung hinterlegt, sondern auch eine alternative, hochschulübergreifende Bezahlmöglichkeiten für die Mensa, Bars, Automaten, Ausleihen oder die Nutzung des Spinds geboten. Eingezahlte Euro werden tokenisiert und den Wallets der Studenten gutgeschrieben. Dies geschieht mit 1:1 gebildeten Rücklagen. Wird tokenisiertes Geld ausgezahlt, so werden dazugehörige Token der Hochschulkonten vermindert. Zusätzlich könnten zukünftig weitere Währungen als Anreizsysteme für besondere Leistungen, fristgerechte Abgaben oder Ähnlichem entstehen.

4.2 Ansprüche

- Jede Hochschule oder Uni kann sich mit dem Aufsetzen eines Knoten und dem Einlagern von Kapital dem Blockchain Netzwerk anschließen.
- Die Blockchain bietet einen Explorer um jegliche Vorgänge einsehen zu können.
- Mittels Forks ist es möglich, systemweite Netzwerk-Updates zu implementieren.
- Das Netzwerk muss mehrere hundert Transaktionen pro Sekunde verarbeiten.
- Über die Weboberfläche können sich Mitarbeiter der Hochschulen und Studenten Benutzer-Accounts erstellen.
- In den Accounts ist es möglich Ausweise, Zertifikate, Zeugnisse oder Zugänge zu als NFT oder Smart Contract fälschungssicher abzubilden.
- Mit Automaten an den Universitäten und Hochschulen ist es möglich, Geld auf die Accounts einzuzahlen und untereinander Zahlungen zu tätigen.
- Mittels Verknüpfung zu IoT-Geräten ist es möglich Account-Daten zu überprüfen und verknüpfte Türen und Spinde zu benutzen.
- Die Accounts und alle verknüpften Services lassen sich mittels der Weboberfläche und mobiler App geräteübergreifend und vollständig bedienen.
- Sowie Weboberfläche als auch mobiler App sind stetig aktualisierbar und Funktionieren innerhalb der Entwicklung auch mit Testnetzwerken.
- Über die App lassen sich zu lokalen Dateien Nachweise und Zeitstempel auf der Blockchain sichern, um fristgerechte Abgaben zu dokumentieren.
- Es existieren Service-Schnittstellen, sodass Drittanbieter wie Lokale oder Ämter Nutzerdaten anfragen und nachweisen können.
- Die Hardware, welche bei Automaten und IoT-Geräten benutzt wird soll für zukünftige Systeme austauschbar sein.
- Die mobile App funktioniert energiesparen, um eine Nutzung über den kompletten Unialltag ohne Einschränkungen zu ermöglichen.

5 Anforderungen und Use Cases

Aus den Ansprüchen, die aus zahlreichen Meetings im Kurs E-Entrepreneurship II sowie in Gesprächen mit Mitarbeitern des Blockchain Competence Center Mittweidas entstanden und diskutiert wurden, konnten die Anforderungen an Educhain konkretisiert werden. Hervorgehoben wurde nochmals die Benutzer- und Schnittstellenfreundlichkeit als nichtfunktionale Anforderung, um allen Studenten einen einfachen Einstieg zu ermöglichen und das Netzwerk schnell an Funktionalität wachsen zu lassen. Die funktionalen Anforderungen wurden in Form von folgenden Anwendungsfällen genauer spezifiziert.

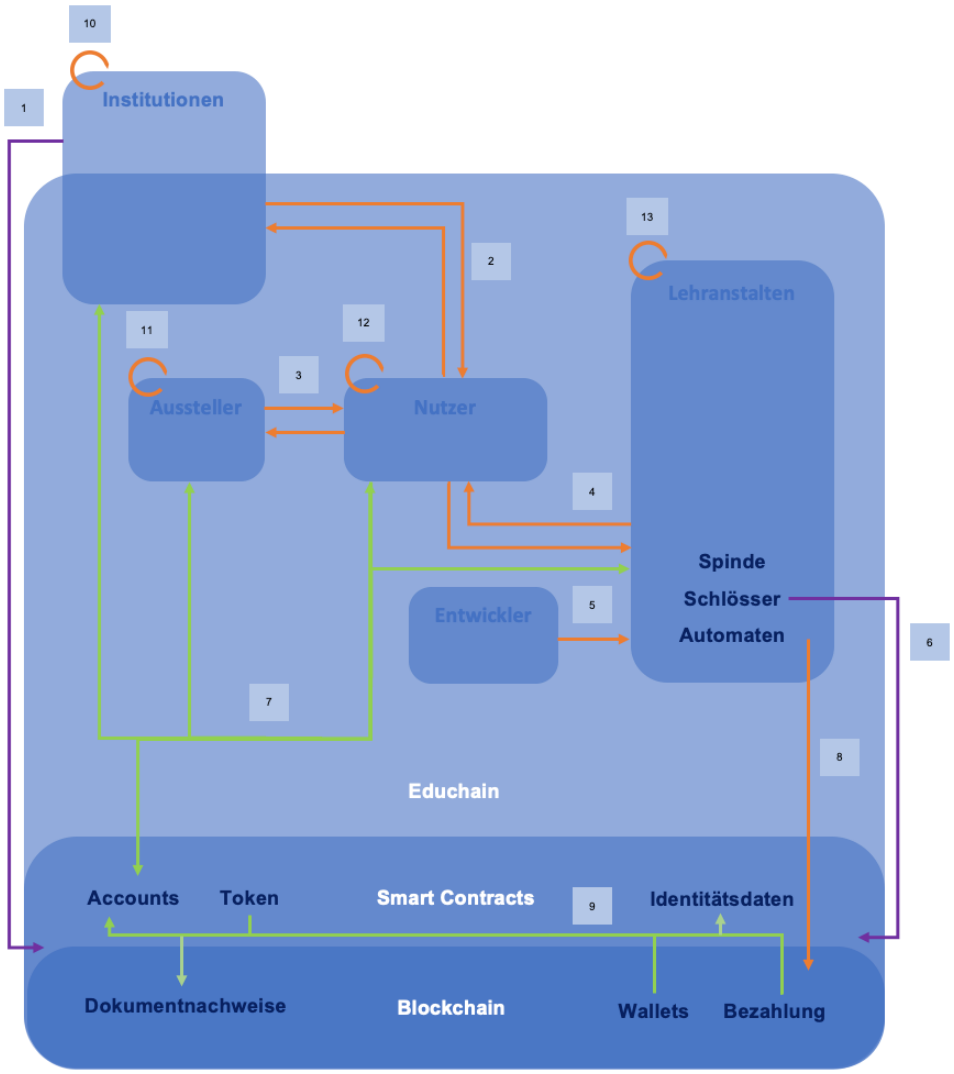

Abbildung 2 Komplexes Use Case Übersicht

In der Übersicht zu den Use Cases wird ersichtlich, dass der Nutzer im Zentrum der Interaktionen steht. Die Vorgänge wurden in externe (lila), interne (orange) sowie kontobezogene (grün) Use Cases unterteilt.

Der Hauptschwerpunkt der Logik sind die Accounts, welche auf der Blockchain in Form von Smart Contracts aufgesetzt werden.

1. Die Institution wie zum Beispiel das BAföG-Amt, Restaurants, Museen usw. verifiziert Daten auf der Blockchain ohne selbst einen Account zu besitzen um Zugänge, Rabatte oder Ähnliches auszugeben.
2. Bei Institutionen mit einem Account sind Nutzer in der Lage, Institutionen Zugriff auf ihre Identitätsdaten zu gewähren. Daraufhin können diese Informationen von den Accounts abrufen. Falls der Nutzer es wünscht, kann der Zugriff auch wieder entzogen werden.
3. Aussteller können Anfragen zum Erstellen von einem Identitätsdatensatz vom Nutzer erhalten und nach einer Prüfung solche ausstellen. Gegebenenfalls ist es möglich, dass der Aussteller Identitätsdatensätze mit speziellen Rechten versieht, sodass er diese auch wieder vom Benutzerkonto streichen kann.
4. Der Nutzer kann in Lehranstalten Geld auf seinen Account ein- und auszahlen, sowie mit seinem digitalen Ausweis Zugriff auf Räumlichkeiten erhalten. Falls installiert, kann der Benutzer auch Spinde mieten und benutzen oder Automaten vom Handy aus bedienen.
5. Entwickler binden Lehranstalten ins System ein und kennzeichnen diese transparent. Sie können ebenso Token-Automaten berechtigen, Geld ein- und auszahlen zu können.
6. Schlösser können auch extern von Hochschulen eingesetzt werden, da sie lediglich Informationen prüfen und keinen Account benötigen.
7. Nutzer, Aussteller und Lehranstalten besitzen Accounts um miteinander zu kommunizieren. Institutionen können ebenso ein Konto erstellen um Datensätze von Nutzern entgegenzunehmen.
8. Lehranstalten beteiligen sich mit Kapital am Konsensalgorithmus der Blockchain und sichern das Netzwerk mit einer Vielzahl von Server-Nodes ab.
9. Mit einem Account werden die Wallets der Geräte verknüpft, Bezahlungen getätigt sowie FT oder NFT ausgetauscht. Accounts können Identitätsdaten und Dokumentennachweise erstellen und diese mit sich verknüpfen.
10. Institutionen ist es möglich Nutzer-Accounts zu erstellen und gleiche Funktionen in ihrem eigenen Account wahrzunehmen.
11. Aussteller besitzen Nutzer-Accounts und können gleiche Funktionen in ihrem eigenen Account wahrnehmen. Aussteller sind ebenso in der Lage Daten-Masken für Identitätsdaten anzulegen, sich anzeigen zu lassen und wieder zu löschen.
12. Der Nutzer ist in der Lage einen Account zu erstellen und diesen wiederherzustellen, sich einzuloggen, seine Account-Adresse anzuzeigen, sich Zugriffsschlüssel anzulegen, anzusehen, zu speichern, zu löschen oder neue Schlüssel mit seinem Account zu verknüpfen. Der Nutzer kann seinem Account ebenso Identitätsdaten hinzufügen und diese bearbeiten, sich anzeigen lassen oder entfernen. Identitätsdaten können auch zu Ausweisen gewandelt und ebenso wieder gelöscht werden. Nutzer können zusätzlich untereinander Zahlungen tätigen. Falls Vermietungen oder Abgaben getätigt wurden, so kann sich der Nutzer diese ebenso aus dem Account heraus anzeigen.
13. Lehranstalten können Spinde, Schlösser und Token-Automaten einrichten und wieder entfernen, falls diese gewartet oder nicht mehr benötigt werden.

5.1 Anwendungsfälle

5.1.1 Prozesse der Benutzer und des Ökosystems

- **Account erstellen:** Der Student oder Mitarbeiter ruft Educhain als neue Person auf und findet seinen noch leeren Account vor. Es wird ein neuer Smart Contract samt einer Wallet mit Backup-Key erstellt.
- **Einloggen:** Der Student oder Mitarbeiter ruft Educhain mit einem verknüpften Gerät oder über aktivierte Browserwallet auf. Der Nutzer erhält Zugriff auf seinen Account. Es werden Informationen vom Smart Contract abgerufen und ihm mögliche Funktionalitäten angezeigt.
- **Account-Adresse anzeigen:** Der Student oder Mitarbeiter ruft seine Account-Adresse als QR-Code und Text ab um sie mit anderen Personen teilen zu können. Die Informationen stammen direkt vom Smart Contract, welcher mit der Wallet verknüpft ist.
- **Zugriffsschlüssel anlegen:** Der Student oder Mitarbeiter kann aus seinem Account heraus beliebig viele neue Geräteschlüssel mit unterschiedlichen Rechten oder Backup-Keys generieren. Es werden neue Schlüsselpaare erzeugt und dem Smart Contract hinzugefügt.
- **Zugriffsschlüssel ansehen:** Der Student oder Mitarbeiter kann aus seinem Account heraus alle verknüpften Geräteschlüssel mit unterschiedlichen Rechten oder Backup-Keys ansehen. Aus der Blockchain werden alle Zugriffsschlüssel vom Smart Contract des Benutzers abgerufen und angezeigt.
- **Zugriffsschlüssel speichern:** Der Student oder Mitarbeiter kann aus seinem Account heraus alle verknüpften Geräteschlüssel mit unterschiedlichen Rechten oder Backup-Keys als Datei speichern oder ausdrucken. Aus der Blockchain werden alle Zugriffsschlüssel vom Smart Contract des Benutzers abgerufen und in eine Datei überführt.
- **Zugriffsschlüssel löschen:** Der Student oder Mitarbeiter kann aus seinem Account heraus alle Geräteschlüssel oder Backup-Keys entfernen, zu denen das aktuelle Gerät genügend Rechte besitzt. Es wird eine Transaktion gesendet, welche die Verknüpfung vom Schlüsselpaar zum Smart Contract auf der Blockchain aufhebt.
- **Account wiederherstellen:** Der Student oder Mitarbeiter nutzt einen seiner Backup-Keys, welchen er separat und sicher gespeichert hat, um seinen Account von einem kaputten Gerät wiederherzustellen. Es wird über den Backup-Key eine Transaktion gesendet, welche den zugehörigen Smart Contract Account für das aufrufende Gerät freischaltet.
- **Account verknüpfen:** Der Student oder Mitarbeiter nutzt einen seiner Geräteschlüssel, welchen er separat und sicher gespeichert hat, um seinen Account mit einem weiteren Gerät zu verknüpfen. Es wird über einen Geräteschlüssel eine Transaktion gesendet, welche den zugehörigen Smart Contract Account für das aufrufende Gerät freischaltet.

- **Identitätsdaten hinzufügen:** Der Student oder Mitarbeiter fügt Inhalte zu seinem Account hinzu, indem er seine eingegebenen Daten von verifizierten Institutionen wie der Hochschule, Universität oder Unternehmen bestätigen lässt. Die Institutionen können, je nach Art des Datensatzes, eingegangene Anfragen manuell oder automatisiert prüfen um zu verifizieren. Identitätsdaten können ebenso mit einem Ablaufdatum oder einer Datenhoheit ausgestellt werden. Bei der Ausführung wird entweder ein eigener Datensatz zum Smart Contract hinzugefügt oder eine Transaktion als Anfrage zum Hinzufügen eines Tokens an einen Aussteller gesendet.
- **Identitätsdaten bearbeiten:** Der Student oder Mitarbeiter bearbeitet Teile von eigenen Identitätsdaten. Dies tritt eine neue Transaktion zum Datensatz los, die festgelegte Parameter ändert.
- **Identitätsdaten anzeigen:** Der Student oder Mitarbeiter erhält eine Übersicht mit allen Daten die mit seinem Account verbunden sind. Diese Daten werden vom Smart Contract auf der Blockchain abgerufen.
- **Daten-Maske anlegen:** Der Aussteller von digitalen Identitätsdaten gibt über eine Schnittstelle dem Educhain-Netzwerk eine Maske vor. Sie beinhaltet, welche Informationen von Nutzern für das Aushändigen eines Zertifikates benötigt werden. Der Aussteller verknüpft seine Maske mit dem Netzwerk, welche dann von anderen für Anfragen genutzt werden können.
- **Daten-Maske anzeigen:** Der Aussteller von digitalen Identitätsdaten kann angelegte Daten-Masken in seinem Account ansehen. Die Daten aus den verknüpften Smart Contracts werden von der Blockchain abgerufen.
- **Daten-Maske löschen:** Der Aussteller von digitalen Identitätsdaten kann angelegte Daten-Masken in seinem Account löschen. Es wird eine Transaktion gesendet, welche die Verknüpfung von der Maske zum Account aufhebt und die Daten daraus löscht.
- **Identitätsdaten ausstellen:** Über eine Schnittstelle verifizieren die Hochschulen oder andere Institutionen Anfragen von Accounts die ihnen Daten zur Überprüfung zusenden. Ein neuer Identitätsdatensatz wird mit dem Ausstellen erstmalig in einen Smart Contract gewandelt und dem Benutzer Account zugewiesen.
- **Zugriff auf Identitätsdaten gewähren:** Der Student oder Mitarbeiter erteilt Institutionen das Recht, bestimmte Identitätsdaten einsehen und verwenden zu können.
- **Zugriff auf Identitätsdaten entziehen:** Der Student oder Mitarbeiter entzieht Institutionen das Recht, bestimmte Identitätsdaten einsehen und verwenden zu können.
- **Identitätsdaten erhalten:** Über eine Schnittstelle fragen die Hochschulen oder andere Institutionen identitätsbezogene Daten von Accounts ab. Der Nutzer erhält eine Anfrage und muss bestätigen, dass er seine Daten weitergeben möchte. Es wird eine Transaktion mit Aufruf zur Datenfreigabe gesendet.
- **Identitätsdaten selbst entfernen:** Der Student oder Mitarbeiter entfernt Daten aus seinem Account. Der Datensatz wird vom Smart Contract gelöst und gelöscht, sodass dieser dem Nutzer nicht mehr zuordenbar ist.

- **Identitätsdaten streichen:** Ein verifizierter Aussteller, welcher die Hoheit über einen Datensatz besitzt, streicht diesen von einem Nutzer-Account. Der Datensatz wird vom Smart Contract gelöst, sodass dieser dem Nutzer nicht mehr zuordenbar ist.
- **digitalen Ausweis erstellen:** Der Student oder Mitarbeiter bestätigt, dass er bestimmte hinzugefügten Identitätsdaten als Ausweis benutzen möchte. Es wird eine Transaktion gesendet, welche den Hashwert der Daten eines Identitätsdatensatzes öffentlich nachprüfbar macht.
- **digitale Ausweisen löschen:** Der Student oder Mitarbeiter bestätigt, dass er einen bestimmten Ausweis nicht mehr nutzen möchte. Es wird eine Transaktion gesendet, welche den Hashwert der Daten in einem Identitätsdatensatzes nur noch privat zugänglich macht.
- **digitales Ausweisen:** Der Student oder Mitarbeiter benutzt Ausweise vom Mobilgerät oder Browser aus um Teile seiner Identität preiszugeben. Die Daten werden vom Account auf der Blockchain abgerufen und können von jedem Netzwerkknoten mittels Kryptographie auf Richtigkeit geprüft werden.
- **Geld einzahlen:** Der Student oder Mitarbeiter zahlt Bargeld von einem Automaten der Hochschule ein und erhält die Gutschrift auf sein Konto. Es wird der gleiche Betrag an Tokens erstellt und seinem Account zugewiesen. Die Hochschule erhält einen Prüf-Token, damit die Reserve der Hochschulstelle digital dokumentiert werden kann.
- **Geld auszahlen:** Der Student oder Mitarbeiter möchte sein tokenisiertes Rest-Geld wiederhaben. Er nimmt dies bei einer Hochschulstelle wahr, welche über ein Nutzerkonto mit entsprechender Reserve an Prüf-Token verfügt. Beide Tokens werden vernichtet.
- **digitales Bezahlen:** Der Student oder Mitarbeiter tätigt mit seinem Educhain Account Zahlungen zu anderen Accounts, deren Adressen via QR-Code eingescannt oder manuell hinzugefügt wurden. Die Blockchain führt die Transaktionen aus, wenn genug Geld vorhanden ist. Korrekte Transaktionen ohne doppelte Ausgaben werden dem Netzwerk hinzugefügt.
- **Zugänge erhalten:** Der Student oder Mitarbeiter öffnet mit seinem Mobilgerät Türen. Die IoT-Devices rufen dazu Informationen von der Blockchain ab, um zu überprüfen ob Zugriff gewährt werden darf.
- **Mieten anzeigen:** Der Student oder Mitarbeiter erhält eine Übersicht mit allen aktuellen Mieten. Diese Daten werden vom Smart Contract auf der Blockchain abgerufen.

- **Spind mieten und benutzen:** Der Student oder Mitarbeiter bezahlt einen Spind für eine variabel ausgewählte Zeit und kann ihn für diesen Zeitraum öffnen. Es wird eine Transaktion mit der Geldeinheit und einem Enddatum an die Smart Contract Instanz der Spind-Adresse gesendet, um danach prüfen zu können, ob der Account über die Berechtigung zum Öffnen verfügt. Nach dem Bezahlen kann der Student oder Mitarbeiter einen Spind für die ausgewählte Zeit öffnen. Der Mikrocontroller prüft mit jedem Vorgang, ob der dazugehörige Account noch über die nötigen Rechte verfügt.
- **Spind-Zeit bearbeiten:** Der Student oder Mitarbeiter verkürzt oder verlängert über das Educhain Interface seine Mietzeit des Spindes und löst damit eine Bezahlung oder Rückerstattung aus. Es wird eine erneute Transaktion an den Spind gesendet, welcher entsprechend mit einer Antwort-Transaktion reagiert.
- **Abgaben anzeigen:** Der Student oder Mitarbeiter erhält eine Übersicht mit allen Abgaben. Diese Daten werden vom Smart Contract auf der Blockchain abgerufen.
- **Abgaben nachweisen:** Der Student oder Mitarbeiter lädt ein Dokument lokal in Educhain und bestätigt eine einzigartige Transaktion auf der Blockchain an die Adresse vom Hashwert der Datei, inklusive Zeitstempel und Ersteller-Adresse. Über den Account wird eine Transaktion veröffentlicht und dem Netzwerk hinzugefügt.
- **Validität prüfen:** Der Student oder Mitarbeiter weist den fälschungssicheren Erhalt eines Dokumentes oder Identitätsdaten nach, indem er sie lokal in Educhain lädt. Educhain prüft den Hashwert und zeigt den dazugehörigen Ersteller sowie das Datum der Einreichung aus der dazugehörigen Transaktion an.

Für die konkreten Abläufe der Benutzer und des Ökosystems sind alle Szenarien der Features im Anhang „Anwendungsvorgehen“ mit einzelnen Schritten aufgelistet.

