Aufgabenbeschreibung „Zimmervermietung“

Inhaltsverzeichnis

Gegebenes Szenario 2
Minimum 2
Erweiterung 2
Grafik 2
Mietzimmer 3
Angebot 3
Buchung 3
Umsetzung 4
Smart Contract mit Ethereum 4
Whisper Messages Protokoll für Ethereum 4
web3.js Client bei Ethereum Blockchain 4
REST-Schnittstelle bei Hyperledger Blockchain 4
User Account für UI 4
User Interface 4
Transaktion durchführen 4
Schnittstellen des Smart Contracts 5
Offene Punkte 5
Meilensteine 6
Nach Gruppen 6
Nach Umfang der Umsetzung 7
Organisatorisches 7
Kleingruppen 7
Veranstaltungstermine 7
Kurzvorträge 8
Seminararbeit 8
Präsentation 8

Gegebenes Szenario

Minimum

- 1 Zimmer: kann immer wieder vermietet werden
- 1 Mieter: mietet ein Zimmer
- 1 Vermieter: vermietet ein Zimmer

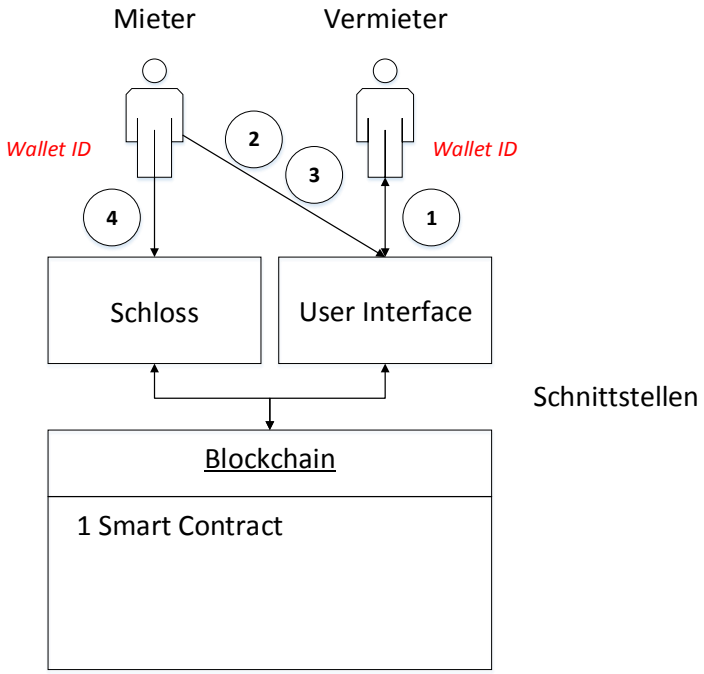
Anmerkungen:

- Ein Zimmer entspricht einer Tür.
- Es ist die Möglichkeit gegeben, während des gebuchten Zeitraums das Schloss mehrmals zu öffnen (Umsetzung mit Whisper).
- Das Schloss soll als Hardware vorhanden sein.

Erweiterung

- Mehrere Zimmer
- Mehrere Mieter
- Mehrere Vermieter

Grafik



1 Vermieter legt Angebot über User Interface in Blockchain an (= Transaktion)
2 Mieter schaut, welche Angebote es gibt (Only Read) und entscheidet sich ggf.
3 Mieter führt Buchung durch (Buchung kann nicht rückgängig gemacht werden und das Geld geht an den Vermieter)
4 Mieter kann Schloss mit Smartphone öffnen

Anmerkungen:

- Es existiert in der Blockchain ein Smart Contract, welcher in einer Datenbank verwaltet wird, um die Daten immer auf den aktuellsten Stand zu halten.
- In einem Smart Contract in der Blockchain wird immer das gleiche Booking durchgeführt.
- Wie sieht die Kommunikation zwischen User Interface und Blockchain aus?
- Ggf. im User Interface auswählen welche Technologie eingesetzt werden soll (Ethereum oder Hyperledger).
- *Spätere Erweiterung von Nummer 3:* Vermieter kann in Angebotserstellung einstellen, wie gut die Bewertung eines Mieters sein muss, damit der Smart Contract die Buchung akzeptiert.

