Dokumentation zum Blockchain Projekt: BlockCertify

Dokumentation zum

Blockchain Projekt

Duale Hochschule Baden-Württemberg Stuttgart

von Gruppe 4:

Tobias Böck

Daniel Widmayer

Theo Remmert

Jacob Haase

Dezember 2020

Matrikelnummern	9087213, 6942414
	9087213, 6942414
Kurs	Blockchain
Gutachter	Marvin König
	Michael Wehrstein

Sperrvermerk

Die vorliegende Dokumentation mit dem Titel *Dokumentation zum Blockchain Projekt: BlockCertify* ist mit einem Sperrvermerk versehen und wird ausschließlich zu Prüfungszwecken am Studiengang der Dualen Hochschule Baden-Württemberg Abgabeort vorgelegt. Jede Einsichtnahme und Veröffentlichung – auch von Teilen der Arbeit – bedarf der vorherigen Zustimmung durch die Firma GmbH.

Inhaltsverzeichnis

1 Einleitung

Tobi

2 Problemstellung

Tobi

3 Konzept

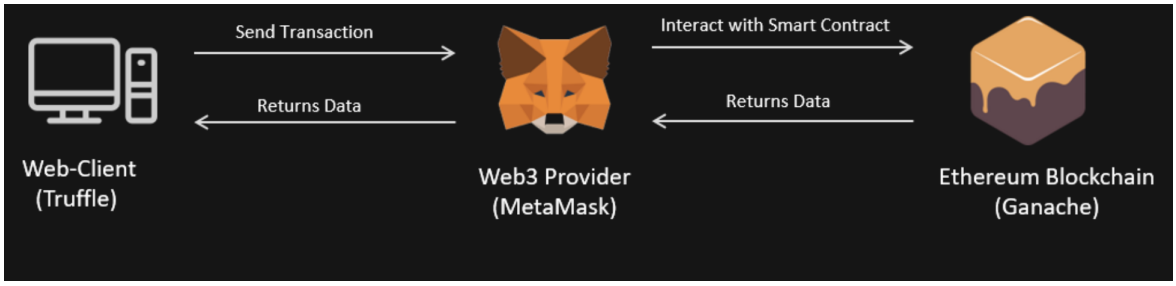
3.1 Vorteile von Blockchain

Zur Lösung des Problems, bietet sich die Blockchain Technologie an. Die Technologie ermöglicht das Abspeichern von Informationen, die danach nicht manipuliert werden dürfen. Zusätzlich werden Informationen dezentral und unabhängig von dritten Parteien gespeichert. Für die Technologie wird ein dezentrales Netzwerk eingesetzt. Zusammen mit einem Konsens Mechanismus werden die Daten (Transaktionen) über viele verteilte Knoten im Netzwerk als identische Kopie gespeichert. Dadurch wird Manipulation eines einzelnen Teilnehmers verhindert und Vertrauen gewährleistet.

Für das Zertifizierungs Projekt werden diese Eigenschaften benötigt. Die Unveränderlichkeit ist wichtig, damit eine Zertifikat nach der Ausstellung nicht gelöscht, oder verändert wird. Das ist vorallem wichtig in Bezug auf Online Kurse von Universitäten, die nach mehreren Jahren die Teilnahme an einem Kurs nachweisen müssen. Darüber hinaus ist die Abspeicherung des Zertifikats über eine Blockchain eine allgemeine, von Online-Kursanbietern unabhängige Lösung, auf die vertraut werden kann.

3.2 Lösungsarchitektur

Die Umsetzung erfolgt nach dem Beispiel des DHBW Truffle Projekts (Marvin König, 2020). Die Anwendung wird vom Nutzer ausgeführt und sendet mit Hilfe von verschiedener Eingaben Transaktionen zur Blockchain. Daraufhin wird die Transaktion mit anderen Transaktionen in einem Block zusammengefasst, der auf mehreren Knoten des Netzwerks, und dadurch sicher und unveränderlich abgespeichert wird.