5.1.2 Interne und administrative Prozesse

- **Lehranstalt ins System einbinden:** Eine Lehranstalt erwirbt eine Softwarelizenz von Educhain, wird zum System hinzugefügt und bindet sich mit Kapital zur Netzwerkbesicherung im Konsensalgorithmus ein. Ein neuer Account wird erstellt und verknüpft.
- **Token-Automat anmelden:** Es wird ein neuer Token Automat in der Lehranstalt aufgestellt, sodass von dort aus Einzahlungen von EDUT und Auszahlungen von Euro vorgenommen werden können.
- **Szenario beim Token-Automat abmelden:** Ein Token-Automat wird nicht länger von der Lehranstalt verwendet und wird vom System entfernt. Die Berechtigung zum Tokenisieren wird entfernt.
- **Szenario beim Türschloss anmelden:** Ein neues Türschloss wird der Hochschule ausgestellt und folglich installiert. Das Modul wird mit einer variablen Zugriffs-Vorlage der Lehranstalt verknüpft.
- **Szenario beim Türschloss abmelden:** Ein Türschloss wird vom System entfernt, da es die Lehranstalt nicht mehr benutzt oder zurückgesetzt werden soll. Der Geräte-Account wird gelöscht.
- **Szenario beim Spind anmelden:** Ein Spind-Fach wird zum System hinzugefügt um von Studenten oder Mitarbeitern benutzt werden zu können. Ein neuer DWK wird dem Account hinzugefügt.
- **Szenario beim Spind abmelden:** Ein Spind wird nicht mehr benötigt und vom System getrennt. Der DWK wird vom Account der Lehranstalt entfernt.

Für die konkreten Abläufe der internen und administrativen Prozesse sind alle Szenarien der Features im Anhang „Anwendungsvorgehen“ mit einzelnen Schritten aufgelistet.

5.2 Aktoren

In diesem Kapitel werden die einzelnen Objekte aufgelistet, die im Ökosystem agieren. Ebenso werden die Funktionalitäten beschrieben. Dabei lassen sich drei Parteien ausformulieren: Personen und Einrichtungen, Hardwareintegrationen und Softwarelösungen.

5.2.1 Personen und Einrichtungen

- **Nutzer:** Ist Student oder Mitarbeiter und der Benutzer der Educhain-Plattform. Er kann sich einen Account erstellen und wiederherstellen, Zugriffsschlüssel managen, Identitätsdaten hinzufügen, bearbeiten und selbst entfernen, sich mit seinen Identitätsdaten digital ausweisen, Geld ein- und damit bezahlen, sowie gegebenenfalls auszahlen. Er kann ebenso Zugänge erhalten, Spinde mieten und deren Mietzeit verkürzen sowie verlängern, Abgaben nachweisen und Validität prüfen.
- **Institution:** Ist eine Einrichtung, die Identitätsdaten von der Blockchain abfragen und deren Validität nachprüfen kann. Sie kann über Mitarbeiter-Accounts verfügen, wenn sie Account-Daten von Nutzern einsehen möchte.
- **Aussteller:** Ist eine verifizierte Institution die Identitätsdaten für Accounts ausstellt und gegebenenfalls streichen kann. Sie besitzt selbst mindestens einen Mitarbeiter-Account, welcher Daten-Masken erstellen kann.
- **Lehranstalt:** Ist eine Hochschule oder Universität, die sich aktiv mit Kapital am Konsensalgorithmus der Blockchain beteiligt und mindestens eine Server-Node inklusiver Wallet betreibt um das Netzwerk bei Ausfällen zu stabilisieren. Sie verfügt über Mitarbeiter-Accounts und ist Aussteller. An ihr werden intelligente Schlösser und Spinde sowie Token-Automaten verwendet.
- **Educhain:** Stellt Softwareprodukt sowie Schlösser und Spinde, liefert Support für Hochschulen und Universitäten, bindet sie in das Ökosystem ein und händigt EDUC für die Netzwerkbesicherung aus. Ebenso stellt der Anbieter Token-Automaten und kann sie den Lehranstalten hinzufügen oder entfernen.

5.2.2 Hardwareintegration

- **Intelligentes Schloss:** Modifiziertes Schloss mit einem Mikrocontroller welcher sich ohne Wallet mit der Blockchain verbindet um Identitätsdaten zu validieren.
- **Intelligenter Spind:** Modifizierter Spind mit einem Mikrocontroller welcher sich mit der Blockchain verbinden kann um Transaktionen zu validieren. Die Spinde verriegeln sich automatisch und besitzt eine eigene Wallet.
- **Token-Automat:** Modifizierter Einzahlungsautomat mit Wallet ohne graphisches Benutzerinterface, welcher von der Lehranstalt betrieben wird, das Einzahlen von Euro möglich macht und den identischen Betrag an EDU-Token an den Nutzer, sowie den identischen Betrag an Prüf-Token an das betreibende Hochschulkonto ausstellt.
- **Netzwerk:** Eine Peer-to-Peer Blockchain System aus Server-Nodes welche als Fundament für Educhain dienen und alle Transaktionen und Vorgänge der Nutzer-Accounts beinhalten. Identitätsdatensätze werden verschlüsselt gespeichert.

5.2.3 Softwarelösungen

- **Weboberfläche**: Eine Einbettung von Educhain Anwendung im Browser, in welcher man seine eigene Wallet mit der Blockchain und dem Educhain-Netzwerk verknüpfen kann, um bis auf die IoT-Integration alle Funktionalitäten nutzen zu können.
- **Mobile App:** Ein vollwertiges Educhain Programm mit integrierter Wallet, welches auf allen üblichen Betriebssystemen der Mobiltelefone läuft und jegliche Funktionalität nutzbar macht.
- **IoT-Clients:** Software, welche autonom auf Microchips oder Mikrocontrollern läuft um die Hardwareintegration von Geräten mit Educhain zu ermöglichen. Dabei gibt es zwei Typen:
    - **IoT-Client ohne Wallet:** Kann Daten von der Blockchain validieren
    - **IoT-Client mit Wallet:** Kann Daten validieren und Transaktionen senden

6 Analysemodell

Das Analysemodell beschreibt neben einem Use-Case-Modell die fachliche Logik des zu erstellenden Systems. Dies ist losgelöst von der technischen Realisierung und dient als Anhaltspunkt für die internen Kommunikationswege.

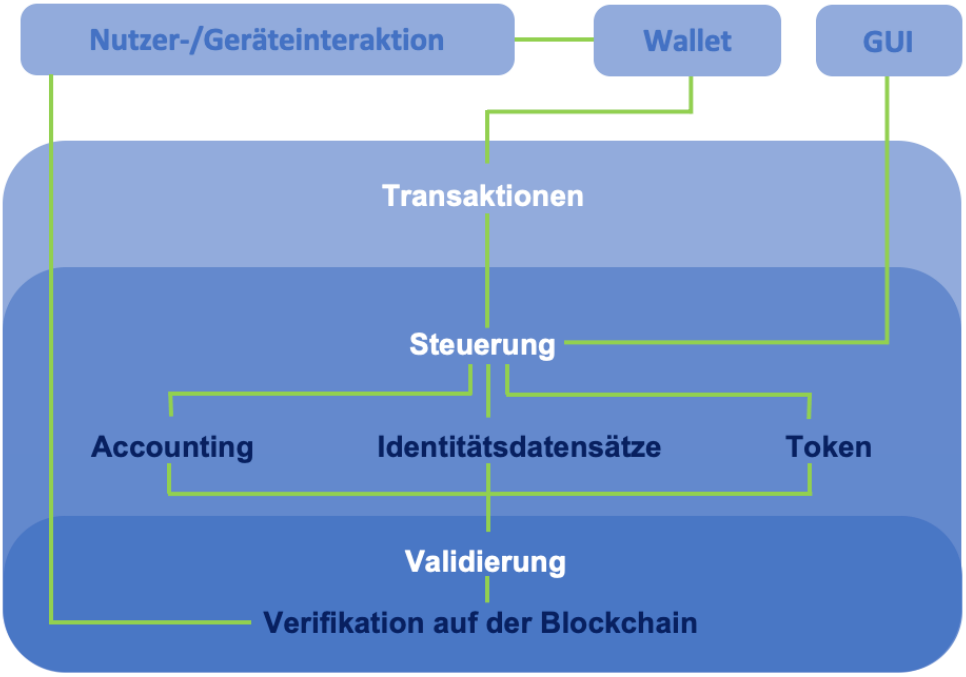

Abbildung 3 Analysemodell

Die Nutzer- und Geräteinteraktion findet über Wallets statt, welche Transaktionen ins Netzwerk senden, um die Steuerung von ihren Accounts sowie deren Identitätsdatensätzen und Token zu realisieren. Diese Steuerungselemente werden grafisch abgerufen. Alle Transaktionen werden auf der Blockchain gespeichert und können wiederherum von Nutzern mittels Interaktion abgerufen und validiert werden.

7 Architekturdokumentation

Die Dokumentation einer Architektur sollte immer einem konkreten Zielpublikum und einem Zweck zugewiesen werden. Educhain unterteilt die zukünftige Architekturdokumentation in zwei Teile: Eine Dokumentation mit Handlungsanweisungen für die Nutze, Aussteller, Lehranstalten und Institutionen sowie eine Dokumentation mit Implementierungsanweisungen als Entwicklerleitfaden, welcher die Schichten und Abläufe im Innern der Software mit konkreten Details beschreiben soll. Die Architektur wird somit von zwei verschiedenen Standpunkten aus betrachtet bzw. bewertet werden und liefert ein umfangreicheres Bild für Umsetzungswünsche.

7.1 Dokumentation für Benutzer

Die Benutzerdokumentation fokussiert sich vor allem auf die Interaktion mit dem Ökosystem. Die beschriebenen produktspezifischen Einflussfaktoren sollten sich hier auf die Unterteilung in verschiedene Akteure und deren individuellen Funktionen, sowie auf das Design der Benutzerschnittstellen beschränken. Bei den technischen Aspekten sollten Benutzer wissen, welche Betriebssysteme und Benutzerschnittstellen verfügbar sind, sowie welche Technologieprojekte geboten werden und wie sie zu verwenden sind. Hier fällt auch der Vorgang beim Aufsetzen einer Server-Node für Lehranstalten und Institutionen hinein. Für die Organisation sollte geklärt werden, wie die Interaktion mit Entwicklern abläuft und wie die Teamkonstellation ist.

Risiken und Lösungen welche zu dokumentieren sind wären zum einen die Sicherung von Accounts und deren Wallet-Schlüssel. Ebenso wie Wartezeiten und Lasten im Netzwerk gehandhabt werden.

Hinweise und zusätzliche Quellen für die Dokumentation wäre unter anderem, wie die Serverstrukturen und das Führen eines Accounts aktuell bei großen Konzernen gelöst werden und welche Vorteile ein nutzer-zentrierter Ansatz bietet. Hier kann die Überleitung zur Datensicherheit und zum Datenschutz gelegt werden. Bei der nutzer-zentrierten Kontoführung bleibt die volle Datenhoheit beim Benutzer. Er reguliert Zugriffe selbstbestimmt.

Für die Architekturschichten ist es für Benutzer unteranderem relevant wie Benutzereingaben im Browser und auf dem Handy entgegengenommen werden und wie mit der Blockchain interagiert wird um Nachweise und Zertifikate abzubilden.

Das Bewerten der Benutzerdokumentation sollte die Speichernutzung beim Betreiben der Software betreffen und abwägen, welche Hardware und (Betriebs-) Software für Nodes, Computer und Mobiltelefone benötigt sind. Der Zeitaufwand bei der Einrichtung sowie Wartezeiten beim Hochfahren und senden von Transaktionen sind ebenfalls Kritikgrundlage. Abschließend sollten Fokuspunkte aus der Sicht des Benutzers aufgestellt werden. Auf diese wird ein besonderes Augenmerk bei der Entwicklung gelegt.

7.2 Dokumentation für Entwickler

Die Entwicklerdokumentation fokussiert sich vor allem auf die Erstellung des Ökosystems. Bei den produktspezifischen Einflussfaktoren kommt es vor allem auf die Modularisierung der unterschiedlichen Technologien an und wie trotz vieler Updates eine hierarchische Struktur aufgebaut werden kann, um eine Ordnung zu wahren. Im gleichen Zusammenhang muss geklärt werden welche Funktionalitäten über einzelne Schnittstellen eingebunden werden sollen. Diese Schnittstellen müssen so angelegt werden, dass neue Projekte kosteneffizient und schnell angedockt werden können, sobald sie aus Forschungsprojekten herausgegeben werden. Die Aufteilung von Benutzerschnittstellen auf verschiedene Akteure sollte hierbei von einem technischen Standpunkt beschrieben werden. Mit der Aufteilung der Benutzerschnittstellen, muss auch die Integration von einheitlichen Designelementen beschrieben werden. Auf der gegenüberliegenden Seite, dem Backend, müssen Themen wie die Lastenverteilung, Datenraten und die Einrichtung von Server-Nodes geklärt werden. Es ist außerdem zu beschreiben, welche Zuverlässigkeit erreicht werden muss, welche Zeitverzögerungen annehmbar sind und wie der Datenschutz und die Datensicherheit bei der Kommunikation zur Blockchain hin aufgestellt ist. Zu allen Vorgängen müssen Testszenarien mit unterschiedlichen Frameworks beschrieben werden. Bei den technischen Einflussfaktoren sind die Hardwareimplementierungen, Prozessorarchitekturen sowie die Netzwerk-Architektur in Bezug auf deren Server und Speicher zu erläutern. Welche Softwareschnittstellen implementierbar sind und wie auch mit kleineren IoT-Geräten an die Blockchain angedockt werden kann, sind zwei wichtige Fragen deren Lösung aufgezeigt werden muss. Themen organisatorischer Natur sind unteranderem, welche Werkzeuge und Methoden bei der Programmierung aber auch beim Managen des Entwicklerteams eingesetzt werden sollen.

Zu dokumentierende Risiken und Lösungen sind, wie das Fehlerhandling gelöst wird, ohne dass eine Beeinträchtigung des Netzwerkes auftritt, aber auch wie eine Umstellung des Netzwerk-Konsensus oder von Hardwarekomponenten abläuft.

Als Hinweise und zusätzliche Quellen für die Dokumentation sind die verwendeten Protokolle mit ihren Technologie-Stacks anzugeben. Ebenso genauere Dokumente für die Prozessoren der IoT-Geräte und den mobilen Betriebssystemen für das Verwenden derer Oberflächen für Speichervorgänge.

Zu beschreibende Architekturschichten sind jegliche Patterns die angewandt werden sollen sowie Architekturstile und -muster auf welche zurückgegriffen wird. Wichtig sind hierbei auch Strukturen beim Deployment der Features.

Bei der Bewertung dieser Dokumentation ist der Entwicklungsaufwand und der Zeitplan im Hinblick auf das Entwicklungsbudget abzustimmen. Ebenso muss bewertet werden, in wie weit die Produktlinienansätze eingehalten werden und ob benutzte Standards, Dateiformate und Frameworks mit den Plattformen der Benutzerschnittstellen kompatibel sind. Ebenso muss geprüft werden, welche Codeabdeckung mit den unterschiedlichen Testmethoden erreicht werden kann. Schlussendlich sollten auch hier Fokuspunkte erörtert werden, auf welche bei der technischen Umsetzung besonderes Augenmerk gelegt werden sollte.

8 Architekturerstellung

Nachdem im letzten Kapitel eine kurze Zusammenfassung über die Architekturdokumentation gegeben wurde, wird nun spezifisch auf die Teilpunkte eingegangen um Strategien zur Bildung einer Softwarearchitektur herausarbeiten zu können. An diesen orientieren sich im späteren Verlauf die Entwurfsentscheidungen des Projektleiters von Educhain, welcher für die Architektur verantwortlich sein wird.

8.1 Spezifikation der Einflussfaktoren

Die Spezifikation der Einflussfaktoren wird in drei Gruppen unterteilt. Es werden separat produktspezifische, technische und Organisatorische Kenngrößen aufgestellt, die für die Architekturerstellung von Relevanz sind.

8.1.1 Produktspezifische Kenngrößen

P1: Funktionale Anforderungen		
Faktoren	*Flexibilität und weitere Informationen*	*Einfluss*
P1.1: Modularisierung		
Die unterschiedlichen Softwaretechnologien sollen jeweils in einem Modul kapselbar sein.	Es ist wahrscheinlich, dass Funktionalitäten mit besseren Technologien oder neueren Versionen umgesetzt werden. Dies soll keinen Einfluss auf das Ökosystem haben.	DE
Die Software soll zukünftig mit neuen Modulen weitere Funktionalitäten erhalten.	Es ist wahrscheinlich, dass das System um Funktionalitäten erweitert wird und neue Technologien unabhängig von bereits Bestehenden, angedockt werden.	DE
P1.2: Aufteilung		
Die Software des Ökosystems soll sich über mehrere Akteure erstrecken und aufteilen.	Zusätzlich zu Desktop- und Mobilanwendungen, ist es notwendig, dass sich das Softwareprodukt an Mikrocontroller unterschiedlicher Größe andocken lässt. Die Softwareentwicklung der Subsysteme soll separat von der Benutzerplattform erfolgen.	DE

Tabelle 1 Produktspezifische Kenngröße: Funktionale Anforderungen

P2: Benutzerschnittstellen		
Faktoren	*Flexibilität und weitere Informationen*	*Einfluss*
P2.1: Oberfläche		
Das Netzwerk als Backend soll mit einer GUI im Browser aufrufbar sein.	Da ein öffentliches Blockchain Netzwerk als Fundament dienen soll, ist es notwendig, dass alle Benutzer grafisch darauf dargestellte Informationen abrufen können.	DE
Das Benutzer-Interface im Browser soll mit einem GUI ausgestattet sein.	Um dem Nutzer ein einheitliches Gefühl zu vermitteln, ist es nötig ein der mobilen Anwendung ähnliches Benutzerinterface für die Verwendung am Computer zu bieten.	DE
Die Mobile Anwendung soll mit einem grafischen GUI ausgestattet sein.	Für die Benutzung auf dem Handy steht die Bedienfreundlichkeit an erster Stelle. Dies erfordert ein natives Handling über ein modernes Benutzerinterface.	DE
P2.2: Änderbarkeit		
Designänderungen am GUI sollen keine Auswirkungen auf Funktionsabläufe haben.	Da sich durch einbinden neuer Technologien die Abläufe und Szenarien bei Benutzerinteraktionen im Hintergrund regelmäßig ändern, sollen Designänderungen an GUI-Elementen getrennt erfolgen.	DE

Tabelle 2 Produktspezifische Kenngröße: Benutzerschnittstellen

P3: Leistung		
Faktoren	*Flexibilität und weitere Informationen*	*Einfluss*
P3.1: Lastenverteilung		
Abfragen und Transaktionen sollen von der nächst näheren Server-Node bearbeitet werden.	Das dezentrale und selbstregulierende Blockchain-Netzwerk soll die Last der Anfragen und Transaktionen auf die nächst nähere Server-Node verteilen welche noch nicht ausgelastet ist.	DE
Verifizierte Transaktionen sollen automatisch an nächste Knoten weitergereicht werden.	Findet eine Server-Node einen Block und verifiziert damit eine Vielzahl an Transaktionen, so wird dieser im Netzwerk verteilt bis alle Knoten neue Transaktionen darauf aufbauen können.	DE
Sharding-Kompatibilität sollte gegeben sein.	Ab einer dreistelligen Server-Anzahl im Netzwerk sollte es möglich sein, Subnetzwerke aufzubauen, um die Skalierung im Netzwerk voran zu treiben. Die Subnetzwerke synchronisieren dann die Mehrzahl an Transaktionen unabhängig voneinander.	DE

Tabelle 3 Produktspezifische Kenngröße: Leistung I

P3.2: Datenrate		
Es sollen zukünftig 30 bis 300 Transaktionen pro Sekunde erreicht werden können.	Das Netzwerk sollte in der Lage sein 30 Transaktionen pro Sekunde auszuführen und mittels Sharding ab einer zweistelligen Lehranstalt-Teilnahme aufgrund der Lastenzunahme auf bis zu 300 Transaktionen pro Sekunde aufrüstbar sein.	DE
Die Aufnahme einer Transaktion sollte nicht länger als 15 Sekunden dauern.	Bei der Benutzung von Spinden oder dem Tätigen von Zahlungen soll der Uni-Alltag nicht verlangsamt werden, weshalb eine schnelle Validierung im Netzwerk essenziell ist.	DE
P3.3: Inbetriebnahme		
Das Hochfahren der App und das Anzeigen der Account-Informationen sollte innerhalb von 10 Sekunden passieren.	Die mobile Anwendung muss schnell betriebsbereit sein um Ausweise verwenden zu können. Die Account-Informationen sollten beim Starten der App zwischengespeichert werden um das nächste Hochfahren zu beschleunigen. Detaillierte Informationen können im Hintergrund abgefragt werden.	DE

Tabelle 4 Produktspezifische Kenngröße: Leistung II

P4: Zuverlässigkeit		
Faktoren	*Flexibilität und weitere Informationen*	*Einfluss*
P4.1: MTTF		
Es sollte nie zu Ausfällen des Netzwerkes kommen, höchstens einer verringerten Rechenkapazität.	Durch den dezentralen Ansatz gibt es keinen Single Point of Failure. Falls sich die Rechenkapazität im Netzwerk verringert, ist mit Verzögerungen, nicht jedoch mit Ausfällen zu rechnen.	DE
P4.2: Fehlertoleranz		
Eine Ausfallquote von 30% des Netzwerkes sollte toleriert werden.	Sollten einige Standorte von Stromausfällen betroffen sein, so muss ermöglicht werden, dass das Netzwerk weiterhin ohne Einschränkungen der Funktionalität nutzbar ist.	DE

Tabelle 5 Produktspezifische Kenngröße: Zuverlässigkeit

P5: Sicherheit		
Faktoren	*Flexibilität und weitere Informationen*	*Einfluss*
P5.1: Datensicherheit		
Alle persönlichen Daten im Netzwerk sollen prinzipiell verschlüsselt werden.	Daten welche ins Netzwerk gelangen sollten nur von berechtigten Personen gelesen werden können.	DE
Es soll Blockchain Architektur zum Einsatz kommen.	In einem dezentralen Peer-to-Peer Netzwerk sollen Transaktionen zu Blöcken zusammengefasst werden, welche in einer Hash-Kette aufeinander aufbauen um diese abzusichern.	DE
Daten sollen von Knoten bestätigt und in ins Netzwerk aufgenommen werden, bevor sie validierbar sind.	Bevor Daten die ins Netzwerk gelangen validierbar sind, soll geprüft werden ob diese rechtmäßig getätigt wurden. Erst nach der Aufnahme in einen Block ist eine Information valide und nach einem Prüfzeitraum von mehreren Blöcken abgesichert.	DE
Das doppelte Ausgeben von Guthaben soll nicht möglich sein.	Mithilfe eines Konsens-Algorithmus der Blockchain-Technologie soll ausgeschlossen werden, dass ein Benutzer vorhandenes Guthaben mehrmals ausgibt, bevor eine fälschliche Transaktion ins Netzwerk gelangt.	DE
Alle Schreibvorgänge im Netzwerk sollten eine Signatur des Nutzerkontos benötigen.	Alle Bezahlungen und Identitätsdaten sind zwangsweise an Accounts bzw. Benutzer gebunden, deshalb sind Signaturen zwangsweise von Nöten um Berechtigungen validieren zu können ohne alle Informationen veröffentlichen zu müssen.	DE

Tabelle 6 Produktspezifische Kenngröße: Sicherheit I

P5.2: Datenschutz		
Wallet-Daten verbleiben vollkommen verschlüsselt und in einer sicheren lokalen Datenbankform gespeichert.	Falls ein Benutzer sein Mobiltelefon verliert, darf es Angreifern nicht gelingen, Klardaten ohne die Berechtigung der Person zu entnehmen.	DE
Es soll eine Anonymität in Bezug auf die Benutzer gewährleistet werden.	Nutzer werden anonym dargestellt um ihre Identitäten vor Institutionen zu bewahren, die Informationen aus dem öffentlich Zugänglichen Netzwerk entnehmen möchten.	DE
Benutzer sollen Zugriffe auf Daten bestätigen, bevor andere diese Informationen sehen können.	Identitätsdaten sind Eigentum einer bestimmten Person, welche die Hoheit darüber hat. Sie selbst entscheidet, mit welche Institutionen auf ihre Account-Daten zugreifen.	DE
Es wird das Prinzip der nutzer-zentrierten Accounts umgesetzt.	Zugriffe zu Accounts und deren Daten werden nicht auf Servern von Institutionen, sondern vom Benutzer selbst gehalten und vollkommen dezentral und verschlüsselt im Netzwerk gespeichert.	DE

Tabelle 7 Produktspezifische Kenngröße: Sicherheit II

P5.3: Digitale Güter		
Es sollten keine Kopien, sondern echte einzigartige Identitätswerte zwischen Accounts transferiert werden.	Alle personenbezogenen Zertifikate, Ausweise und Berechtigungen sind einzigartige Güter, die nur der Person selbst gehören. Dieser Ansatz aus der realen Welt soll auch ins digitale getragen werden.	DE
P5.4: Technischer Defekt		
Bei Fehlern innerhalb der Software, sollte das Netzwerk keinesfalls betroffen sein.	Das Netzwerk erhält neue Informationen über die Softwareanwendungen, sollte jedoch autark von ihm weiterfunktionieren. Die Anwendungsebene sollte die Protokollebene nicht beeinflussen.	DE

Tabelle 8 Produktspezifische Kenngröße: Sicherheit III

P6: Testbarkeit		
Faktoren	*Flexibilität und weitere Informationen*	*Einfluss*
P6.1: Testfreundlichkeit		
Für Tests sollte ein eigenes Netzwerk aufgesetzt werden.	Auf einem Testnetzwerk ist es den Entwicklern möglich, neue Technologien vorzeitig in gleichen Bedingungen wie dem Hauptnetzwerk zu testen.	DE
Es sollte möglich sein, die mobilen Anwendungen auch während der Laufzeit am Computer zu debuggen.	Um eine nahtlose Entwicklung der mobilen Anwendung zu gewährleisten, sollten solche Anwendungen auch über einen Emulator am Computer getestet werden können.	DE
P6.2: Testunabhängigkeit		
Alle Softwaremodule und Subsysteme im Ökosystem sind unabhängig voneinander zu testen.	Da parallel an unterschiedlichen Anwendungen und an der Implementierung von unterschiedlichen Modulen gearbeitet wird, sollte es möglich sein, diese individuell testen zu können.	DE

Tabelle 9 Produktspezifische Kenngröße: Testbarkeit

P7: Agilität		
Faktoren	*Flexibilität und weitere Informationen*	*Einfluss*
P7.1: Kosteneffizienz		
Änderungen an der Funktionalität sollten kosteneffizient durchführbar sein.	Da das Netzwerk ständig mit neuen Funktionalitäten Dritter erweitert wird, ist es notwendig die Kosten für eine Implementierung minimal zu halten.	DE FE
Änderungen am Design sollten kosteneffizient durchführbar sein.	Durch die Symbiose zwischen Browser und mobiler Anwendung sollte es möglich sein den „Look and Feel“-Eindruck durch kosteneffiziente Änderungen zu vereinheitlichen.	DE FE
P7.2: Schnelligkeit		
Erweiterungen und Änderungen sollten schnell ausführbar sein.	Da das System mit wichtigen und vertrauensvollen Informationen der Benutzer umgeht, müssen Fehler schnellstmöglich behoben werden. Erweiterungen sollten ohne Mühen vom Testnetzwerk in die Anwendungen übertragen werden können.	DE

Tabelle 10 Produktspezifische Kenngröße: Agilität

P8: Portierbarkeit		
Faktoren	*Flexibilität und weitere Informationen*	*Einfluss*
P8.1 Plattformumstellung		
Eine Umstellung des Konsensalgorithmus sollte mit Übernahme der Anwendung möglich sein.	Das System von Server-Nodes soll variable auf Updates reagieren, sodass beispielsweise eine bessere Skalierung implementiert werden kann.	DE
P8.1 Hardwareumstellung		
IoT-Clients sollten beim Herstellerwechsel von Türen, Spinden oder Automaten portiert werden können.	Da davon auszugehen ist, dass es bei den eigenen Hardwaremodifikationen zu Lieferengpässen der Hersteller kommen kann, soll die Software auf einer Vielzahl von Mikrocontrollern benutzbar sein.	DE

Tabelle 11 Produktspezifische Kenngröße: Portierbarkeit

8.1.2 Technische Kenngrößen

T1: Hardware		
Faktoren	*Flexibilität und weitere Informationen*	*Einfluss*
T1.1: Prozessoren		
Die Kompatibilität zu ARM Prozessoren für mobile Geräte sollte gegeben sein.	Die breitgefächerte Kompatibilität zu Prozessoren von Mobilgeräten sollte eine größtmögliche Abdeckung im Uni-Alltag hervorrufen. Studenten führen ihr Handy fast jederzeit mit.	DE
Schlösser sollten mit Mikrocontrollern der RF52 Serie von Nordic Semi-conductors kompatibel sein.	Die Mikrocontroller welche in digitale Schlösser passen, dürfen eine Größe von einem Eurostück kaum überschreiten. Es müssen 1MB Speicher und 256KB RAM genügen. [4]	DE
Automaten und Spinde sollten mit dem Micro Controller ESP32 kompatibel sein.	Automaten müssen Mindestanforderungen zum Betreiben einer Wallet und der Funktionalität des Signierens bieten um Transaktionen auf der Blockchain zu veröffentlichen. Der Chip sollte 520KB RAM und 16MB Speicher besitzen. [3]	DE
T1.2: Netzwerk		
Es sollte ein Peer-to-Peer Blockchain Netzwerk mit Smart Contract Support verwendet werden.	Um Identitätsdaten und Accounts erstellen zu können müssen diese in Smart Contracts mit einer großer Funktionalitätsbreite ausgeführt werden können.	DE

Tabelle 12 Technische Kenngröße: Hardware I

T1.3: Speicher		
Das Netzwerk sollte einen Speicher von 100GB pro Jahr nicht überschreiten.	Je höher die Anforderungen für den Speicher, umso weniger dezentral wird das Netzwerk. Es sollte möglichst einfach sein, Knoten auch als Privatperson zu betreiben.	DE
Die ausführbare App auf dem Mobiltelefon sollte 300MB nicht überschreiten.	Das Laden der App aus dem Store sollte auch bei schlechterer Internetverbindung möglich sein. Die Anwendung sollte den Studenten auch bei geringem Speicher nicht einschränken.	DE

Tabelle 13 Technische Kenngröße: Hardware II

T2: Softwaretechnologie		
Faktoren	*Flexibilität und weitere Informationen*	*Einfluss*
T2.1: Betriebssysteme		
Die mobile Anwendung sollte auf Android und iOS lauffähig sein.	Um die Anwendung möglichst breitflächig zu verteilen sollte sie mit allen gängigen Betriebssystemen kompatibel sein.	DE
T2.2: Benutzerschnittstelle		
Die GUI sollte einzige Benutzerschnittstelle darstellen.	Alle Zugänge vom Benutzer zur Software sollten grafischer Natur sein, um einen kinderleichten Einstieg in die Anwendungen zu ermöglichen sowie eine transparente Ansicht das Backend zu gewährleisten.	DE

Tabelle 14 Technische Kenngröße: Softwaretechnologie I

T2.2: Softwareschnittstelle		
Kompatibilität zum Forschungsprojekt ECHT! sollte gegeben sein.	Es wird das Erstellen von Datennachweisen ausgelagert und vom BCCM in einem Forschungsprojekt weitergeführt. Aktuelle Versionen werden in Educhain eingearbeitet. [6]	DE
Kompatibilität zum Forschungsprojekt ID-IDEL sollte gegeben sein.	Es wird das ausgeben von Studentenausweisen ausgelagert und vom BCCM in einem Forschungsprojekt weiterentwickelt. Aktuelle Versionen werden in Educhain eingearbeitet.	DE
T2.3: Programmiersprache		
JavaScript sollte für die mobile Anwendung und die Browserintegration verwendet werden.	JavaScript sollte für die Programmierung mit React sowie React Native verwendet werden, um mit einer Programmiersprache alle Benutzeroberflächen zu kreieren. Dies ist vor allem im Entwickler-Team von Vorteil, weil keine weitere Sprache benötigt und Änderungen mit wenig Aufwand übernommen werden können. Dieser Ansatz ist sehr zeitsparend. [13, 14]	DE
Mittels Solidity sollten alle Anwendungen auf der Blockchain programmiert werden.	Durch Solidity können Smart Contracts auf den Anwendungsebenen von Ethereum-basierenden Blockchains programmiert werden. Solidity bietet die bisher größtmöglichen Funktionalitäten und ist Kompatibel mit allen Account- und Token-Standards	DE
Alle IoT-Geräte sollten die Programmiersprache C benutzen	Die Programmiersprache C benötigt wenig Speicher und Rechenleistung und ist für die Verwendung mit Kleinstgeräten optimal.	DE

Tabelle 15 Technische Kenngröße: Softwaretechnologie II

T2.4: Frameworks		
Es sollte ein eigenes GUI Framework erstellt werden.	Mittels React lassen sich eigene Frameworks für das Gestalten von Benutzeroberflächen entwerfen, welche dann auf allen Geräten benutzt werden können. So ist es möglich, Grafisches und Funktionelles separat zu entwickeln. [13, 14]	DE
T2.5: Libraries		
Die Bibliothek React sollte für die Erstellung der GUI verwendet werden.	React ist eine Library die eine Struktur für die Programmierung eines Benutzerinterfaces für Browser in JavaScript vorgibt. [13]	DE
React Native sollte für die GUI der mobilen Endgeräte verwendet werden.	React Native ist eine Library die eine Struktur für die Programmierung eines Benutzerinterfaces für mobile Geräte in JavaScript vorgibt. [14]	DE
Die Möglichkeit, die web3.js Bibliothek in JavaScript einzubinden sollte genutzt werden.	Web3 ist eine JavaScript Bibliothek zum Verbinden der eigenen Anwendungen mit der Blockchain Technologie. Sie bietet viele vorgefertigte Funktionen für Smart Contract Interaktion und Transaktionen.	DE
IoT-Geräte sollten IN3 benutzen, um sich mit der Blockchain zu verbinden.	IN3 ist der weltweit kleinste Client der IoT-Geräte mit der Blockchain verbinden kann um Informationen zu validieren. [7]	DE
Für das Erstellen einer Wallet auf mobilen Endgeräten sollte Lightwallet benutzt werden.	Lightwallet ist eine JavaScript Bibliothek die minimale Anforderungen an das System stellt und in dApps zum Einsatz kommen kann. Sie lässt sich als Modul vollständig in eigene Systeme integrieren.	DE

Tabelle 16 Technische Kenngröße: Softwaretechnologie III

T2.6 Design Pattern		
Das Composite Pattern sollte für das Kreieren von Identitätsdatensätzen benutzt werden.	Viele einzelne Parameter von persönlichen Informationen und Dokumenten werden zum Erstellen einer großen Ausweis- oder Identitäts- Komponente zusammengetragen.	DE
Es bietet sich an, das Iterator Pattern beim Zugriff von NFC-Geräten auf die Ausweise des Benutzers anzuwenden.	Es werden alle Adressen der Ausweise iteriert und übersendet, um festzustellen ob einer der Ausweise über genügend Rechte verfügt um das IoT-Gerät zu öffnen. Dabei muss die interne Struktur nicht offengelegt werden, was einen Sicherheitsgewinn darstellt.	DE
Das Builder Pattern sollte beim Zusammenfügen eines Identitätsdatensatzes aus Informationen einer Vorlagemaske angewandt werden.	Wenn der Nutzer eine Vorlage des Ausstellers verwendet um einen Identitätsdatensatz anzulegen, muss beim Erstellen ein Smart Contract generiert werden, welcher alle Informationen und die angegebenen Rechte widerspiegelt und Informationen übergibt. Dies soll als separater Builder-Prozess getan werden.	DE
Das Template Pattern sollte genutzt werden um Benutzern Vorlage für die Erstellung von Identitätsdatensätzen zu bieten.	Mit einer Schablonenmethode ist es möglich, den Nutzern Entwürfe zum Einbinden persönlicher Informationen wie Zertifikate, Ausweise oder Dokumenten zu geben. In solche werden dann individuelle Daten hineingegeben.	DE

Tabelle 17 Technische Kenngröße: Softwaretechnologie IV

Das Decorator Pattern sollte benutzt werden um Identitätsdaten mit der Funktionalität von Ausweisen zu erweitern.	Ein Benutzer erhält Identitätsdaten von einem Aussteller welche zu Beginn an nur für beide ersichtlich sind. Richtet er sich diesen Identitätsdatensatz als Ausweis ein, können auch IoT-Geräte oder andere Institutionen ohne eine erneute Anfrage Informationen von diesem Datensatz validieren.	DE
Das Clone Factory Pattern sollte verwendet werden, um Hashs von Dateien mit zusätzlichen Informationen abzuspeichern.	Es gibt einen vordefinierten Smart Contract für das Abspeichern Dokumenthash, sowie deren angefügte Benutzerinformationen. Soll ein neuer Nachweis mit Nutzerinformationen abgespeichert werden, wird ein neuer Smart Contract Klon erstellt. Dies hat zur Folge, dass sehr viel Rechenleistung eingespart wird, da nicht mehr der komplette Smart Contract neu erstellt werden muss, sondern alle auf die ursprüngliche Instanz verweisen.	DE

Tabelle 18 Technische Kenngröße: Softwaretechnologie V

T3: Architekturtechnologie		
Faktoren	*Flexibilität und weitere Informationen*	*Einfluss*
T3.1: Architekturstile		
Es soll ein Aufbau mit Schichten innerhalb der Blockchain existieren.	Im Innern der Blockchain arbeitet eine Art dezentrale Virtuelle Maschine zum Ausführen von Bytecode, Messaging und Services. Diese Schicht wird von der darüberlegenden Ebene mit Informationen gespeist und umfasst digitale Signaturen, Transaktionen, Hashing sowie die Blockkette. In der nächst höheren Schicht kommuniziert das Peer-to-Peer Netzwerk um sich auf einen Konsens zu allen validen Ereignissen zu einigen. In der finalen Ebene setzten dezentrale Anwendungen über Smart Contracts oder grafische Benutzeroberflächen auf das darunterliegende Fundament auf	DE
Identitätsdaten sollen objektorientiert behandelt werden.	Jeder Wertgegenstand der mittels eines Smart Contracts auf der Blockchain dargestellt wird erhält objektorientierte Funktionen zum Abrufen und Ändern diverser Einstellungen. Dies trägt sich bis zu den Accounts hin, welche als Objekt einer Person gesehen werden und individuelle Funktionen besitzen.	DE
Adressen sollen über Implicit Invocation aufgerufen werden.	Auf der Blockchain verbirgt sich jede Funktionalität hinter einer Adresse. Die Instanz, welche Informationen abrufen möchte weiß nicht, ob sich ein Benutzerkonto, ein Identitäts-datensatz oder nur eine einfache Transaktion dahinter verbirgt. Erst mit dem Aufrufen der besagten Adresse und dem Anwendungsfall vorgegebenen Funktionen wird getestet, ob diese tatsächlich ausgeführt werden kann oder es sich um Falschinformationen handelt.	DE

Tabelle 19 Technische Kenngröße: Architekturtechnologie I

T3.2: Architekturmuster		
Nebenläufigkeit der Prozesse auf dem Betriebssystem sollte unterstützt werden.	Bei der Benutzung kommt es zu Hintergrundaktualisierungen der auf dem Account gespeicherten Informationen, welche während aktiver Benutzereingaben getätigt werden sollten.	DE
Buchungsvorgänge auf der Blockchain sollten eine hohe Persistenz besitzen.	Neue Blöcke der Kette referenzieren immer auf den Vorherigen, was zur Folge hat, dass alle Blöcke dezentral und ausfallsicher im Netzwerk der Server-Nodes verwahrt werden müssen um eine Nachprüfbarkeit zu wahren.	DE
Die Netzwerk-Verteilung sollte als Peer-to-Peer Blockchain implementiert werden.	Alle Accounts und transaktionstätigende Geräte werden über gleichberechtigte Wallets in einem Netzwerk ohne Administration erstellt. Das Rechtesystem wird in der Anwendungsebene aufgesetzt.	DE

Tabelle 20 Technische Kenngröße: Architekturtechnologie II

T3.3: Produktlinienansatz		
Es sollte eine contract-basierende Kontoführung verwendet werden.	Für die Blockchain genügt es, eine Wallet mit privatem und öffentlichem Schlüssel zu besitzen. Dies bietet lediglich die Möglichkeit Transaktionen entgegenzunehmen und birgt die Verlustgefahr falls ein Schlüssel nicht mehr auffindbar ist. Mit dem Benutzen von Smart Contract Accounts wird die Benutzerverwaltung auf der Anwendungsebene geführt. Dabei erhält jedes Gerät des Benutzers eine Wallet und es können komplexe Profile mittels Verlinkungen entstehen.	DE
Es sollte ein nutzer-zentrierter Ansatz bei der Datenspeicherung erwägt werden.	Aktuelle Plattformen haben das Problem, dass Nutzerdaten auf Firmenservern gespeichert werden und der eigentliche Nutzer lediglich die Berechtigung hat, sich in seinen Account einzuloggen. Mit einem nutzer-zentrierten Ansatz besitzt der Account die Daten selbst und entscheidet allein über deren Verwendung.	DE

Tabelle 21 Technische Kenngröße: Architekturtechnologie III

T4: Standards		
Faktoren	*Flexibilität und weitere Informationen*	*Einfluss*
T4.1: Betriebssystem		
Die mobile Anwendung sollte auf Android und iOS lauffähig sein.	Um die Anwendung möglichst breitflächig zu verteilen sollte sie mit allen gängigen Betriebssystemen kompatibel sein.	DE
Mittels moderner Browser wie Chrome, Firefox oder Safari sollte die Anwendung mit einer externen Wallet aufgerufen werden können.	Über die MetaMask Schnittstelle sollte dem Benutzer die Möglichkeit gegeben werden, seinen Account auch von einem Rechner einzusehen und mit Webseiten von Institutionen zu interagieren, welche Informationen von der Blockchain abrufen möchten. [8]	DE
T4.2: IKT		
Mittels Browser sollte HTTPS für die Kommunikation benutzt werden.	Das sicher Hypertext-Übertragungsprotokoll dient der Übermittlung von Informationen im Internet. Es ermöglicht ein abhörsicheres Übertragen von Datenströmen.	DE
Bei der Übertragung von IoT-Geräten sollte NFC zum Einsatz kommen.	Die Übertragung durch Nahfeldkommunikation ist ein Standard welcher aus der RFID-Technik abgeleitet wurde. Mit ihm können Daten in einer Reichweite von bis zu 4 Zentimeter übertragen werden, was sicherstellt, dass das Gerät unmittelbar Zugriff zum IoT-Device möchte.	DE
Für Transaktionen und Konsens im Netzwerk sollten die Ethereum Netzwerkprotokolle benutzt werden.	Die Server-Nodes der Ethereum Blockchain führen untereinander einen Konsens zum Validieren von Blöcken herbei und stellen sicher, dass Transaktionen aus dem Memory Pool in die Blockkette gespeist werden.	DE