Die Anwendung wird dem Nutzer, bzw dem Web-Client über einen Server zur Verfügung gestellt. Für die Verbindung zur Blockchain benötigt der Nutzer das MetaMask Web-Brower Plugin. MetaMask ist ein Zwischenspieler, der die Befehle von der Nutzeroberfläche erhält und mit der Blockchain kommuniziert. D. h. bei einer Nutzereingabe wird MetaMask aufgerufen, das dann alle weiteren Schritte mit der Blockchain durchführt. Dadruch können Transaktionen in die Blockchain gespeichert, sowie ausgelesen werden. Die Rollen Aussteller und Teilnehmer werden von einer Benutzeroberfläche abgedeckt. D. h. dass der Aussteller und Teilnehmer über alle Funktionen verfügt. Außerdem ist die Anwendung offen für jeden, also permissionless, mit einer permissionless Blockchain. Um dennoch nachzuweisen, dass ein Zertifikate einen Wert hat, und nicht von irgendeinem Teilnehmer gefälscht wurde, muss die Adresse des Zertifikats überprüft werden. Falls diese Adresse von einem vertrauenswürdigen Aussteller stammt, z. B. die Adresse einer Universität, ist das Zertifikat wertfoll.

4 Umsetzung

4.1 Smart Contract

Zur Umsetung wurde die Ethereum Blockchain verwendet. Die Ethereum Blockchain bietet die Möglichkeit ein kleines Programm, sog. Smart Contract auf die Blockchain als eigene Transaktion zu beschreiben. Dazu wird die Solidity Programmiersprache verwendet.

Im Smart Contract werden die Datentypen von Kursen und von Zertifikaten, jeweils durch ein Struct definiert. Diese beihalten alle wichtigen Felder:

Zum Zertifikat wird eine imageId als String (z. B. "01öder "06") angegeben. Dieser String ein Bild (z. B. ïmages/01.png"), das in der Benutzeroberfläche zum Zertifikat angezeigt wird.

Der Smart Contract speichert die Daten anhand von Mappings in die Blockchain. Diese Mappings wurden so angelegt, das häufige anfragen schnell durchgeführt werden:

- Aus Ausstellersicht werden ausgestellte Zertifikate u. Kurse dargestellt
- Aus Teilnehmersicht werden erhaltente Zertifikate u. teilnehmende Kurse gargestellt

Daraus ergeben sich folgende Mappings:

```
// map issuer to certificate
mapping(address => Certifate[]) public issuerCertificates;
// map participant to certificate
mapping(address => Certifate[]) public participantCertificates;

// map issuer to course
mapping(address => uint[]) public issuerCourses;
// map participant to course
```

```
mapping(address => uint[]) public participantCourses;

// map course id to course
mapping(uint => Course) public courses;
```

Anzumerken ist, dass die Kurse in einem seperaten Mapping gespeichert werden und in anderen Mappings über eine Kurs Id referenziert werden. Wenn ein Kurs Inhaber Teilnehmer zum Kurs hinzufügt oder entfernt, muss dadurch nur ein Eintrag in letzten Mapping verändert werden. Es muss zusätzlich ein Eintrag aus dem Mapping, welches Teilnehmer zu Kursen zuordnet, verändert werden, um Anomalien zu vermeiden. Dieser und andere Abläufe werden in Funktionen des Smart Contracts beschrieben.

Jede Funktionsimplementation ist durch Kommentare beschrieben:

```
// Adds / issues a course with a title.
// The course issuer will be set to the address that calls this method
function addCourse (string memory _title) public {
  ...
}
...
```

4.2 Web-Anwendung

Die Web-Anwendung wird über einen Server mit Hilfe von NodeJS bereitgestellt. Die Benutzeroberfläche wurde mit HTML, CSS und dem Bootstrap Framework erstellt. Um MetaMask bei Nutzerinteraktionen aufzurufen, wurde Javascript mit der Truffle, der JQuery und der Web3.js API eingesetzt. Sobald in der Oberfläche z. B. ein Formular abgeschickt wird, wird Javascript über JQuery Events aufgerufen, woraufhin die Daten ausgelesen werden:

```
$('#submit').click(async function() {
        var title = $('#course-title-input').val();
        ...
        await App.addCourse(title);
        ...
 });
```