Tabelle 22 Technische Kenngröße: Standards I

T4.3: Datenbank		
Die Ethereum Blockchain sollte als Datenbank dienen.	Innerhalb der Ethereum Blockchain, als dezentrale Datenbank, werden Transaktionen und Programme im Netzwerk sicher verwahrt und nachträglich mit der Erstellung neuer Blöcke besichert.	DE
T4.4: Datenformate		
Über das JSON Format sollten Abfragen und ABI Dateien gesichert werden.	Mittels JSON ist es möglich mittels ID-IDEAL Identitäten zu bestätigen. Ebenso wird das Datenformat benötigt um die Smart Contracts auf der Blockchain bedienen zu können, da man sonst durch Implicit Invocation zwischen Accounts nicht wüsste, wie die Schnittstellen anzusprechen sind [5]	DE
Für die Veröffentlichung von Smart Contracts werden Vorlagen in SOL gespeichert.	Die Anwendung muss solche Dateien kreieren können um sie in Bytecode umzuwandeln und als Identitätsdatensatz auf der Blockchain zu veröffentlichen.	DE
Für die Ausgabe und den Druck von Geräte- und Backup-Schlüsseln sollte PDF verwendet werden.	Das geräteunabhängige Datenformat eignet sich ideal für die Darstellung in unterschiedlichen Auflösungen und benötigt zudem sehr wenig Speicher.	DE

Tabelle 23 Technische Kenngröße: Standards II

8.1.3 Organisatorische Kenngrößen

O1: Management		
Faktoren	*Flexibilität und weitere Informationen*	*Einfluss*
O1.1: Bauen oder Kaufen		
Es sollte eine IN3 Lizenz gekauft werden um IoT Geräte zu verknüpfen.	Bei einer kleinen Menge an Geräten ist keine Lizenz notwendig, jedoch sollte sie erworben werden, damit dem schnellen Anstieg bei der Integration nichts im Weg steht. [7]	DE FE
Die Nutzungsrechte für ID-IDEAL sind vorerst kostenlos zu verwenden.	Das ID-IDEAL Projekt des BCCM befindet sich noch in der Forschung und soll möglichst früh Anwendungsfälle finden und genutzt werden. Es entsteht eine Symbiose mit Educhain. [5]	DE FE
Die Nutzungsrechte für ECHT! sind vorerst kostenlos zu verwenden.	Das ECHT! Projekt des BCCM befindet sich noch in der Forschung und soll möglichst früh Anwendungsfälle finden und genutzt werden. Es entsteht eine Symbiose mit Educhain. [6]	DE FE

Tabelle 24 Organisatorische Kenngröße: Management I

O1.2: Zeitplan		
Aufsetzen des Netzwerkes und verwenden von einem digitalen Studentenausweis in den ersten 3 Monaten sollte erreicht werden.	Priorisiert wird das Erstellen eines Blockchain Fundaments auf welchem Nutzer Accounts erstellen können um einen digitalen Ausweis mitzuführen und Dateinachweise erstellen. Dies benötigt das Vorhandensein eines Explorers, einer Weboberfläche und einer mobilen App.	ZE
Ausbau der Benutzer-schnittstellen mit allen konto-basierten Use Cases innerhalb der nächsten fünf Monate.	Um das volle Potential von contract-basierter Kontoführung ausreizen zu können, muss es möglich sein Identitätsdaten auszustellen und hinzuzufügen. Eine Vielzahl an Features zum Managen des Eigenen Accounts wird ebenso implementiert.	ZE
Als erste Erweiterung sollte das Benutzen von Schlössern und Spinden innerhalb von sechs Monaten hinzugefügt werden.	Hierzu müssen IoT-Clients geschrieben werden die sich mit der Blockchain verbinden können und in moderne Schlösser eingefasst werden.	ZE
Hinzufügen von Bezahlung innerhalb weiterer drei Monate sollte möglich sein.	Mit der bereits gesammelten Expertise mit der Hardwareintegration fehlt lediglich die Integration der Automat-Funktionalität in einem Smart Contract und die Einfassung in einen Geldautomaten.	ZE

Tabelle 25 Organisatorische Kenngröße: Management II

O1.3: Geschäftsziele		
Der Kundenkreis von Hochschulen soll stetig erweitert werden.	Das hochschulübergreifende Benutzen steht im Mittelpunkt des Ökosystems und dient als eines der Hauptziele. Je mehr Nutzer an unterschiedlichen Lehranstalten partizipieren umso mehr lohnt sich der Beitritt zum System.	FE
Das Ökosystem soll nach der Entwicklungszeit von zwei Jahren vollständig abgebildet sein und erste Gewinn erwirtschaften.	Initial ist das Projekt von Investments und Forschungen abhängig, es soll sich jedoch zusätzlich zur Software die Integration von eigens modifizierter Hardware möglich gemacht werden. Um diesen Zweig auszubauen müssen vorerst Gewinne mit dem Softwaresystem erzielt werden.	FE
Der Marktanteil für digitale Studentenlösungen soll vergrößert werden.	Die Software soll möglichst schnell neue Funktionalitäten aus der Forschung beinhalten um einen Vorsprung zur Konkurrenz zu bekommen und den Marktanteil zu erhöhen. Nur mit einem hohen Marktanteil kann länderüberschreiten ausgebaut werden.	FE

Tabelle 26 Organisatorische Kenngröße: Management III

O2: Mitarbeiter		
Faktoren	*Flexibilität und weitere Informationen*	*Einfluss*
O2.1: Anwendungskreis		
Es sollten Entwickler mit Blockchain Know-How eingestellt werden.	Die Implementierung vom Netzwerk und das Programmieren von Smart Contracts ist der Kernbestandteil des kompletten Produktes.	DE
Es sollten Entwickler mit Erfahrungen in IoT und Programmierung für Mikro-controller eingestellt werden.	Für die Anbindung an Schlösser, Spinde und Token-Automaten ist es zwangsweise nötig eigene Software auf Mikrocontrollern zu implementieren, welche dann über die mobile App eingerichtet werden können.	DE
Es sollten Entwickler mit Erfahrung in JavaScript und der mobilen Entwicklung eingestellt werden.	Alle Funktionalitäten sollen mittels Browser und einer eigenen mobilen App aufrufbar sein. Um Entwickler und Kommunikation zu sparen, werden beide Lösungen in JavaScript erstellt.	DE

Tabelle 27 Organisatorische Kenngröße: Mitarbeiter I

O2.2: Werkzeuge		
Die Benutzung von React bzw. React Native sollte den Mitarbeitern bekannt sein.	Mit den React-Libraries zur Entwicklung von Benutzeroberflächen im Browser oder nativen Geräten kann mittels einer einzigen Programmiersprache sowohl die mobile Entwicklung als auch die Browserschnittstelle vorangetrieben werden. [13, 14]	DE
Es ist gewünscht, dass Mitarbeiter die Remix IDE im Browser benutzen.	Eine Entwicklungsumgebung zum Programmieren und Testen von Smart Contracts auf der Blockchain, welche sich identisch wie die Educhain Benutzeroberfläche im Browser mit MetaMask verknüpft. [8, 9]	DE
Es sollte Visual Studio Code als hauptsächliche Entwicklungsumgebung für GUI genutzt werden.	Die Entwicklungsumgebung Visual Studio Code ist ein modular aufgebautes Programm mit großflächigen Erweiterungen zum Debuggen, Schreiben von Tests sowie der Build-Erstellung durch ein integriertes Terminal.	DE
O2.2: Methodik		
Es sollte ein agiler Entwicklungsprozess angestrebt werden.	Durch die starke Modularisierung innerhalb des Projektes und der stetigen Auslieferung von Updates ist ein agiles Projektmanagement eine große Stütze um die Übersicht zu bewahren.	DE
Es ist gewünscht SCRUM als Vorgehensmodell einzusetzen.	Für das Produkt und Projektmanagement ist es nötig sich an das agile Arbeiten anzupassen. SCRUM liefert eine perfekte Struktur um Features in festgelegten Sprints umzusetzen.	DE

Tabelle 28 Organisatorische Kenngröße: Mitarbeiter II

O3: Prozess und Entwicklungsumgebung		
Faktoren	*Flexibilität und weitere Informationen*	*Einfluss*
O3.1: Plattform		
Git sollte als Entwicklerplattform zur verteilten Versionsverwaltung dienen.	Durch viele unterschiedliche Entwicklungszweige und verschiedene Stadien, in denen sich bestimmte Features befinden, sollte Git mit unterschiedlichen Repositories und Branches ideal sein um die Entwicklung auf mehrere Mitarbeiter aufzuteilen. [15]	DE
O3.2: Entwicklungsprozess		
Es ist gewünscht SCRUM für die Koordination im Entwicklungsprozess einzusetzen.	SCRUM liefert eine perfekte Struktur um Features in festgelegten Sprints innerhalb der agilen Softwareentwicklung umzusetzen.	DE

Tabelle 29 Organisatorische Kenngröße: Prozess und Entwicklungsumgebung I

O3.3: Entwicklungstools		
Die Benutzung von React bzw. React Native sollte in Betracht gezogen werden.	Mit den React-Libraries zur Entwicklung von Benutzeroberflächen im Browser oder nativen Geräten kann mittels einer einzigen Programmiersprache sowohl die mobile Entwicklung als auch die Browserschnittstelle vorangetrieben werden. [13, 14]	DE
Es ist gewünscht die Remix DIE im Browser zu benutzen.	Eine Entwicklungsumgebung zum Programmieren und Testen von Smart Contracts auf der Blockchain, welche sich identisch wie die Educhain Benutzeroberfläche im Browser mit MetaMask verknüpft. [8, 9]	DE
Es sollte Visual Studio Code als hauptsächliche Entwicklungsumgebung genutzt werden.	Die Entwicklungsumgebung Visual Studio Code ist ein modular aufgebautes Programm mit großflächigen Erweiterungen zum Debuggen, Schreiben von Tests sowie der Build-Erstellung durch ein integriertes Terminal.	DE

Tabelle 30 Organisatorische Kenngröße: Prozess und Entwicklungsumgebung II

O3.4: Testprozess		
Es sind automatisierte Black-Box Tests für Benutzereingaben gewünscht.	Über verschiedenste JavaScript Vorlagen und Bibliotheken sollen vor dem bereitstellen nochmals alle Benutzereingaben geprüft werden.	DE
Manuelle Szenario-basierte Tests von Smart-Contract Funktionalitäten sind gewünscht.	Um die Funktionalität einzelner Methoden in einem Contract zu prüfen wird separat ein Test mit unterschiedlichsten Dummy-Daten eingepflegt	DE
Performance-Tests innerhalb der mobilen App sollten regelmäßig durchgeführt werden.	Beim Erstellen von Identitätsdaten oder PDF Dateien sind Auslastungen und Wartezeiten zu testen. Gleiches gilt beim Ausführen von Smart-Contract-Transaktionen.	DE
O3.5: Testwerkzeuge		
Das JEST Testframework sollte im Frontend zum Einsatz kommen.	JEST bietet viele Funktionalitäten zum Testen von React-Applikationen im Frontend und beinhaltet Code Coverage Tools sowie die Möglichkeit zur Screen-Analyse. [12]	DE
Das MOCHA Testframework sollte im Backend der Apps zum Einsatz kommen.	MOCHA basiert auf NodeJS und bietet nützliche Möglichkeiten um Funktionalitäten im Inneren der App sowie die Übergabe von Parametern an die Blockchain nachzuprüfen. [11]	DE
Das TRUFFLE Framework sollte bei Solidity Code zum Einsatz kommen.	TRUFFLE bietet Test-Möglichkeiten für Smart Contracts auf der Blockchain. Mit diesem Framework lassen sich Funktionen auch außerhalb der Ethereum Blockchain lokal testen. [10]	DE

Tabelle 31 Organisatorische Kenngröße: Prozess und Entwicklungsumgebung III

O4: Entwicklungszeitplan		
Faktoren	*Flexibilität und weitere Informationen*	*Einfluss*
O4.1: Time-to-Market		
Ein erster Launch sollte bereits nach 3 Monaten möglich sein.	In diesen 3 Monaten soll die Ethereum Blockchain Aufgesetzt und mit Accounts bestückt sein, welche dann Daten nachweisen und ihre Hochschulidentität bestätigen lassen können.	DE ZE
Vollwertiges System nach spätestens 2 Jahren Entwicklung sollte angestrebt werden.	Binnen 2 Jahre soll ein vollwertiges Managen von Identitätsdaten ermöglicht und schrittweise die Anbindung an IoT-Geräte erreicht werden.	DE ZE
O4.2: Lieferzeitplan		
Sollten weitere neue Technologie-Module eingebunden werden, sollten dies nicht länger als 2 Monate dauern.	Da das Ökosystem sehr modular und agil Entwickelt werden soll, ist das hinzufügen neuer Technologien einfach gestrickt und muss lediglich separat in die Benutzerabläufe und dem GUI eingearbeitet werden.	DE ZE

Tabelle 32 Organisatorische Kenngröße: Entwicklungszeitplan I

O4.3: Meilensteine		
Aufsetzen des Netzwerkes und verwenden von einem digitalen Studentenausweis.	Es wird ein Blockchain Fundament erstellt auf welchem Nutzer Accounts erstellen können, um einen digitalen Ausweis mitzuführen und Dateinachweise zu erstellen. Über eine Weboberfläche und einer mobilen App ist der Dienst nutzbar.	DE ZE
Ausbau der beiden Apps mit allen konto-basierten Use Cases.	Es wird eine große Bandbreite an Funktionalitäten hinzugefügt, sodass Accounts nun Identitätsdaten hinzufügen und mit ersten Ausstellern kommunizieren können.	DE ZE
Benutzen von Schlössern und Spinden.	IoT-Clients werden erstellt und verbinden sich mit der Blockchain. Sie werden in moderne Schlösser eingefasst und schrittweise in den Markt gegeben.	DE ZE
Hinzufügen von Bezahlung.	Integration der Automat-Funktionalität in einem Smart Contract und die Einfassung in einen Geldautomaten zum Tokenisieren von Euro.	DE ZE

Tabelle 33 Organisatorische Kenngröße: Entwicklungszeitplan II

O5: Entwicklungsbudget		
Faktoren	*Flexibilität und weitere Informationen*	*Einfluss*
O5.1: Anzahl Mitarbeiter		
Zwei Gründer als Projektleiter und Controller, monatlich je 1500€ als Werksstudent.	Vom Start bis zum Vorzeigen eines Prototyps genügt es, dass Educhain von zwei Personen betreut wird.	FE
Das Team soll um zwei Entwickler in den Bereichen Mobile und IoT ergänzt werden, monatlich je 2800€ als Junior Developer.	Mit einer ersten Preview geht es an den Ausbau der Mobilen App und der Interaktion mit IoT Geräten. Dies erfordert sehr viel Erfahrung und kann nicht mehr von nur einem Entwickler getragen werden.	DE FE
O5.2: Kosten für Tools		
Lizenz für IN3 nach anderthalb Jahren Entwicklungszeit.	Mit dem Launch und Einsatz von mehreren Hundert Schlössern und Türen muss eine Lizenz erworben werden. Anfangs ist diese für nicht-geschäftliche Zwecke kostenlos. [7]	FE
O5.3: Kosten für Hardware		
Betreiben von Nodes und Modifikation von Schlössern, Spinden und Automaten, 15.000€.	Die Integration von Hardware muss ausgiebig getestet werden und verlangt das Aufsetzen mehrerer Server, sowie das Testen mit Schlössern und Spinden.	DE FE

Tabelle 34 Organisatorische Kenngröße: Entwicklungsbudget

8.2 Plattformen

In diesem Kapitel werden beschrieben, welche Software und Hardwareintegrationen anhand der Spezifikationen zum Einsatz kommen sollen und welche Eigenschaften sie mit sich bringen.

8.2.1 Software

Ethereum bietet eine Open Source Plattform der Blockchain-Technologie an, welche es ermöglicht, digital und dezentral Bezahlungen zwischen Parteien wahrzunehmen, welche sich grundsätzlich nicht vertrauen müssen. Auf ihr können mittels Smart Contracts ganze Benutzerkonten samt Rechtesystemen aufgebaut, sowie deren digitale Zertifikate verwaltet werden. Ebenso lassen sich eine Vielzahl von Funktionalitäten wie die Technisierung von analogen Gütern, digitale Währungen oder das Ausführen von Wahlvorgängen realisieren.

Die Blockchain lässt sich über die Ethereum *Lightwallet* mit minimalen Anforderungen in JavaScript-Anwendungen einfassen und bietet eine Vielzahl an Möglichkeiten zur Generierung neuer Adressen, dem Senden von Transaktionen oder dem Interagieren von Smart Contracts.

Im Browser soll *MetaMask* als Wallet-Anbindung genutzt werden. Die Software ist eine Browser-Erweiterung und muss separat installiert werden, bietet aber gleiche Funktionalitäten wie *Lightwallet* und sogar ein GUI für Benutzer. [8]

Für die Programmierung besagter GUI kommt *React* bzw. *React Native* zum Einsatz. Es ist ein Framework für Benutzerschnittstellen und bietet Möglichkeiten eigene Designs unabhängig von Programmabläufen zu implementieren. [13, 14]

ID-IDEAL ist ein blockchain-unabhängiges Forschungsprojekt des BCCM, welches sich darauf spezialisiert, digitale Identitäten auszuhändigen und zu validieren. Die Identitäten sind wie reguläre Brieftaschen an einzelne Personen geknüpft und können Ausweise, Zertifikate, Zeugnisse oder Zugänge beinhalten. Mit Einwilligung des Benutzers können diese an verschiedenste Akteure herausgegeben und von solchen geprüft werden. Eine einfache Verifizierung des Studentenausweises ist über HTTPS mittels JSON möglich. [5]

Auch *ECHT!* ist, ein Forschungsprojekt des BCCM, welches sich mit damit befasst, wie Dokumente digital, fälschungssicher und nachprüfbar auf der Blockchain-Technologie verwaltet werden können. [6]

IN3 ist eine Softwarelösung der Firma Blockchains LLC, ehemals slock.it aus Mittweida, welche sich darauf spezialisiert hat, es IoT-Geräten zu ermöglichen, mit der Blockchain zu kommunizieren. Die Software kann mit einer Größe von 300KB selbst auf den kleinsten Mikrocontrollern zum Einsatz kommen. [7]

8.2.2 Hardwarekomponenten

Zum Einsatz kommen zwei Typen von Mikrocontrollern. Mikrocontroller welche in digitale Schlösser passen, dürfen eine Größe von einem Eurostück kaum überschreiten. Im Durchschnitt reichen schon 1MB Speicher und 256KB RAM um einfache Verifikationen von einem Netzwerk aus Nodes entgegenzunehmen. Mikrocontroller der RF52 Serie von Nordic Semiconductors sind sehr preisgünstig und kompatibel mit einer Vielzahl von Schließ-Systemen. [4] Somit ließen sich Türen abdecken.

Automaten und Spinde sollten mit dem Mikrocontroller ESP32 von Espressif [3] kompatibel sein, da hier zusätzlich eine Wallet benötigt wird um Transaktionen zu signieren und diese im Blockchain-Netzwerk zu veröffentlichen. Der Chip hat 520KB RAM und bis zu 16MB Speicher.

8.3 Entwurf und Dokumentation

Im vorherigen Kapitel wurde ausgelegt, dass es eine Dokumentation mit Handlungsanweisungen für die Nutzer Aussteller, Lehranstalten und Institutionen, aber auch eine Dokumentation mit Implementierungsanweisungen als Entwicklerleitfaden geben soll, welcher die Schichten und Abläufe im Innern der Software mit konkreten Details beschreibt. Beim Entwurf in diesem Kapitel wird erstmalig nur eine Zusammenfassende Sicht geteilt, da lediglich die Programmierung eines Prototyps entstehen soll. Für alle zukünftigen Versionen wird die Architekturerstellung ausgebaut bzw. gegebenenfalls geteilt.

8.3.1 Betrachtung des Projektkomplexes

Mit dem Educhain Projekt sollen nacheinander drei Meilensteine realisiert werden. In den ersten drei Monaten, soll das Blockchain-Netzwerk aufgesetzt und mit einem digitalen Studentenausweis benutzbar sein. Bis dahin ist es über den Browser und der mobilen App möglich einen Nutzer-Account zu erstellen, in welchen der digitale Ausweis eingefasst wird. Der zweite Meilenstein ist der Ausbau der beiden Benutzerschnittstellen mit allen kontobasierten Use Cases nach fünf Monaten. Hier erwachen die Accounts zum Leben und können mit eingeschränkten Identitätsdaten bestückt und organisiert werden. Ausstellern wird es ermöglicht Identitäten auszugeben, welche Institutionen dann von der Blockchain aus verifizieren können. Mit dem letzten Meilenstein nach weiteren sechs Monaten werden Schlösser und Spinde über IoT-Geräte implementiert. Diese können vollkommen über Accounts benutzt werden. Nach spätestens zwei Jahren soll das System mit dem Hinzufügen von digitalem Bezahlen und der Verwendung von Automaten abgeschlossen werden.

Zu den Geschäftszielen zählen das erweitern und halten des Kundenkreises nach der ersten Benutzung des digitalen hochschulübergreifenden Ausweises. Je mehr Nutzer an unterschiedlichen Lehranstalten partizipieren umso mehr lohnt sich der Beitritt zum System. Nach der Entwicklungszeit soll das Produkt schon erste Gewinne durch eine geringe monatliche Nutzungsgebühr und Hardware erzielt werden. Angesiedelt wird das Projekt im Business-to-Business Markt mit den Hochschulen, welche die Kosten geringfügig am Semesterbetrag anpassen. Die Blockchain ist zudem offen für neue Entwicklungen anderer Studenten, die ihre Funktionalitäten andern zur Verfügung stellen möchten.

Das Projekt wird Anfangs mit zwei Gründern als Projektleiter und Controller gestartet, soll jedoch um zwei weitere Junior Entwickler aus den Bereichen Mobile Development und IoT-Integration ergänzt werden. Beide sollen Blockchain-Know-How sowie gute Kenntnisse in ihren Entwicklungsgebieten vorweisen können.

8.3.2 Erster Entwurf

Das System wird nur mittels zwei Benutzeroberflächen vom Browser und vom Mobiltelefon aus bedient werden. Ebenso gibt es eine Backend-Oberfläche in Form einer Webseite, mit der es Benutzern möglich gemacht wird die Blockchain im Hintergrund zu betrachten und Einträge in ihr zu suchen. Die Interaktion im Browser kann mit Mausklicks und Touch erfolgen. Auf dem Mobiltelefon wird ausschließlich auf die Touch-Oberfläche zurückgegriffen.

Beide Benutzeroberflächen werden in der Sprache JavaScript verfasst. Beim Design greifen beide Benutzeroberflächen auf ein und dasselbe Framework zu, um die Benutzerschnittstellen immer auf einem Niveau zu halten und den Nutzern ein einheitliches Gefühl zu bieten. Für die Entwicklung im Browser wird React [13] als Bibliothek verwendet, für die Mobiltelefone wird mit React Native [14] Support für iOS und Android geliefert. Die ARM Architektur wird vollkommen unterstützt. Die Größe der finalen App überschreitet nach Annahmen 300MB nicht und ist somit auch bei schlechterer Internetverbindung in moderater Zeit herunterzuladen. Die leichtgewichtigen Apps rufen Ihre Informationen lediglich aus der Blockchain ab, was die erste Inbetriebnahme innerhalb von 15 Sekunden möglich machen soll. Mittels Cache wird diese Zeit verkürzt. Für Beide Anwendungen kommunizieren über HTTPS um ein abhörsicheres Übertragen von Datenströmen zu gewährleisten. Dies ist auch für die Kommunikation mit den Schnittstellen von ID-IDEAL [5] und ECHT! [6] nötig. ID-IDEAL händigt einfache Studentenausweise aus, ECHT! wird für Datennachweise zum Einsatz kommen. Beide Anwendungsfälle werden ausgelagert und docken an die Forschungsprojekte des BCCM an. Vorerst sind diese Technologien ohne Lizenz nutzbar. Da es zukünftig geplant ist weitere Technologien und Forschungsprojekte einzubinden, sind Änderung durch das Vorhandensein einer starken Modularität sehr kosteneffizient und schnell umzusetzen. Angepeilt ist es, dass für die Implementierung pro Technologiemodul nicht länger als zwei Monate benötigt wird. Beim Design der Oberflächen greift man auf ein Framework zu um doppelten Zeitaufwand einzusparen.

Als Blockchain Netzwerk im Hintergrund kommt es zur Peer-to-Peer Lösung Ethereum. Ethereum ist die aktuell am meist verwendete Blockchain für Entwickler, mit der Möglichkeit Code der Programmiersprache Solidity auszuführen. Sie dient als Datenbank aller schreibenden Aktionen im Ökosystem. Auf der Anwendungsebene dieser Blockchain werden Accounts, Tokens und Identitätsdaten mittels Smart Contracts abgebildet. Dieses Prinzip löst die ursprüngliche schlüssel-basierte Kontenführung auf der Blockchain ab. Mit der contractbasierter Kontoführung werden eine Menge von Geräte-Schlüsseln an ein Konto gebunden, über welche als einzelnes Benutzerkonto gehandelt wird. Dies ist sehr Benutzerfreundlich, da mehrere Geräte mit unterschiedlichen Rechten eingebunden, sowie endlos viele Backup-Schlüssel erstellt werden können, wenn es zum Verlust eines Gerätes kommt. Das Prinzip ist ebenfalls sehr nutzer-zentriert, da alle gespeicherten Identitätsdaten nur dieser Person gehören und keine Instanz jemals Kopien davon erhält.

Falls Zugriff auf Daten oder Ausweise benötigt wird, so muss der Nutzer aktiv zustimmen. Alle Abfragen und Transaktionen dieser Accounts werden dabei automatisch an die nächst nähere Server-Node weitergegeben, welche noch nicht ausgelastet ist. Verifizierte Transaktionen werden automatisch im Netzwerk verteilt. Aktuell operiert Ethereum noch auf einem PoW Konsens-Algorithmus. Dabei wird Rechenleistung genutzt um die Blockchain zu besichern. Dies soll noch dieses Jahr umgeschlagen werden. Ethereum ist kurz davor seinen Konsensalgorithmus auf PoS zu ändern, wobei das Netzwerk dann mit Kapital besichert wird. Dieses Verfahren ist weniger rechenintensiv, sodass selbst kleine Rechner sich beteiligen können. Geplant ist, dass Hochschulen im Ökosystem diesen Staking-Prozess übernehmen und somit die Educhain absichern. Mit dem Wechsel auf PoS wird auch eine Sharding-Kompatibilität hinzugefügt, welche es ermöglicht, die allgemeine Netzwerk-Skalierbarkeit zu erhöhen. Geplant ist es, ab einer Größe von hundert Server-Nodes Sub-Netzwerken zu erlauben Transaktionen aufzuteilen. Dies sorgt dafür, dass die PoS-Transaktionsgeschwindigkeit von 30 TX/s auf bis zu 300 TX/s steigt. Dies sollte für ein Hochschulnetz genügen. In zukünftigen Architekturen ist zu überlegen, Rollups in das Netzwerk zu integrieren. Mit dieser Technologie können mittels Kryptographie eine Vielzahl von Transaktionen in eine Einzelne geschrumpft werden. Anfangs genügt die ursprüngliche Datenrate jedoch völlig. Bis zur Aufnahme einer Transaktion dauert es etwa 15 Sekunden. Dies ist akzeptabel für eine Wartezeit zum Bezahlen des Mensa-Essens oder dem Öffnen eines Spindes. Die Speicherung des Netzwerkes überschreitet 100GB pro Jahr nicht, sodass auch externe Institutionen mit Leichtigkeit eine Node zum Nachvollzug der Datensätze betreiben können. Im Netzwerk sollte es nie zu Ausfällen, höchstens kleinen Verzögerungen kommen. Durch den dezentralen Ansatz gibt es keinen Single Point of Failure. Eine Fehlertoleranz von bis zu 30% der Netzwerkknoten kann in Kauf genommen werden ohne, dass es im Netzwerk zu Einschränkungen kommt. Durch die Abkapslung der Anwendungen vom Netzwerk, ist diese bei Softwarefehlern nicht betroffen. Es kann höchstens zu vermehrten Transaktionen und dadurch zu einer allgemeinen Verzögerung kommen.

Um von den Anwendungen aus mit der Blockchain interagieren zu können, kommt die Web3-Bibliothek in Verbindung mit der Ethereum Lightwallet zu Einsatz. Mit diesen beiden Technologien ist es möglich Ethereum-basierte Wallets zu erstellen und sie mit einer eigenen Blockchain zu verknüpfen. Im Browser wird die Erweiterung MetaMask [8] verwendet, welche von Chrome, Firefox, Brave und Edge unterstützt wird. Ebenso existiert eine App für die Verwendung mit Safari.

Um mit IoT-Geräten interagieren zu können werden kleine Clients in der Programmiersprache C erstellt. Dabei gibt es zwei verschiedene Arten: Validierer und transaktionsfähige Clients. Schlösser sind Validierer und müssen lediglich Informationen nachprüfen- nämlich ob einer der Ausweise des Mobiltelefons, welches sich mit NFC zum Gerät verbindet, berechtigt ist, die Tür zu öffnen. NFC wurde hier als ITK ausgewählt, da garantiert wird, dass das sich das Gerät mit maximal vier Zentimetern Abstand in unmittelbarer Nähe befindet. Das Schloss wird bei der Einrichtung mit einer Account-Adresse versehen, welcher alle Ausweise beinhaltet, die berechtigt sind die Tür zu öffnen. Dieser Account kann von der Lehranstalt variabel verändert werden. Mittels IN3 [7] Software geben sie die vom Mobiltelefon erhaltenen Ausweisadressen gemeinsam mit der eigenen Ausweisadresse an ein Subsystem von Nodes weiter und lassen von mehreren Instanzen berechnen, ob eine der Ausweise übereinstimmt. Liefert die Mehrzahl der Nodes die richtige Antwort, so öffnet sich die Tür. Für einfache Weitergaben genügen Mikrocontroller der Serie RF52 von Nordic Semiconductor. [4] Diese sind etwa so groß wie eine 10 Cent Münze und passen in internetfähige Schlösser. Solche Schlösser sollen zukünftig mit diesem Chip modifiziert werden. Der Mikrocontroller bietet 1MB Speicher und 256KB RAM. Doch nicht immer reicht so ein kleiner Chip, weswegen für Spinde und Automaten transaktionsfähige Clients benötigt werden. Spinde müssen nicht nur Transaktionen validieren, sondern sollen sich auch selbst verwalten können und bei Verminderung der Mietzeit Rückerstattungen ausführen. Gleiches gilt für Token-Automaten bei denen Euro ein- und ausgezahlt werden um die gleiche Zahl an Token sowie Prüf-Token an das Nutzerkonto bzw. sich selbst auszuzahlen. Hierbei wird der ESP32 Mikrocontroller von Espressif zum Einsatz kommen. Er bietet 520KB RAM und 16MB Speicher und genügt den Anforderungen zum Besitz einer Wallet. [3] In beiden Varianten werden die dazugehörigen Geräte an die Accounts der Hochschulen angebunden.

In Bezug auf die Datensicherheit werden alle Daten im Netzwerk verschlüsselt um nur von berechtigten Personen gelesen zu werden. Zudem fasst die Blockchain-Architektur Transaktionen zu Blöcken zusammen welche in einer Hashkette aufeinander aufbauen und sich gegenseitig absichern. Bevor die Transaktionen ins Netzwerk aufgenommen werden, müssen sie von Knoten bestätigt und geprüft werden. Dies schließt auch doppelte Ausgaben von Token aus. Durch das Verwenden von Wallets in einer Blockchain ist es nötig, dass alle Schreibvorgänge im Netzwerk eine Signatur des Benutzerkontos besitzen. Demzufolge sind Bezahlungen und Identitätsdaten zwangsweise und fälschungssicher an einen Account bzw. eine Wallet-Adresse gebunden.

Im System muss auch der Datenschutz sehr akribisch wahrgenommen werden, da mit Identitätsdaten von Benutzern umgegangen wird. Die Wallet-Daten bleiben vollkommen verschlüsselt und in einer sicheren lokalen Datenbankform gespeichert in der Ethereum Lightwallet. Im Browser werden die Daten sicher in MetaMask verwahrt. [8] Beide Wallets werden bereits täglich hunderttausendfach von Personen verwendet. Benutzer müssen zudem explizit bestätigen, wenn Institutionen oder Aussteller ihre Identitätsdaten einsehen möchten. Der Zugriff kann jederzeit gesperrt werden, da auf der Blockchain keine Kopien versendet werden, sondern die Daten immer wieder neu direkt vom Benutzerkonto abgerufen werden müssen.

Beim Testen wird vorerst ein eigenes Testnetzwerk zum Einsatz kommen, dass Entwicklern ermöglicht neue Technologien vorzeitig in den gleichen Bedingungen, wie dem Hauptnetzwerk zu testen. In Verbindung mit React Native [14] kann so die mobile Anwendung auch während der Laufzeit am Computer debuggt werden. Trotzdem sind alle Softwaremodule und Subsysteme unabhängig voneinander zu testen. Für Benutzereingaben werden über verschiedene Vorlagen und Bibliotheken automatisierte Black-Box Tests getätigt. Smart Contract Funktionalitäten einzelner Methoden werden mit manuellen Szenario-basierten Tests durch unterschiedliche Dummy-Daten eingepflegt. Mit Performance-Tests in der mobilen App beim Erstellen von Identitätsdaten oder PDF-Dateien sind Auslastungen und Wartezeiten zu testen. Restliche Daten werden lediglich von der Blockchain über das Internet geladen und hängen von der Internetgeschwindigkeit ab. Explizit zum Einsatz kommen das JEST Testframework für JavaScript-Benutzeroberflächen. [12] Das Framework bietet viele Funktionalitäten zum Testen von React-Applikationen und beinhaltet Code-Coverage Tools sowie die Möglichkeit zur Screen-Analyse. Gegenüberliegend wird bei Backend-Tests das auf NodeJS basierende Framework MOCHA [11] verwendet, welches eine Vielzahl an Möglichkeiten zum Testen im Innern der App bietet und die Übergabe von Parametern an die Blockchain überprüfen kann. Der Testzyklus geht dort nahtlos über. Verwendet wird TRUFFLE [10] als Framework zum Testen von Solidity Code auf der Blockchain. Dieses Framework ermöglicht sogar lokale Tests um weitere Fehlerquellen herauszufiltern. Beim Testen wird auch die Erstellung verschiedenster Datenformate untersucht. Schnittstellen wie ID-IDEAL liefern JSON, Smart Contracts auf der Blockchain benötigen dieses Format ebenso für ihre ABI. Bei der Erstellung von Identitätsdatensätzen auf der Blockchain werden die Eingaben erst in eine SOL Datei geschrieben, welche dann in Bytecode gewandelt auf die Blockchain transferiert wird. Auch hier muss geprüft werden, ob der Contract-Builder genau arbeitet. Ein weiterer Build-Prozess wird beim Druck von Geräte- und Backupschlüsseln verwendet, wenn PDF-Dateien zu generieren sind.

Im Zusammenhang mit dem Testen steht auch die Portiermöglichkeit des Systems. Wie schon beschrieben kommt es bei Ethereum zu einem Wechsel von PoW zu PoS. Eine Umstellung des Konsensalgorithmus mit der Übernahme des gegenwertigen Datensatzes ist also gewährleistet. Auch der Code von IoT-Geräten muss bei Hardwareumstellungen weiter nutzbar sein und ist bei beiden Mikrocontroller -Produktfamilien gewährleistet, welche in Betracht gezogen wurden.

Bei der Entwicklung selbst kommt die Browser-Entwicklungsumgebung Remix [9] zum Einsatz um Smart Contracts in Solidity zu programmieren und zu testen. Identisch zur Benutzeroberfläche im Internet, wird hier die MetaMask Wallet benutzt. [8] So lässt sich zugleich die Symbiose mit externer Software testen. Für JavaScript und C wird die IDE Visual Studio Code genutzt. Diese Entwicklungsumgebung bietet modular Erweiterungen zum Debuggen und Schreiben von Tests und kann wahlweise über ein integriertes Terminal Builds erstellen. Der Entwicklungsprozess innerhalb des Teams ist durch das stückweise Hinzufügen von neuen Funktionalitäten sehr agil. Verwendet wird die SCRUM Methode um diese Softwareerweiterungen geordnet in Sprints auszuliefern. Gekoppelt wird dieses Prinzip mit Git [15] als Entwicklerplattform um die unterschiedlichen Entwicklungszweige in verschiedenen Stadien unter Kontrolle halten zu können. Repositories und Branches sind ideal um die Entwicklung auf mehrere Mitarbeiter aufzuteilen.

Als Architekturmuster kommen Nebenläufigkeit, Persistenz und das Peer-to-Peer-Prinzip im System vor. Bei der Benutzung der Oberflächen kommt es zu Hintergrundaktualisierungen, während aktiver Verwendung. Dazu gehören beispielsweise Events, welche von der Blockchain aus an den Nutzer geleitet werden. Buchungsvorgänge der Blockchain besitzen wiederum eine hohe Persistenz. Neue Blöcke der Kette referenzieren immer auf den Vorherigen, was zur Folge hat, dass alle Blöcke dezentral und ausfallsicher im Netzwerk der Server-Nodes verwahrt werden müssen um eine Nachprüfbarkeit zu wahren. Im Netzwerk ist jeder Server-Node gleichberechtigt und es kommt zu einem Peer-to-Peer Konstrukt. Alle Accounts und transaktionstätigende Geräte werden über gleichberechtigte Wallets in einem Netzwerk ohne Administration erstellt.

Es kommen ebenso jede Menge an Design Patterns vor. Das Composite Pattern wird für das Kreieren von Identitätsdatensätzen benutzt. Hierbei werden viele einzelne Parameter von persönlichen Informationen und Dokumenten werden zum Erstellen einer großen Ausweis- oder Identitäts-Komponente zusammengetragen. Es bietet sich ebenso an, das Iterator Pattern beim Zugriff von IoT-Geräten auf die Ausweise des Benutzers anzuwenden. Dabei werden von den IoT-Geräten mittels NFC alle Adressen der Ausweise iteriert und an ein Server-Netzwerk übersendet, um festzustellen ob einer der Ausweise über genügend Rechte verfügt, um das IoT-Gerät öffnen öffnen zu können. Dabei muss die interne Struktur nicht offengelegt werden, was einen Sicherheitsgewinn darstellt. Das Builder Pattern wird beim Zusammenfügen eines Identitätsdatensatzes aus Informationen einer Vorlagemaske angewandt. Wenn der Nutzer eine Vorlage des Ausstellers verwendet, um einen Identitätsdatensatz anzulegen, muss beim Erstellen ein Smart Contract generiert werden, welcher alle Informationen und die angegebenen Rechte dazu widerspiegelt. Dies wird als separater Build-Prozess realisiert.

Das Template Pattern wird dann genutzt, wenn Benutzer Vorlagen für die Erstellung von Identitätsdatensätzen verwenden. Mit einer Schablonenmethode ist es möglich, den Nutzern Entwürfe zum Einbinden persönlicher Informationen wie Zertifikate, Ausweise oder Dokumenten zu bieten. In solchen werden dann individuelle Daten hineingegeben. Das Decorator Pattern wird benutzt um Identitätsdaten mit der Funktionalität von Ausweisen zu erweitern. Ein Benutzer erhält Identitätsdaten von einem Aussteller welche zu Beginn an nur für beide ersichtlich sind. Richtet er sich diesen Identitätsdatensatz als Ausweis ein, können auch IoT-Geräte oder andere Institutionen ohne eine erneute Anfrage Informationen von diesem Datensatz validieren. Es wird also eine vorher versteckte Funktion zugänglich.

Das Clone Factory Pattern sollte verwendet werden, um Hashs von Dateien mit zusätzlichen Informationen abzuspeichern. Dabei gibt es einen vordefinierten Smart Contract für das Abspeichern des Dokumenthashs sowie deren angefügten Benutzerinformationen. Soll ein neuer Nachweis mit Nutzerinformationen abgespeichert werden, wird lediglich ein neuer Smart Contract Klon erstellt. Dies hat zur Folge, dass sehr viel Rechenleistung eingespart wird, da nicht mehr der komplette Smart Contract neu erstellt werden muss. Alle Klone verweisen bei Aufrufen, zur ursprünglichen Instanz um die identische Funktionalität zu erhalten.

Es kommen hauptsächlich drei Architekturstile zum Einsatz. Innerhalb der Blockchain existiert ein Aufbau in Schichten. Im Kern arbeitet eine Art dezentrale Virtuelle Maschine zum Ausführen von Bytecode, Messaging und Services. Diese Schicht wird von der darüberlegenden Ebene mit Informationen gespeist und umfasst digitale Signaturen, Transaktionen, Hashing sowie die Blockkette. In der nächst höheren Schicht kommuniziert das Peer-to-Peer Netzwerk, um sich auf einen Konsens aller validen Ereignissen zu einigen. In der finalen Ebene setzten dezentrale Anwendungen über Smart Contracts oder grafische Benutzeroberflächen auf das darunterliegende Fundament auf.

Der zweite Architekturstil ist die objektorientierte Programmierung. Alle abgebildeten Identitätsdaten werden auf der Blockchain als eigene Objekte angesehen. Jeder dieser Wertgegenstände, der mittels eines Smart Contracts dargestellt wird, erhält objektorientierte Funktionen zum Abrufen und Ändern diverser Einstellungen. Dies trägt sich bis zu den Accounts hin, welche als Objekte einer Person gesehen werden.

Als letzter Architekturstil wird Implicit Invocation für Adressen genutzt. Auf der Blockchain verbirgt sich jede Funktionalität hinter einer Adresse. Die Instanz, welche Informationen abrufen möchte weiß nicht, ob sich ein Benutzerkonto, ein Identitätsdatensatz oder nur eine einfache Transaktion dahinter verbirgt. Erst mit dem Aufrufen der besagten Adresse und dem Anwendungsfall vorgegebenen Funktionen wird getestet, ob diese tatsächlich ausgeführt werden kann, oder ob es Falschinformationen sind.

Hier findet sich ein Ablaufdiagramm für das Öffnen eines IoT-Schlosses über die Blockchain. Aufgrund des Umfangs des Ökosystems wurde diese Übersicht lediglich für einen Anwendungsfall erstellt.