Über Truffle und Web3.js wird daraufhin eine Transaktion an MetaMask weitergeleitet:

```
function addCourse: async function (title) {
        return App.contracts.Certification.deployed()...
}
```

alle anderen Funktionen wurde dieses Schema ähnlich implementiert. Die Benutzeroberfläche sieht dann wie folgt aus:

4.3 Blockchain

Als Blockchain wird eine persönliche Ethereum Blockchain verwendet, die über die Ganache Umgebung emuliert wird. So können Transaktionen zu Testzwecken gebührenfrei und schnell durchgeführt werden. Falls die dezentrale Anwendung für die Öffentlichkeit zur Verfügung gestellt wird, ist weiterer Arbeitsaufwand nötwendig.

5 Fazit

Abschließend lässt sich festhalten, dass Blockchain für unseren Anwendungsfall eine sehr gute Platform bietet. Jegliche Zertifizierung findet öffentlich statt und es kann leicht nachvollzogen werden, von wem und aus welchem Grund ein Zertifikat erstellt wurde. Dazu kommt der platformbedingte Vorteil grundsätzlicher Unveränderlichkeit getätigter Transaktionen auf einer Blockchain. Durch die Nutzung von Smart Contracts ist die dargestellte Lösungsarchitektur zudem in der Lage, die durch unsere User Stories definierten Anforderungen alle umzusetzen. Verschiedene Kurse können vom Endanwender der Blockhain auf Knopfdruck erstellt und die Teilnehemerzertifikate verwaltet werden. Das Interface bietet dabei ebenso wie die Zertifizierung an sich eine hohe Übersichtlichkeit und Transparenz. Nachteile unseres Ansatzes finden sich hingegen im Zertifikats-Mapping der Smart Contracts. Zu den Basisfunktionalitäten des Programs gehörend, findet an dieser Stelle die Verknüpfung zwischen Transaktion und Zertifikat statt. In der Umsetzung war das Erstellen und Ausarbeiten der Mappings intensiv und zeitaufwendig. Zwar sind im konkreten Projekt Aspekte wie das User-Feedback und Error-Handling noch ausbaufähig und auch das Nutzen, sowie Entwickeln der Anwendung bedingt durch MetaMask teilweise kompliziert und zeitaufwendig, allerdings Überwiegen die gebotenen Vorteile im Große und Ganzen. Solidity bietet in Sachen Blockchain eine solide Basis für Smart Contract basierte Blockchain-Anwendungen. Somit war es möglich, auch in kurzer Zeit eine funktionierendes Programm, mitsamt interface und Administration mit den wichtigsten Anforderungen zu erstellen.

6 Ausblick

Ausgehend von der bestehenden Anwendung bietet sich in mehrern Punkten die Möglichkeit des Ausbaus oder der Optimierung. Die hier genannten Optionen dienen primär der Integration in andere öffentliche Netzwerke und Anwendungsgebiete. Durch die Verlinkung auf Lernplatformen beispielsweise, kann nicht nur die Verbreitung, sondern auch die Nutzung unserer Blockchain gesteigert werden. Neben der erhöhten Reichweite in Nutzerkreisen, hat dies zudem den Vorteil transparenter und digitaler Zertifizierungen in allerlei Anwendungsgebieten, wie öffentliche Kurse oder interne Zertifizierungsprogramme. Um die Blockchain gegen Missbrauch, insbesondere Fake-Zertifizierungen zu wappnen, wird zudem noch eine CA[1] benötigt, um Zertifizierern eine Vertrauensgrundlage zu geben. Andernfalls wäre eine solche nur bei persönlichem Kennen und dem Austausch einzelner Blockchain Teilnehmer gewährleistet. Abschließend gibt es die Möglichkeit, die bisher lokal laufende Blockchain öffentlich zu integrieren. Nebst besserer Verfügbarkeit sind dadurch auch weiterführende Verwendung in Cloud-Infrastrukturen und Web-Services möglich.

[1] Certificate Authority

Literaturverzeichnis

Marvin König, N. V. (2020), 'Dhbw truffle project'. https://github.com/mkqavi/dhbw-truffle-project.