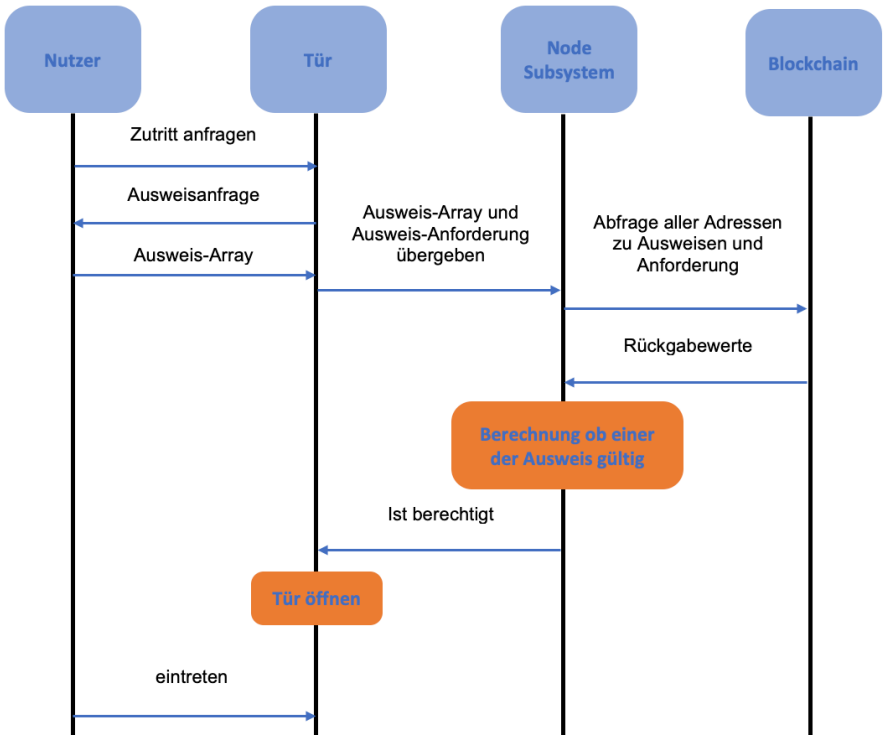

Abbildung 4 Ablaufdiagramm: Tür öffnen

8.4 Szenario-basierte Bewertung

Die Bewertung wird mit der ATAM Methode, ausgeschrieben der „Architecture Tradeoff Analysis Method“, durchgeführt. Regulär nehmen das Bewertungsteam, das Projektteam sowie Skakeholder daran teil. Die Methode ist der Ansatz für ein Umfangreiches Assessment und gibt Aufschluss, wie gut die entworfene Architektur ist. Die ATAM Methode wird in vier Phasen unterteilt: Vorbereitung, Architekturzentrierte Bewertung, Stakeholderzentrierte Bewertung sowie die Nachbearbeitung. In der Vorbereitungs-Phase wird das Bewertungsteam aufgesetzt, die Zusammenarbeit zwischen Projekt und Bewertungsteam abgestimmt und ein erstes Meeting über das Projekt gehalten. In der Nachbereitungs-Phase wird ein Abschlussbericht verfasst, Verbesserungsvorschläge dokumentiert und alle Artefakte festgehalten. Da im Beleg nur eine Person involviert ist und alles bereits schriftlich ausformuliert wurde, entfallen diese Phasen.

8.5 Phase 1

In der ersten Phase werden Geschäftsziele und Architektur präsentiert, die zugrundeliegenden Architekturansätze identifiziert und ein Utility Tree erstellt. Ebenso erfolgt eine erste Analyse der Architekturansätze. Da die Geschäftsziele und Architektur samt deren Ansätze schon im vorherigen Kapitel des Erstentwurfs beschrieben wurde, folgt nun lediglich das Erstellen eines Utility Tree´s sowie die erste Analyse.

Zur Bewertung: Mit dem Architekturansatz der Nebenläufigkeit wird ein fließendes bedienen der App möglich, ohne dass viele Daten gecached werden müssen. Das Nachladen von Informationen stellt für aktuelle Mobiltelefone und Browser keine Herausforderung dar. Es wird für die Nutzung als sehr sinnvoll betrachtet. Auch die beiden Architekturansätze der Persistenz in einem Peer-to-Peer Netzwerk lassen sich durch die Blockchain leicht und ohne eigenen Entwicklungsaufwand verknüpfen. Das System ist ideal und sehr ausfallsicher, da kein zentraler Knotenpunkt entsteht, an welchem Daten verloren gehen können. Zudem werden die Transaktionen der vergangenen Blöcke immer fälschungssicherer für Angriffe. Die Symbiose der Beiden Architekturansätze eignet sich hervorragend, vor allem, da nur die Nutzer selbst Zugriff auf die Daten haben und manuell Berechtigungen geben müssen.

Mit der Bewertung von Pattern und Architekturstilen sieht es ähnlich gut aus. Beim Betrachten der Vielzahl von Szenarien, welche im System existieren kann es schnell unübersichtlich werden. Genau dazu sollten standardisierte Composite Pattern helfen, um Parameter einzelner Identitätsdaten zu kapseln und als ein Objekt abzuspeichern. Das Pattern kommt dabei lediglich in den Apps zum Einsatz, in die Blockchain werden dann fertige Komponenten geladen. Das Pattern eignet sich äußerst gut bei der objektorientierten Programmierung, da die Komponenten nicht nur einfache Datenobjekte darstellen, sondern als Objekte gesehen werden. Diese Parallele zur echten Welt ist hervorragend. Da sich jedoch nahezu das komplette Ökosystem auf der Anwendungsebene mittels Smart Contracts abspielt, ist

es von Bedeutung energie- und lasteneinsparende Konzepte zu suchen. Genau dies bietet das Clone Factory Pattern. Hier können identische Objektformen wie spezifische Ausweise kinderleicht verwendet werden, indem individuelle Ausweisdaten in einen Klon des ursprünglichen Smart Contracts geschrieben werden. Zu beachten ist, dass der initiale Smart Contract keine Funktion zum Löschen implementiert hat, da sonst alle Klone die auf das ursprüngliche Objekt verweisen, nichtig werden könnten. Beim Iterator Pattern werden alle Ausweise nacheinander Durchlaufen. Da hier lediglich Adressen weitergegeben werden und das IoT-Gerät nicht selbst nachprüft, kommt es nicht zu Verzögerungen. Auch große Ausweis-Mengen können so auf dem gleichen Weg iteriert werden. Da die Ausweise mit den Anforderungen der Schlösser verglichen werden, hilft Implicit Invocation. Da verschiedenste Ausweise auch unterschiedliche Datenformen zurückliefern, kann so Unpassendes direkt ausgesiebt werden, wenn Adressen mit einer geforderten Funktion nicht kompatibel sind. Implicit Invocation hilft auch bei der Anonymität, da Dritte oder Instanzen nie genau wissen, welche Daten sich hinter den Adressen verbergen. Es kann nur geprüft werden, indem man passende Funktionsaufrufe kennt oder die Rechte zur Einsicht besitzt. Das spielt dem Decorator Pattern direkt in die Karten, da hier ermöglicht wird, bestimmte Funktionalitäten für Personen freizuschalten, die zuvor nicht ersichtlich waren. Das Nutzen von Template Pattern beim Anwenden von Nutzerschablonen und das daraus folgende Erstellen von Smart Contracts mittels abgekapselten Builder-Patterns bietet sich sehr gut an. Auch hier gibt es keine weiteren Verbesserungsvorschläge. Das Schichtensystem im Blockchain-Netzwerk wird als hervorragend vermerkt, da Ethereum aktuell die führende Blockchain zum Entwickeln von dezentralen Anwendungen ist und es sich auch global seit Jahren bei großen Lasten bewährt hat Lediglich die Transaktionskosten sind durch den globalen Finanzmarkt exorbitant hoch- solche Lasten werden bei einem Schulsystem jedoch nicht erreicht. Durch Open Source ist das Schichtensystem ohne Programmieraufwand inbegriffen, wenn Nodes aufgesetzt werden. Es sollte ein sehr gutes Fundament für alle darauf aufsetzenden Architekturen darstellen.

Auf der nächsten Seite ist ein Ausschnitt des Utility Trees dargestellt. Aufgrund des Umfangs der Softwarearchitektur wurden einzelne Faktoren zusammengefasst und lediglich einige Beispiele der produktspezifischen Anforderungen eingetragen.

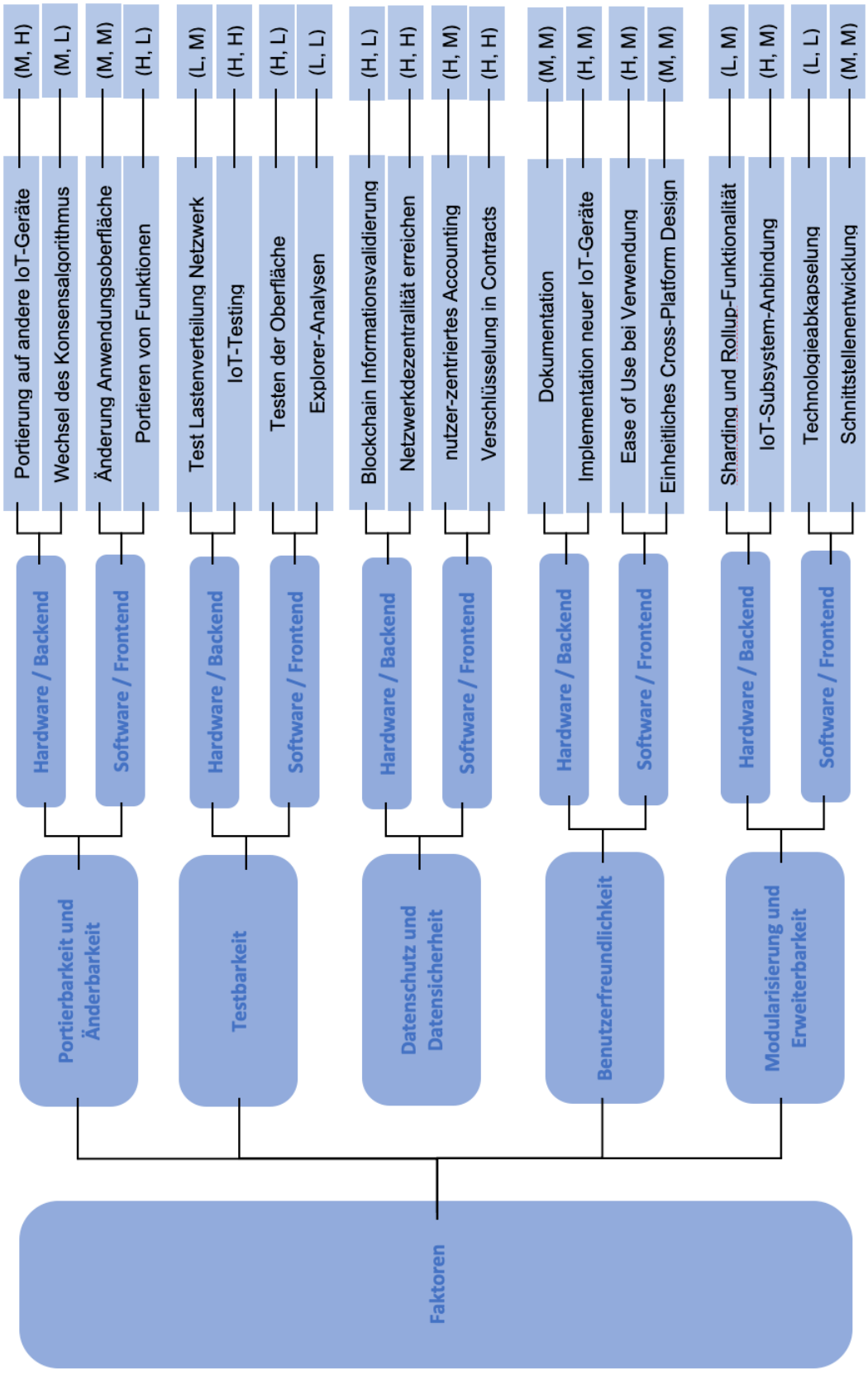

Abbildung 5 Utility Tree

8.6 Phase 2

In der zweiten Phase geht es um eine Stakeholder-zentrierte Bewertung. Es wird ein Brainstorming zu den Szenarien und Use Cases gemacht sowie die dazugehörige Architektur analysiert. Anschließend werden die Ergebnisse präsentiert.

Die ausformulierten Szenarien sind vollständig und stimmen mit den Architekturansätzen überein. Im Anhang können alle Szenarien schrittweise nachgelesen werden. Es gibt keine Einbahnstraßen von Benutzereingaben und alle Daten können bearbeitet oder entfernt werden, nachdem sie hinzugefügt wurden. Dies trifft auf Identitätsdaten, Masken und Ausweise zu. Die Szenarien verweisen zudem auf sich selbst und ergeben einen Kreislauf beim Nutzerverhalten. Auch die im Entwurf beschriebenen Meilensteine stimmen mit den Szenarien überein und Funktionalitäten können nach und nach hinzugefügt werden. So funktionieren eigene Identitätsdatensätze autark von den Datennachweisen und können über Aussteller mittels Masken erweitert werden. Im späteren Verlauf werden auch die IoT-Integration, sowie die Bezahlung unabhängig voneinander möglich sein. Bei der Interaktion mit den Schnittstellen ID-IDEAL [5] und ECHT! [6] sind noch keine Szenarien bekannt, da sich die Projekte bisher noch in der Forschung befinden. Was jedoch bekannt ist: die Kommunikation findet mittels JSON über HTTPS statt und die Software selbst wird mit einer Vielzahl von Blockchains kompatibel sein. Es sind allgemein keine weiteren Verbesserungsvorschläge im Hinblick auf die Architektur entstanden.

9 Architekturumsetzung

Aus dem ersten Entwurf und der Szenario-basierten Bewertung sind keine größeren Bedenken aufgekommen. Somit wird der erste Entwurf für die Architekturumsetzung verwendet. Es soll erstmalig ein Prototyp umgesetzt werden der über einen Smart Contract Datennachweise samt Zeitstempel liefern und Token ausschütten kann.

Da die Umsetzung in diesem Fall nur einen kleinen Teilbereich der gesamten Architektur abbildet, wird der Aufbau kurz beschrieben. Hinweise für Tools und Frameworks sind, dass beim Prototyp lediglich eine Benutzerschnittstelle im Browser gebaut wird. Damit entfällt das Aufsetzen einer mobilen Anwendung sowie der eigenen Blockchain. Als Blockchain-Fundament dient erstmal nur das Ethereum-Testnetzwerk Ropsten. Die Erstellung des Prototyps erfolgt ebenso ohne ein Design-Framework und ohne account-basierte Nutzerverwaltung. Es wird lediglich die Browsererweiterung MetaMask als Wallet mit grafischer Oberfläche verwendet. [8] Transaktionen werden also erst nicht über einen Account, sondern auf direktem Weg ausgeführt.

Die Entwicklung erfolgt wie beschrieben in JavaScript. Mittels der Bibliothek React [13] wird das Abbilden des Benutzerinterfaces realisiert. Für die Programmierung der Blockchain-Anwendung wird über die Remix IDE in Solidity gearbeitet. [9] Um sich von der dApp aus mit der Blockchain zu verbinden wird die Web3 Bibliothek in JavaScript verwendet. Die Schichtenarchitektur der Blockchain im Hintergrund bleibt bestehen. Für die Umsetzung des Smart Contracts wird ein Clone Factory Pattern benutzt. Für die Verwaltung dient GitHub, sodass die dApp über den Dienst Netlify auf einer Webseite im Browser veröffentlicht werden kann. Die grobe Interaktion im Prototyp wird anhand des folgenden Bildes ersichtlich.

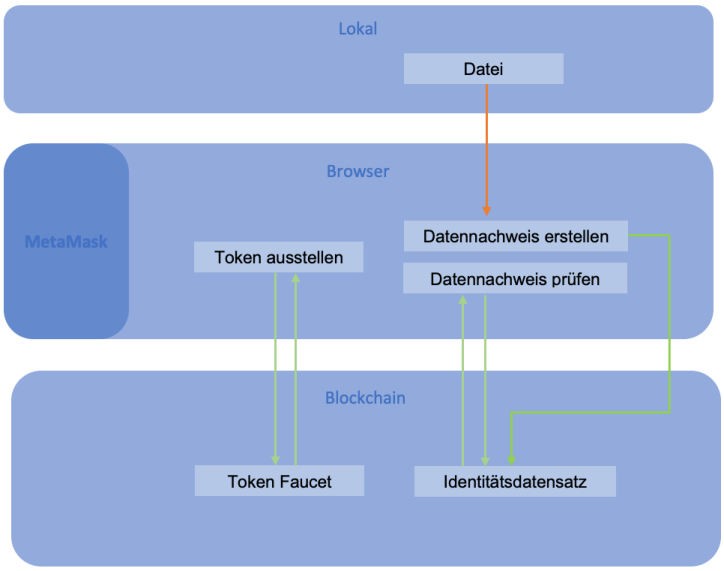

Abbildung 6 Interaktion im Prototyp

10 Exemplarische Umsetzung

In diesem Kapitel wird beschrieben, wie der vorgestellte Prototyp aus dem vorherigen Kapitel umgesetzt wird. Dazu werden Erläuterungen zum verwendeten Pattern, der Benutzeroberfläche im Prototyp sowie Installationshinweise gegeben.

10.1 Design Pattern

Im Smart Contract kommt das Clone Factory Pattern zum Einsatz. Dabei gibt es einen vordefinierten Clone Factory Smart Contract welcher über einen Operator eingebunden wird. Bei der Erstellung des Operators, wird ebenso eine DataProof Contract-Instanz eingebettet. Der Operator verlinkt also DataProof Contracts, welche Informationen zu den Datensätzen erhalten, mit den eigentlichen Dateihashs. Soll nun also ein neuer Nachweis mit Nutzerinformationen abgespeichert werden, wird der Operator aufgerufen. Dieser erstellt einen Klon von der DataProof-Instanz und verlinkt deren Adresse mit dem Dateihash. Alle Klone referenzieren auf ein DataProof Element, ohne selbst die Funktionalitäten zu besitzen. Somit wird sehr viel Rechenleistung eingespart, da nicht ständig komplette Smart Contracts neu erstellt werden müssen, sondern alle Klone lediglich „wissen“ müssen, wo die Funktion steht. Die Funktionalität bleibt dieselbe. In der Abbildung sieht man die Verbindungen untereinander. Will man einen Datensatz nachprüfen, so ruft man den Operator mit dem Dateihash auf und bekommt die passende Adresse des Smart Contracts zurück. Diese ruft man im nächsten Schritt auf. Nun leitet der Contract der Adresse an die ursprüngliche Instanz weiter um die Funktionalität gewährleisten zu können.

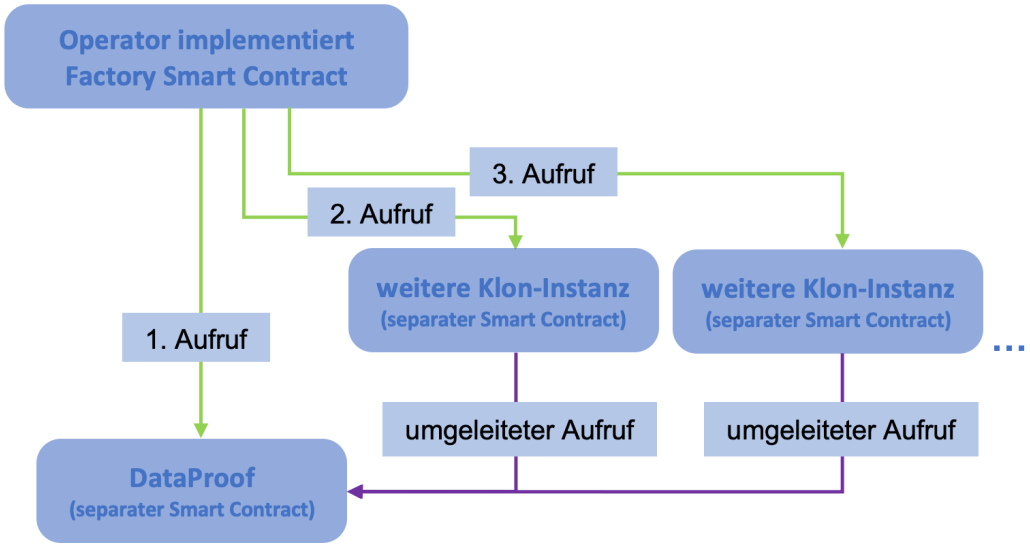

Abbildung 7 Clone Factory Pattern

10.2 Prototyp

Der Prototyp läuft wie bereits erwähnt in der Browseroberfläche. Die Browsererweiterung MetaMask muss dazu vom Benutzer installiert worden sein. In diesem Kapitel finden Sie nun Bilder und eine Beschreibung des Prototyps.

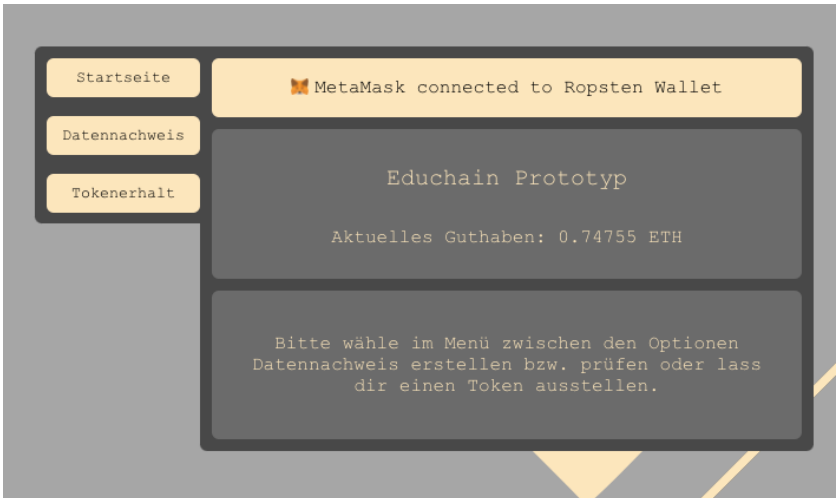

Abbildung 8 Vorschau: Hauptmenü

Ruft man den Prototyp auf, so befindet man sich im Startbildschirm wieder. Von hier aus kann man über die Seitenleiste navigieren. Wählt man Datennachweis, so erhält man die Möglichkeit einen Datennachweis zu erstellen oder zu prüfen.

Abbildung 9 Vorschau: Dateinachweis erstellen

Beim Erstellen eines Datennachweises, wählt man eine Datei von seinem Gerätespeicher aus und gibt eine Beschreibung sowie einen Namen an. Die Informationen sowie der Hash der Datei werden auf der Blockchain mit aktuellem Datum und Adresse versehen. Die Datei bleibt jedoch lokal auf dem Gerät.

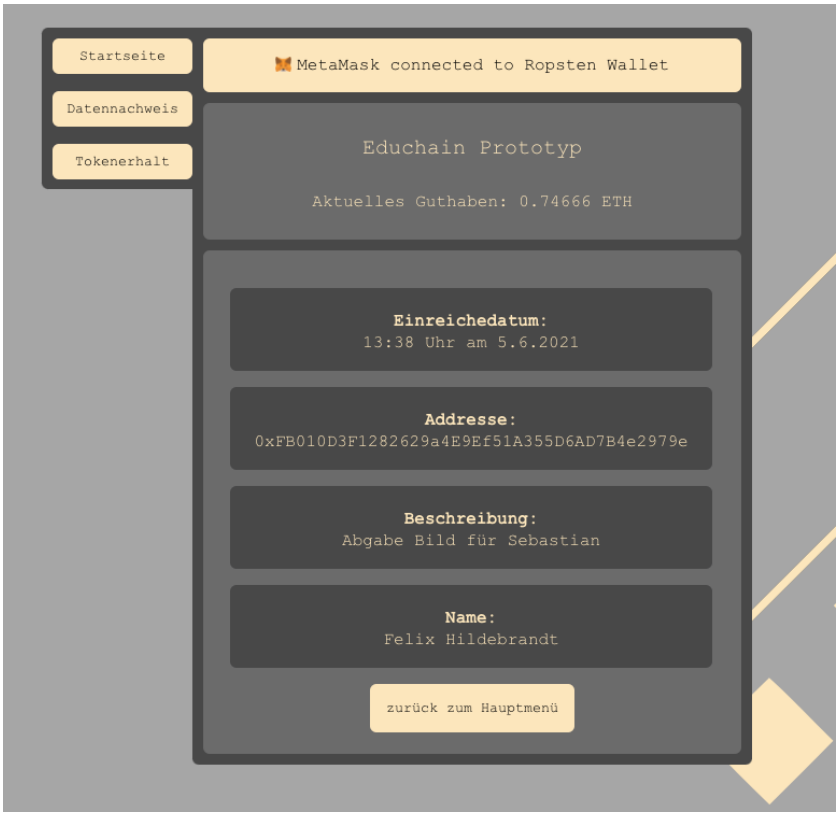

Abbildung 10 Vorschau: Dateinachweis erhalten

Möchte man einen Datennachweis prüfen, so muss man lediglich die Datei von seinem Gerät auswählen. Von dieser wird lokal der Hash erstellt und auf der Blockchain abgerufen, ob zu diesem Hash bereits ein Nachweis eingegangen ist. Falls ja, kann der Benutzer diese Informationen anfragen und erhält eine Übersicht.

Abbildung 11 Vorschau: Tokenerhalt

Möchte man selbst Educhain Token erwerben, so kann man dies auch anfragen. Es wurden auf der Blockchain ERC20 Tokens erstellt und an einen eigenen Faucet übertragen, welcher diese an Wallets ausgibt. Aller zehn Minuten kann der Benutzer einen Token erhalten, mit dem er zukünftig gegebenenfalls Gimmicks freischalten kann. Um den Token in seiner MetaMask Wallet sehen zu können, muss man seine Adresse aus der Oberfläche kopieren und diese beim Hinzufügen eines neuen Tokens einfügen. Er besitzt keine Dezimalstellen.

10.3 Installation

Die Installationsanweisungen werden lediglich benötigt, wenn die App lokal im Browser laufen soll. Alternativ kann auch die Webseite aufgerufen werden um alle Funktionalitäten zu nutzen.

Webbrowser:

Rufen Sie folgende URL auf

```
educhain-mw.netlify.app
```

Lokal:

1. Entpacken und wechseln Sie in das Archiv

   ```
   cd educhain_prototype
   ```

2. Installieren Sie die App

   ```
   npm i
   ```

3. Starten Sie das Programm

   ```
   npm start
   ```

11 Zusammenfassung

Das Projekt im Kurs Architektur komplexer Softwarearchitektur hat dem Educhain-Konzept im Kurs E-Entrepreneurship II einen ersten Prototyp, sowie einen umfangreichen Plan zur Softwareerstellung beigesteuert. Dies dient hervorragend zur Untermauerung der Anwendungsfälle und zeigt das Bestreben innerhalb des Studiums, Erstelltes auch noch zukünftig nutzen zu können. Generell bin ich sehr zufrieden mit der Dokumentation der Architektur, welche dem Educhain-Kosmos zugutekommen kann, wenn ein Forschungsprojekt entsteht. Die Bewertung der Softwareteile war aufgrund meiner Erfahrungen und der Einzelarbeit nicht voll möglich. In einer weiteren Instanz, werden sich Mitarbeiter nochmals mit der Begutachtung auseinandersetzen müssen.

Zukünftig ist denkbar, den ersten Prototypen auf einer eigenen Blockchain zu betreiben und weitere Stabilitätstests durchzuführen. Es kann durch die tiefreichende Ausarbeitung des Belegs ohne große Umstände mit der Erstellung angefangen werden.

Anlagen

Anlage 1: Anwendungsvorgehen…… A-I

Anlage 2: Clone Factory Pattern…… A-II

Anlage 1: Anwendungsvorgehen

Prozesse der Benutzer und des Ökosystems

Szenario beim Account erstellen

1. Der Nutzer ruft Educhain von einem nicht verbundenen Gerät oder Browser auf
2. Der Nutzer wählt die Funktion „Account generieren“
3. Der Nutzer muss ein Passwort für die App hinterlegen
4. Wallet wird generieren
5. EDUC zur Erstellung des SCA anfordern
6. SCA erstellen
7. BWK generieren und mit SCA verknüpfen
8. Home-Bildschirm anzeigen

Szenario beim Einloggen

1. Der Nutzer ruft Educhain von einem verbundenen Gerät oder Browser auf
    a. Mobile App startet
        i. Nutzer gibt sein gesetztes Passwort ein
        ii. Passwort wird überprüft und falls korrekt Zugriff gewährt
        iii. Gespeicherter DWK wir geladen
    b. Browseranbindung startet
        i. Verknüpfung zur Wallet wird gesucht
        ii. Nutzer gibt Wallet-Passwort ein
        iii. Verknüpfter DWK wird abgerufen
2. Home-Bildschirm mit Account-Adresse wird angezeigt

Szenario beim Account-Adresse anzeigen

1. *Szenario „Einloggen“*
2. Der Nutzer wählt „Adresse“ in der oberen rechten Ecke
3. Erstellung eines QR-Codes der Account Adresse
4. Es wird die Account-Adresse als Text und QR-Code angezeigt

Szenario beim Zugriffsschlüssel anlegen

1. Der Nutzer ruft Educhain von einem verbundenen Gerät oder Browser auf
2. Der Nutzer wählt „Zugriffsschlüssel“ im Hauptmenü
3. Der Nutzer wählt „Zugriffschlüssel anlegen“ aus der Liste an Optionen
4. Auswahl an „Backup-Keys“ oder „Geräteschlüssel“
    a. Nutzer wählt „Backup-Keys“
        i. BWK generieren und mit SCA verknüpfen
    b. Nutzer wählt „Geräteschlüssel“
        i. Anzeigen einer Rechtemaske für die Generierung
        ii. Nutzer wählt passende Rechte und klickt „erstellen“
        iii. DWK mit Rechten generieren und mit SCA verknüpfen

Szenario beim Zugriffsschlüssel ansehen

1. Der Nutzer ruft Educhain von einem verbundenen Gerät oder Browser auf
2. Der Nutzer wählt „Zugriffsschlüssel“ im Hauptmenü
3. Der Nutzer wählt „Zugriffschlüssel ansehen“ aus der Liste an Optionen
4. Auswahl an „Backup-Keys“ oder „Geräteschlüssel“
    a. Nutzer wählt „Backup-Keys“
        i. Es werden alle BWK samt ihren Details angezeigt
    b. Nutzer wählt „Geräteschlüssel“
        i. Es werden alle DWK samt ihren Details angezeigt

Szenario beim Zugriffsschlüssel speichern

1. *Szenario „Zugriffsschlüssel ansehen“*
2. Der Nutzer wählt BWK oder DWK Ansicht
3. Der Nutzer wählt gewünschten Listeneintrag aus
4. Popup mit Auswahl „Textdatei speichern“, „QR-Code speichern“ oder „Drucken“
    a. Nutzer wählt „Textdatei speichern“
        i. BWK wird als Datei erstellt
        ii. Betriebssystem-spezifisches Popup zum speichern
        iii. Nutzer wählt Speicherort
        iv. Datei wird transferiert
        v. Popup „erfolgreich“
        vi. Ausgangsliste wird angezeigt
    b. Nutzer wählt „QR-Code speichern“
        i. BWK wird als QR-Bild generiert
        ii. Betriebssystem-spezifisches Popup zum speichern
        iii. Nutzer wählt Speicherort
        iv. Bild wird transferiert
        v. Popup „erfolgreich“
        vi. Ausgangsliste wird angezeigt
    c. Nutzer wählt „Drucken“
        i. Temporäre PDF-Datei mit BWK als Text und Bild wird erstellt
        ii. Betriebssystem-spezifisches Popup zum drucken
        iii. Nutzer wählt Drucker und bestätigt
        iv. PDF wird gedruckt
        v. PDF wird gelöscht
        vi. Popup „erfolgreich“
        vii. Ausgangsliste wird angezeigt

Szenario beim Zugriffsschlüssel löschen

1. *Szenario „Einloggen“*
2. Der Nutzer wählt „Zugriffsschlüssel“ im Hauptmenü
3. Der Nutzer wählt „Zugriffschlüssel löschen“ aus der Liste an Optionen
4. Auswahl an „Backup-Keys“ oder „Geräteschlüssel“
    a. Nutzer wählt „Backup-Keys“
        i. Es werden alle BWK angezeigt, die das Gerät löschen darf
        ii. Der Nutzer wählt den BWK den er löschen möchte
        iii. Ein weiteres Fenster zum bestätigen erscheint
        iv. Nutzer bestätigt
        v. Verknüpfung mit BWK wird entfernt
    b. Nutzer wählt „Geräteschlüssel“
        i. Es werden alle DWK angezeigt, die das Gerät löschen darf
        ii. Der Nutzer wählt den DWK den er löschen möchte
        iii. Ein weiteres Fenster zum bestätigen erscheint
        iv. Nutzer bestätigt
        v. Verknüpfung mit DWK wird entfernt

Szenario beim Account wiederherstellen

1. Der Nutzer ruft Educhain von einem nicht verbundenen Gerät oder Browser auf
2. Der Nutzer wählt „Account wiederherstellen“
3. Feld mit Eingabe für BWK und erscheint
4. Popup zur Webcam/Kamera-Nutzung
    a. Nutzer bestätigt
        i. Feld mit Eingabe zeigt Webcam/Kamera-Input
        ii. Nutzer zeigt BWK als QR-Code von Foto oder Gerät
        iii. QR-Code wird in BWK übersetzt
        iv. BWK wird in aktiven DWK umgewandelt
        v. Account wird wiederhergestellt
        vi. Home-Bildschirm anzeigen
    b. Nutzer verweigert
        i. Nutzer gibt BWK ein und bestätigt
        ii. BWK wird in aktiven DWK umgewandelt
        iii. Account wird wiederhergestellt
        iv. Home-Bildschirm anzeigen

Szenario beim Account verknüpfen

1. Der Nutzer ruft Educhain von einem nicht verbundenen Gerät oder Browser auf
2. Der Nutzer wählt „Account verknüpfen“
3. Feld mit Eingabe für DWK und erscheint
4. Popup zur Webcam/Kamera-Nutzung
    a. Nutzer bestätigt
        i. Feld mit Eingabe zeigt Webcam/Kamera-Input
        ii. Nutzer zeigt DWK als QR-Code von Foto oder Gerät
        iii. QR-Code wird in DWK übersetzt
        iv. DWK wird als aktiv gekennzeichnet
        v. Account wird verknüpft
        vi. Home-Bildschirm anzeigen
    b. Nutzer verweigert
        i. Nutzer gibt DWK ein und bestätigt
        ii. DWK wird als aktiv gekennzeichnet
        iii. Account wird verknüpft
        iv. Home-Bildschirm anzeigen

Szenario beim Identitätsdaten anzeigen

1. Der Nutzer ruft Educhain von einem verbundenen Gerät oder Browser auf
2. Der Nutzer wählt „Identitätsdaten“ im Hauptmenü
3. Anzeigen einer Übersicht aller gebundener Daten
4. Menü „Eigene Daten“ oder „Daten Anderer“ kann gewählt werden
    a. Der Nutzer wählt „Eigene Daten“
        i. Übersicht wird gefiltert und zeigt nur noch eigene Daten
    b. Der Nutzer wählt „Eigene Ausweise“
        i. Übersicht wird gefiltert und zeigt nur noch Ausweise
    c. Der Nutzer wählt „Daten Anderer“
        i. Übersicht wird gefiltert und zeigt nur noch Daten Anderer
5. Der Nutzer wählt gewünschten Datensatz aus und sieht Details

Szenario beim Daten-Masken anlegen

1. *Szenario „Einloggen“*
2. Der Aussteller wählt „Daten-Maske“
3. Liste von allen Masken des Ausstellers wird angezeigt
4. Der Aussteller wählt „Neue Maske mittels JSON implementieren“
5. Popup zum Hochladen einer JSON-Informationshierarchie
6. Der Aussteller klickt „einbinden“
7. Lokal wird ein Informations-Schema erzeugt
8. Der Aussteller bestätigt mit „jetzt hinzufügen“
9. Anhand des Schemas wird eine Maske in Form eines SC erstellt
10. Die Maske wird mit dem Aussteller-Account verknüpft

Szenario beim Daten-Masken anzeigen

1. *Szenario „Einloggen“*
2. Der Aussteller wählt „Daten-Maske“
3. Liste von allen Masken des Ausstellers wird angezeigt

Szenario beim Daten-Masken löschen

1. *Szenario „Einloggen“*
2. Der Aussteller wählt „Daten-Maske“
3. Liste von allen Masken des Ausstellers wird angezeigt
4. Der Aussteller wählt zu löschenden Datensatz aus
5. Der Aussteller wählt „löschen“
6. Popup zur Absicherung des Löschvorgangs
7. Der Aussteller bestätigt „einverstanden“
8. Transaktion zum Aufheben der Verknüpfung und Löschen des SC der Maske mit dem Account des Ausstellers wird ausgeführt
9. Die Maske wurde vom Account entfernt

Szenario beim Identitätsdaten hinzufügen

1. *Szenario „Identitätsdaten anzeigen“*
2. Der Nutzer wählt „Identitätsdaten hinzufügen“ aus der Liste an Optionen
3. Auswahl „eigenen Daten verknüpfen“ oder „Aussteller anfragen“
    a. Der Nutzer wählt „eigene Daten verknüpfen“
        i. Liste mit allen kompatiblen NFT-Masken wird gezeigt
        ii. Der Nutzer wählt gewünschte Vorlage
        iii. Der Nutzer bindet Informationen und Dateien ein
        iv. Der Nutzer bestätigt
        v. Aus Maske wird NFT auf der Blockchain erzeugt
        vi. NFT wird dem SCA hinzugefügt
        vii. Identitätsdaten-Fenster anzeigen
    b. Der Nutzer wählt „Maske von Aussteller wählen“
        i. Ausstellersuche mittels Account-Adresse wird gezeigt
        ii. Der Nutzer wählt seinen Aussteller
        iii. Liste mit allen Eingabemasken des Ausstellers wird gezeigt
        iv. Der Nutzer wählt gewünschte Maske
        v. Der Nutzer bindet Informationen und Dateien ein
        vi. Der Nutzer bestätigt
        vii. Anfrage aus Maske geht an den Aussteller
        viii. Szenario „Identitätsdaten ausstellen“
        ix. Ausstellung des Daten-Token bestätigen
        x. Daten-Token wird mit Account verknüpft
        xi. Identitätsdaten-Übersicht anzeigen

Szenario beim Identitätsdaten bearbeiten

1. Szenario „Identitätsdaten anzeigen“ *„Eigene Daten“*
2. Der Nutzer wählt den gewünschten Eintrag aus
3. Der Nutzer wählt „bearbeiten“ hinter einem zu änderbaren Parameter
4. Es öffnet sich eine neue Maske zum bearbeiten
5. Der Vorherige Inhalte wird übertragen
6. Der Nutzer ändert den Inhalt oder die Datei
7. Der Nutzer bestätigt mit „fertig“
8. Eine neue Transaktion wird an den SC gesendet und macht die Änderung wirksam

Szenario beim Identitätsdaten ausstellen

11. *Szenario „Einloggen“*
12. Der Aussteller wählt „Ausstellungsanfragen“
13. Eine Liste aller erhaltener Ausstellungsanfragen wird angezeigt
14. Der Aussteller wählt die gewünschte Anfrage aus
15. Es werden bereits ausgefüllte Daten angezeigt, die überprüft werden können
    a. Der Aussteller fügt zusätzlich eigene Dateien hinzufügen
    b. Der Aussteller fügt zusätzlich keine eigenen Dateien hinzufügen
16. Der Aussteller bestätigt „Identitätsdatensatz erstellen“
17. Ein Fenster mit Rechten und Zusatzeigenschaften wird gezeigt
18. Der Aussteller bestätigt „Identitätsdatensatz erstellen“
19. Mittels Maske wird ein verifizierter Daten-Token generiert samt Rechten
20. Der Aussteller bestätigt „Identitätsdatensatz verifizieren und zuweisen“
21. Der Daten-Token wird vom Aussteller signiert
22. Der Daten-Token wird an den anfragenden Account gesendet

Szenario beim Zugriff auf Identitätsdaten gewähren

1. *Szenario „Einloggen“*
2. Der Nutzer wählt „Zugriffanfragen“
3. Eine Liste aller erhaltener Zugriffsanfragen wird angezeigt
4. Der Nutzer wählt die gewünschte Anfrage aus
5. Der Nutzer wählt „Zugriff erteilen“
6. Eine Transaktion wird getätigt, welche der Institutionsadresse die Rechte für das Einsehen des Datensatzes ermöglicht

Szenario beim Zugriff auf Identitätsdaten entziehen

1. *Szenario „Einloggen“*
2. Der Nutzer wählt „Zugriffe“
3. Eine Liste aller ausgestellter Zugriffe wird angezeigt
4. Der Nutzer wählt die gewünschten Zugriffserlaubnis aus
5. Der Nutzer wählt „Zugriff entziehen“
6. Eine Transaktion wird getätigt, welche der Institutionsadresse die Rechte für das Einsehen des Datensatzes entzieht

Szenario beim Identitätsdaten erhalten

1. *Szenario „Einloggen“*
2. Das Institut wählt „Identitätsdaten anzeigen“
3. Das Institut wählt „Identitätsdaten erhalten“
4. Popup zur Webcam/Kamera-Nutzung
    a. Nutzer bestätigt
        i. Feld mit Eingabe zeigt Webcam/Kamera-Input
        ii. Nutzer zeigt Identitätsdaten-Adresse als QR-Code von Foto oder Gerät
        iii. QR-Code wird in Adresse übersetzt
    b. Nutzer verweigert
        i. Institution gibt Adresse manuell ein
5. Die Institution bestätigt
6. *Szenario „Zugriff auf Identitätsdaten gewähren“*
7. Institution wurde Zugriff erteilt
8. *Szenario „Identitätsdaten anzeigen“*
9. Die Institution wählt den zugehörigen Datensatz aus
10. Informationen werden abgerufen
11. Informationen werden grafisch angezeigt

Szenario beim Identitätsdaten selbst entfernen

1. *Szenario „Identitätsdaten anzeigen“*
2. Der Aussteller wählt zu löschenden Datensatz aus
3. Der Aussteller wählt „löschen“
4. Popup zur Absicherung des Löschvorgangs
5. Der Aussteller bestätigt „einverstanden“
6. Transaktion zum Aufheben der Verknüpfung und dem Löschen des SC der Maske mit dem Account des Ausstellers wird ausgeführt
7. Die Maske wurde vom Account entfernt

Szenario beim Identitätsdaten streichen

1. *Szenario „Einloggen“*
2. Der Aussteller wählt „Ausgestellte Identitätsdaten“
3. Eine Liste aller erhaltener Ausstellungsanfragen samt Suchleiste wird angezeigt
4. Der Aussteller sucht nach der gewünschten Identitätsdatensatz
5. Der Aussteller wählt den Identitätsdatensatz aus
6. Der Aussteller wählt „Dieser Identitätsdatensatz ist nicht länger gültig“
7. Popup zur Absicherung der Aufhebung
8. Der Aussteller bestätigt „einverstanden“
9. Transaktion zum Aufheben der Verknüpfung und Löschen des SC des Identitätsdatensatzes mit allen verknüpften Accounts wird ausgeführt
10. Der Identitätsdatensatz wurde vom Account entfernt

Szenario beim digitalen Ausweis erstellen

1. *Szenario „Identitätsdaten anzeigen“*
2. *Nutzer wählt „Identitätsdaten als Ausweis benutzen“*
3. *Eine Transaktion wird ausgeführt, welche den Hash des Datensatzes öffentlich einsehbar macht*

Szenario beim digitalen Ausweis entfernen

1. *Szenario „Identitätsdaten anzeigen“*
2. *Nutzer wählt „Ausweis entfernen“*
3. *Eine Transaktion wird ausgeführt, welche den Zugriff auf den Hash des Datensatzes sperrt*

Szenario beim digitalen Ausweisen

1. *Institution fragt Ausweis-Adresse ab (Text oder QR-Code)*
2. *Szenario „Einloggen“*
3. *Szenario „Identitätsdaten anzeigen“*
4. *Nutzer zeigt QR-Code oder übermittelt Ausweis-Adresse an die Institution*
5. *Daten werden an eigene Node weitergegeben*
6. *Node prüft mit Kryptographie, ob Ausweis und deren Aussteller korrekt sind*
7. *Node bestätigt Identität falls Ausweisdaten von Nodes und verknüpftem Account identisch und korrekt sind, ohne deren kompletten Inhalt sehen zu müssen/können*

Szenario beim Geld einzahlen

1. *Szenario „Einloggen“*
2. Der Nutzer hält sein Mobiltelefon an den Token-Automat
3. Die eigene Adresse wird übermittelt (NFC)
4. Der Nutzer zahlt den gewünschten Geldbetrag ein
5. Der Token-Automat prüft das eingezahlte Geld
6. Der Token-Automat ruft den SC des Tokens auf
7. Der Total Supply wird mittels Transaktion vom Token-Automat erhöht
8. Dem Token-Automat-Konto werden neue Prüf-Token sowie EDUT zugewiesen
9. Der Token-Automat transferiert EDUT an den Account des Benutzers
10. Der Token-Automat transferiert Prüf-Token an seinen Account der Institution, dies dient als Nachweis der Rücklage

Szenario beim Geld auszahlen

1. *Szenario „Einloggen“*
2. Der Nutzer hält sein Mobiltelefon an den Token-Automat
3. Die eigene Adresse wird an den Token-Automat übertragen (NFC)
4. Der Nutzer gibt den Auszahlungsbetrag am Token-Automaten ein
5. Der Token-Automat sendet eine unsignierte Transaktion mit dem identischen Betrag an den Account des Nutzers (NFC)
6. Der Nutzer erhält eine Anfrage zum Bestätigen der Transaktion
7. Der Nutzer bestätigt
8. Die Wallet der App signiert und führt die Transaktion aus
9. Der Token-Automat sendet die erhaltenen EDUT samt Prüf-Token an eine Burn-Adresse, sodass alle Token samt Rücklage-Nachweis zerstört und der Total Supply des SC des Tokens um den identischen Betrag verringert werden
10. Der Token-Automat zahlt Geldbetrag aus
11. Der Nutzer entnimmt das Geld aus dem Automaten

Szenario beim digitalen Bezahlen

1. *Szenario „Einloggen“*
2. Der Nutzer wählt „Bezahlen“ im Hauptmenü
3. Popup zur Webcam/Kamera-Nutzung
    a. Der Nutzer bestätigt
        i. Feld mit Eingabe zeigt Webcam/Kamera-Input
        ii. Der Empfänger/Nutzer zeigt Empfangsadresse von Foto oder Gerät
        iii. Adresse wird als Empfangsadresse eingetragen
    b. Der Nutzer verweigert
        i. Der Nutzer gibt Empfangsadresse manuell ein
4. Der Nutzer gibt Betrag der Zahlung ein
5. Der Nutzer bestätigt die Transaktion
6. Die Transaktion wird auf der Blockchain ausgeführt

Szenario beim Zugänge erhalten

1. *Szenario „Einloggen“*
2. *Der Nutzer hält Handy an Intelligentes Schloss (NFC)*
3. *IoT-Gerät fragt Ausweise des Accounts ab (NFC)*
4. *Gerät übermittelt erhaltene Adressen intern gespeicherter Account-Adresse an ein Sub-Netzwerk von Nodes*
5. *Nodes rufen Adresse auf und prüfen mit Kryptographie unabhängig, ob Ausweisdaten und deren Aussteller korrekt sind, ohne deren kompletten Inhalt sehen zu müssen/können*
6. *Nodes vergleichen, ob einer der Ausweise den Anforderungen aus der gespeicherten Account-Adresse genügt*
7. *Nodes geben unabhängig Antwort an IoT-Gerät zurück*
8. *IoT-Gerät gewährt Zugriff, falls die Mehrheit der Nodes das identische Ergebnis rücksenden und Daten mit den Accounts übereinstimmen*

Szenario beim Mieten anzeigen

5. *Szenario „Einloggen“*
6. Der Nutzer wählt „Mieten“ im Hauptmenü
7. Es wird eine Liste aller aktueller Mietvorgänge angezeigt

Szenario beim Spind mieten und benutzen

1. *Szenario „Einloggen“*
2. *Der Nutzer hält Mobiltelefon an Intelligenten Spind (NFC)*
3. *IoT-Gerät fragt Adresse des Accounts ab (NFC)*
4. *Gerät übermittelt erhaltene Account-Adresse und verknüpfte Rechte-Adresse an ein Sub-Netzwerk von Nodes*
5. *Nodes rufen Adresse und deren Ausweise auf und prüfen mit Kryptographie unabhängig, ob Ausweisdaten und deren Aussteller korrekt sind, ohne deren kompletten Inhalt sehen zu müssen/können*
6. *Nodes geben unabhängig Antwort an IoT-Gerät zurück*
7. *IoT-Gerät prüft, ob die Mehrheit der Nodes das identische Ergebnis rücksenden und ob Account zur aktuellen Zeit noch zum Öffnen berechtigt ist*
    a. *Account ist nicht berechtigt*
        i. *IoT-Gerät überträgt seine Adresse*
        ii. *Das Bezahlfenster öffnet sich*
        iii. *Die erhaltene Spind-Adresse wird eingefügt*
        iv. *Der Nutzer gibt den gewünschten Zeitraum an*
        v. *Der Nutzer bestätigt die Transaktion*
        vi. *Die Miete wird mit einer Transaktion an den Spind gesendet*
        vii. *Es wird ein neuer Identitätsdatensatz mit allen Mietinformationen erstellt und dem Nutzer zugewiesen*
        viii. *Der Identitätsdatensatz wird Automatisch zum Ausweis*
        ix. *Der Spind öffnet sich*
        x. *Es wird die Miet-Übersicht angezeigt*
    b. *Account ist berechtigt*
        i. *Spind öffnet sich*

Szenario beim Spind-Zeit bearbeiten

1. *Szenario „Mieten anzeigen“*
2. Der Nutzer wählt gewünschten Miet-Fall aus
3. Der Nutzer verlängert oder verkürzt die Mietzeit
4. Der Nutzer gibt neues Ablaufdatum an
5. Eine Guthaben-Vorschau für die Zahlung/Rückerstattung wird angezeigt
6. Der Nutzer bestätigt
    a. Miet-Zeit wird verlängert
        i. Es wird eine Transaktion mit mehr Guthaben an die Adresse des IoT-Gerätes gesendet
        ii. Der Identitätsdatensatz zum Miet-Fall wird bearbeitet und der Zeitraum verlängert
        iii. Eine neue Transaktion wird an den SC des Identitätsdatensatzes gesendet und macht die Änderung wirksam
    b. Miet-Zeit verkürzen
        i. Es wird eine unsignierte Transaktion mit einer Rückerstattung an die Adresse des IoT-Gerätes gesendet
        ii. IoT-Gerät fragt Adresse der Transaktion ab
        ii. Gerät übermittelt erhaltene Account-Adresse und verknüpfte Rechte-Adresse an ein Sub-Netzwerk von Nodes
        iii. Nodes rufen Adresse und deren Ausweise auf und prüfen mit Kryptographie unabhängig, ob Ausweisdaten und deren Aussteller korrekt sind, ohne deren kompletten Inhalt sehen zu müssen/können
        iv. Nodes geben unabhängig Antwort an IoT-Gerät zurück
        v. IoT-Gerät prüft, ob die Mehrheit der Nodes das identische Ergebnis rücksenden und ob Account zur aktuellen Zeit noch zum Öffnen berechtigt ist
        vi. Das IoT-Gerät signiert und veröffentlicht die Transaktion über das Netzwerk
        vii. Der Nutzer erhält die Rückerstattung
        iii. Der Identitätsdatensatz zum Miet-Fall wird bearbeitet und der Zeitraum verkürzt
        iv. Eine neue Transaktion wird an den SC des Identitätsdatensatzes gesendet und macht die Änderung wirksam

Szenario beim Abgaben anzeigen

1. *Szenario „Einloggen“*
2. Der Nutzer wählt „Datennachweise“ im Hauptmenü
3. Es wird eine Liste aller getätigten Datennachweise angezeigt
4. Der Nutzer wählt konkrete Abgabe aus um Details zu erhalten

Szenario beim Abgaben nachweisen

1. *Szenario „Einloggen“*
2. Der Nutzer wählt im Hauptmenü „Datennachweise“
3. Der Nutzer wählt „Dateinachweis anlegen“
4. Ein betriebssystem-/browserspezifisches Fenster zum Öffnen einer Datei erscheint
5. Der Nutzer wählt eine lokale Datei aus
6. Es wird lokal der Hashwert der Datei gebildet
7. Es wird lokal eine Blockchain-Adresse aus dem Hash gebildet
8. Popup „Nachweis jetzt festhalten“
9. Der Nutzer bestätigt
10. Es wird eine Transaktion ausgeführt, die vom Account an die Hash-Adresse geht
11. Es wird die Adresse als Text und QR-Code angezeigt
12. Der Nutzer kopiert sich die Adresse oder den QR-Code und fügt sie an seine Abgabe an

Szenario beim Validität prüfen

1. Der Nutzer erhält eine Datei und Adresse des Nachweises
2. *Szenario „Einloggen“*
3. Der Nutzer wählt im Hauptmenü „Datennachweis prüfen“
4. Popup zur Webcam/Kamera-Nutzung
    a. Nutzer bestätigt
        i. Feld mit Eingabe zeigt Webcam/Kamera-Input
        ii. Nutzer zeigt Abgabe-Adresse als QR-Code
        iii. QR-Code wird in reguläre Adresse gewandelt
        iv. Adresse wird zwischengespeichert
    b. Nutzer verweigert
        i. Nutzer gibt Adresse manuell ein und bestätigt
        ii. Adresse wird zwischengespeichert
5. Der Nutzer wählt „weiter“
6. Adresse wird überprüft und das Datum sowie die Account-Adresse zwischengespeichert
7. Ein betriebssystem-/browserspezifisches Fenster zum Öffnen einer Datei erscheint
8. Der Nutzer wählt die abgegebene Datei aus
9. Es wird lokal der Hashwert der Datei gebildet
10. Es wird lokal eine Blockchain-Adresse aus dem Hash gebildet
11. Die beiden Adressen werden miteinander verglichen
12. Ergebnis wird angezeigt
    a. Adressen identisch
        i. Popup „Die eingegebenen Daten stimmen überein und wurden von [Adresse] am [Datum/Zeit] abgegeben
    b. Adressen verschieden
        i. Popup „Die eingegebene Daten stimmt nicht überein. Gegebenenfalls wurde die Datei nachträglich geändert“

Interne und administrative Prozesse

Szenario bei der Lehranstalt ins System einbinden

1. Die Lehranstalt bezahlt Softwarelizenz und gibt Kapital für Netzwerkabsicherung
2. Die Lehranstalt erstellt Account mit Wallets
3. Educhain händigt dafür einer Wallet Anteil an EDUC aus
4. Die Lehranstalt setzt eine Node auf, um mit EDUC dem Konsensmechanismus beizutreten und zu unterstützen
5. Educhain gibt der Lehranstalt-Adresse einen Identitätsdatensatz zur Verifikation
6. Account-Adresse der Lehranstalt wird transparent veröffentlicht

Szenario beim Token-Automat anmelden

1. *Szenario „Zugriffsschlüssel anlegen“*
2. *Szenario „Zugriffsschlüssel speichern“*
3. Die Lehranstalt navigiert ins Hauptmenü eines Mobiltelefons
4. Die Lehranstalt wählt „Gerät einrichten“
5. Die Lehranstalt wählt „Token-Automat“ aus der Liste
6. Die Lehranstalt verbindet sich über NFC mit dem Gerät
7. Der Token-Automat verbindet sich mit dem Mobiltelefon
8. Die Lehranstalt wählt „weiter“
9. Die Lehranstalt gibt WLAN-Daten ein
10. Der Token-Automat verbindet sich mit dem WLAN
11. Die Lehranstalt wählt „weiter“
12. Die Lehranstalt gibt gespeicherten DWK bei der Einrichtung ein
13. Es wird eine Wallet aus den Schlüsseln importiert
14. Der DWK wird als aktiv gekennzeichnet
15. Die Lehranstalt wählt „Einrichtung abschließen“
16. Der Token-Automat deaktiviert die erneute Einrichtung
17. Educhain gibt dem DWK die Berechtigung, Operationen im EDUT SC auszuführen

Szenario beim Token-Automat abmelden

1. Educhain entzieht DWK die Berechtigung, Ein- und Auszahlungen auszuführen
2. *Szenario „Zugriffsschlüssel löschen“*
3. Die Lehranstalt entfernt den DWK aus ihrem Account

Szenario beim Türschloss abmelden

1. *Szenario „Zugriffsschlüssel löschen“*
2. Die Lehranstalt entfernt alle Ausweise aus dem Tür-Account

Szenario beim Türschloss anmelden

1. *Szenario „Account erstellen“*
2. Die Lehranstalt verlinkt alle Identitätsdatensätze, welche Zutritt erhalten dürfen
3. Die Lehranstalt markiert alle Identitätsdaten-Ränge als Ausweise
4. Die Lehranstalt navigiert ins Hauptmenü eines Mobiltelefons
5. Die Lehranstalt wählt „Gerät einrichten“
6. Die Lehranstalt wählt „Türschloss“ aus der Liste
7. Die Lehranstalt verbindet sich über NFC mit dem Gerät
8. Der Token-Automat verbindet sich mit dem Mobiltelefon
9. Die Lehranstalt wählt „weiter“
10. Die Lehranstalt gibt WLAN-Daten ein
11. Der Token-Automat verbindet sich mit dem WLAN
12. Die Lehranstalt wählt „weiter“
13. Die Lehranstalt trägt die Adresse des Zugriffsrechte-Accounts ein
14. Die Lehranstalt wählt „Einrichtung abschließen“
15. Das Türschloss deaktiviert die erneute Einrichtung
    Das Türschloss prüft jedes Mal, ob geprüfte Identität eins der angegebenen Rechte besitzt

Szenario beim Spind anmelden

1. *Szenario „Zugriffsschlüssel anlegen“*
2. *Szenario „Zugriffsschlüssel speichern“*
3. Die Lehranstalt navigiert ins Hauptmenü eines Mobiltelefons
4. Die Lehranstalt wählt „Gerät einrichten“
5. Die Lehranstalt wählt „Spind“ aus der Liste
6. Die Lehranstalt verbindet sich über NFC mit dem Gerät
7. Der Token-Automat verbindet sich mit dem Mobiltelefon
8. Die Lehranstalt wählt „weiter“
9. Die Lehranstalt gibt WLAN-Daten ein
10. Der Token-Automat verbindet sich mit dem WLAN
11. Die Lehranstalt wählt „weiter“
12. Die Lehranstalt gibt gespeicherten DWK bei der Einrichtung ein
13. Es wird eine Wallet aus den Schlüsseln importiert
14. Der DWK wird als aktiv gekennzeichnet
15. Die Lehranstalt wählt „Einrichtung abschließen“
16. Der Token-Automat deaktiviert die erneute Einrichtung

Szenario beim Spind abmelden

17. *Szenario „Zugriffsschlüssel löschen“*
18. Die Lehranstalt entfernt den DWK aus ihrem Account

Anlage 2: Clone Factory Pattern

```
pragma solidity ^0.4.26;

/*
The MIT License (MIT)
Copyright (c) 2018 Murray Software, LLC.
Permission is hereby granted, free of charge, to any person obtaining
a copy of this software and associated documentation files (the
"Software"), to deal in the Software without restriction, including
without limitation the rights to use, copy, modify, merge, publish,
distribute, sublicense, and/or sell copies of the Software, and to
permit persons to whom the Software is furnished to do so, subject to
the following conditions:
The above copyright notice and this permission notice shall be included
in all copies or substantial portions of the Software.
THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS
OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF
MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT.
IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY
CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT,
TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE
SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.
*/
//solhint-disable max-line-length
//solhint-disable no-inline-assembly

/* Contract which implements the Clone Factory Pattern
   Code from https://github.com/optionality/clone-factory/
   blob/master/contracts/CloneFactory.sol

*/
contract CloneFactory {

  function createClone(address target) internal returns (address result) {
    bytes20 targetBytes = bytes20(target);
    assembly {
      let clone := mload(0x40)
      mstore(clone, 0x3d602d80600a3d3981f3363d3d373d3d3d363d730000000000000000000000000000000000000000)
      mstore(add(clone, 0x14), targetBytes)
      mstore(add(clone, 0x28), 0x5af43d82803e903d91602b57fd5bf3000000000000000000000000000000000000000000)
      result := create(0, clone, 0x37)
    }
  }

  function isClone(address target, address query) internal view returns (bool result) {
    bytes20 targetBytes = bytes20(target);
    assembly {
      let clone := mload(0x40)
      mstore(clone, 0x363d3d373d3d3d363d7300000000000000000000000000000000000000000000000000000000)
      mstore(add(clone, 0xa), targetBytes)
      mstore(add(clone, 0x1e), 0x5af43d82803e903d91602b57fd5bf3000000000000000000000000000000000000000000)

      let other := add(clone, 0x40)
      extcodecopy(query, other, 0, 0x2d)
      result := and(
        eq(mload(clone), mload(other)),
        eq(mload(add(clone, 0xd)), mload(add(other, 0xd)))
      )
    }
  }
}
```

Clone Factory Pattern Code [2]

Quellenangaben

[1] Haji, W. (o. D.). Learn Solidity: The Factory Pattern - Better Programming. Medium. Abgerufen am 9. Juli 2021, von https://betterprogramming.pub/learn-solidity-the-factory-pattern-75d11c3e7d29

[2] optionality/clone-factory. (o. D.). GitHub. Abgerufen am 9. Juli 2021, von https://github.com/optionality/clone-factory/blob/master/contracts/CloneFactory.sol

[3] ESP32 Wi-Fi & Bluetooth MCU I Espressif Systems. (o. D.). Espressif. Abgerufen am 9. Juli 2021, von https://www.espressif.com/en/products/socs/esp32

[4] nRF52832 - Nordic Semiconductor. (o. D.). Nordicsemi. Abgerufen am 9. Juli 2021, von https://www.nordicsemi.com/Products/nRF52832

[5] ID-IDEAL. (o. D.). ID-IDEAL BCCM. Abgerufen am 9. Juli 2021, von https://id-ideal.hs-mittweida.de/

[6] ECHT! - Fälschungssicheres Dokumentenverwaltungssystem. (o. D.). Blockchain-Schaufensterregion Mittweida -. Abgerufen am 9. Juli 2021, von https://blockchain-mittweida.com/echt/

[7] blockchainsllc/in3. (o. D.). GitHub. Abgerufen am 9. Juli 2021, von https://github.com/blockchainsllc/in3

[8] MetaMask. (o. D.-b). MetaMask. Abgerufen am 9. Juli 2021, von https://metamask.io/

[9] Remix - Ethereum IDE. (o. D.). Ethereum REMIX IDE. Abgerufen am 9. Juli 2021, von https://remix.ethereum.org/

[10] Truffle. (o. D.). Truffle Suite. Abgerufen am 9. Juli 2021, von https://www.trufflesuite.com/truffle

[11] Mocha - the fun, simple, flexible JavaScript test framework. (o. D.). MOCHA. Abgerufen am 9. Juli 2021, von https://mochajs.org/

[12] Configuring Jest · Jest. (o. D.). JEST JS. Abgerufen am 9. Juli 2021, von https://jestjs.io/docs/configuration

[13] React – A JavaScript library for building user interfaces. (o. D.). React. Abgerufen am 9. Juli 2021, von https://reactjs.org/

[14] React Native · Learn once, write anywhere. (o. D.). React Native. Abgerufen am 9. Juli 2021, von https://reactnative.dev/

[15] Git. (o. D.). Git SCM. Abgerufen am 9. Juli 2021, von https://git-scm.com/

Im Zusammenhang mit der Educhain Projektarbeit:

Webseite: *educhain-mw.de*

Prototyp: *educhain-mw.netlify.app*

Hilfsmittel und Werkzeuge

Microsoft Office Word	Textverarbeitung
Remix IDE	Entwicklungsumgebung Solidity
Visual Studio Code	Entwicklungsumgebung JavaScript
Netlifly	Hosting-Anbieter für Applikationen
GitHub	Versionierung und Sicherung von Software
MetaMask	Browser Ethereum-Wallet
MD5	Hashverfahren
React	Library für Benutzeroberflächen
web3.js	Library für Blockchain-Kommunikation

Selbstständigkeitserklärung

Hiermit erkläre ich, dass ich die vorliegende Arbeit selbstständig und nur unter Verwendung der angegebenen Literatur und Hilfsmittel angefertigt habe. Stellen, die wörtlich oder sinngemäß aus Quellen entnommen wurden, sind als solche kenntlich gemacht. Diese Arbeit wurde in gleicher oder ähnlicher Form noch keiner anderen Prüfungsbehörde vorgelegt.

Mittweida, 09.07.2021

Felix Hildebrandt