Mietzimmer

Angebot

Variable	**Datentyp**
Angebot ID	Long
Tür ID	Long
Mietpreis (pro Nacht)	Long
Gültigkeitszeitraum Inserat (Zeit von /Zeit bis)	Date (Verwendung von UNIX)
Adresse des Objekts	Text
Name des Vermieters	String
Vermieter Wallet ID → *Public Key*	ca. 20 zufällige Zeichen
Beschreibung	String
evtl. Zimmerbild	

Buchung

Variable	**Datentyp**
Angebot ID	Long
Buchungszeitraum Zimmer (Check-in /Check-out)	Date (Verwendung von UNIX)
Mieter Wallet ID → *Public Key*	ca. 20 zufällige Zeichen

Anmerkungen:

- Tür ID ist Zieladresse für Whisper (quasi auch *Public Key*).
- Gültigkeitszeitraum Inserat wird durch den Vermieter festgelegt (Zeitraum kann auch leer gelassen werden und ist dadurch bis zur Löschung gültig).
- Buchungszeitraum Zimmer wird durch den Mieter festgelegt.
- Mieter Wallet ID und Vermieter Wallet ID sind „quasi“ die Benutzernamen.
- *Public* und *Private Key* werden automatisch generiert und werden für die Transaktionen benötigt.
- Mieter Wallet ID und Vermieter Wallet ID in JSON-File oder Datenbank (Serverseitig) speichern?

Umsetzung

Smart Contract mit Ethereum

- Ein oder mehrere Smart Contracts pro Angebot
- Verwaltung von Public Keys (Ethereum-Seite) und überlegen wie diese in Block geschrieben werden
- Umsetzung mit dem Framework Truffle + Ganache
- Smart Contract ID referenziert immer auf Ursprungsblock
- Welche Aktionen werden innerhalb des Smart Contracts ausgeführt?

Whisper Messages Protokoll für Ethereum

- Skizze bereits vorhanden
- Keine Transaktionen (und dadurch keine Kosten)
- Integriertes Schloss (Hardware) → Zugriff über Blockchain
- Smartphone sendet Token, welches zur Authentifizierung des Mieters durch den Vermieter dient
- Whisper signiert und hinterlegt einen Public Key

web3.js Client bei Ethereum Blockchain

- Umsetzung mit Angular und einem Node
- Anbindung der Daten in der Blockchain allgemein
- Kommunikation zwischen Blockchain und User Interface

REST-Schnittstelle bei Hyperledger Blockchain

- Anbindung der Daten in der Blockchain allgemein
- Kommunikation zwischen Blockchain und User Interface

User Account für UI

- Eigene Verwaltung von Accounts, Geld und Keys
- Plugin oder direkt über Ethereum bzw. Hyperledger Wallet

User Interface

- Schickt Nachrichten an die Tür
- Webinterface (grafische Benutzeroberfläche)
- Frontend Darstellung / wie buche ich überhaupt?

Transaktion durchführen

- Hyperledger: Private Key, Username (Wallet ID? Fingerprint) / E-Mail Adresse
- Ethereum: Key Phrase → Private Key, Wallet IDs / E-Mail Adresse
    - Schnittstelle zum Handy gibt es fertig als App

Schnittstellen des Smart Contracts

Per Konvention gibt es aktuell 3 Schnittstellen:

1. `insertOffer`: mit sämtlichen Parametern eines Angebotes aus dem Datenmodell
2. `deleteOffer`: mit der ID vom Offer
3. `rentAnOffer`: mit sämtlichen Parametern einer Buchung aus dem Datenmodell

Vorschlag Iterator z.B. für Buchungen, da diese nur von den betroffenen Ver-/Mietern eingesehen werden sollen:

- `getOffersLength() return uint;`
- `getOffer(uint index) return (uint price, string ownerName, .....);`
- `getBookingsLength() return uint;`
- `getBooking(uint index) return (uint checkIn, uint checkOut, .....);`

Für die Tür muss man wissen, ob der Mieter zu dieser Zeit die Erlaubnis hat in die gegebene TürID rein zu kommen:

`isAllowedAt`: mit der TürID, der AngebotsID und der aktuellen Zeit

Authentifizierungskonzept:

Benötigt wird ein eigener Contract für das Mappen einer Login-Adresse auf eine Ethereum-Adresse.

Offene Punkte

- Bei Whisper für Ethereum Verwendung von Public Key (P2P) *oder* Wallet ID
- Public und Private Key Verwaltung auf Diversen Geräten
- Vorhandenen Code für die Authentifizierung in unsere Applikation einbinden und evtl. für Hyperledger umschreiben bzw. erweitern

Meilensteine

Nach Gruppen

Gruppe	Meilensteine	Zuständig	Deadline
Hyperledger	Schnittstellen / Anbindung an: ○ Tür ○ Smart Contract ○ Web3 Client		
Tür mit Whisper	○ Hardware ○ Tür geht auf durch Whisper (mit Über-prüfung)		
Smart Contract	○ Contract abschließen ○ Contract Test Coverage überprüfen auf Vollständigkeit ○ Implementierung mit der Tür ○ Implementierung mit dem Frontend		
Web3 Client	○ Kommunikation mit der Tür via MQTT (Message Queue Telemetry Transport) ○ Einzelne Zugriffe auf die Hyperledger Blockchain (Angebot einstellen, buchen, anzeigen; User registrieren)		

Nach Umfang der Umsetzung

Tests der Umsetzung	Inhalt	Deadline
Minimum	1 Zimmer 1 Mieter 1 Vermieter	
Erweiterung	Mehrere Zimmer Mehrere Mieter Mehrere Vermieter	

Erledigt:

- Architektur festlegen (Skizze)

In Arbeit:

- Schloss und User Interface Implementierungen & Implementierung und Integration der Komponenten in der Blockchain
- Mieter und Vermieter mit Wallets einrichten

Organisatorisches

- Pflege von Einzelaufgaben und Einzelnen Projektteilen übernimmt jede Kleingruppe selbst.
- Besprechung immer Dienstag 30-60 Minuten. Danach Arbeiten in Kleingruppen.

Kleingruppen

- Hyperledger: Deniz, Frank
- Tür mit Whisper: Felix, Max, Peter
- Smart Contract: Anna, James, Rene
- Web3 Client: Alex, Daniel, Michi

Veranstaltungstermine

- Dienstag 29.05.2018
- Dienstag 05.06.2018
- Dienstag 12.06.2018
- Dienstag 19.06.2018
- Dienstag 26.06.2018

Kurzvorträge

Team-Mitglied	Thema	Datum
Anna		
Alex		
Daniel	OpenChain	24.04.2018
Deniz	Hyperledger	08.05.2018
Felix		
Frank	Hyperledger	08.05.2018
James	Solidity und Testing mit Ethereum	29.05.2018
Max		
Michi		
Peter		
Rene		

Seminararbeit

- Umfang maximal 5 Seiten pro Team-Mitglied (Text und Bilder)
- Kapitel mit Namen kennzeichnen
- Anna fügt alles zusammen und kümmert sich um das Layout etc.
- Abgabe (durch Anna) am Freitag den **13.07.2018** per E-Mail als PDF-Datei

Präsentation

- Dienstag **03.07.2018** und Dienstag **10.7.2018** *oder* Montag 02.07. *oder* Donnerstag 05.07. jeweils **ab 17 Uhr**
- Vortrag im Team; Dauer ca. 10 Minuten pro Team-Mitglied
- Umfang: Aspekte der Umsetzung/Vergleiche/andere Möglichkeiten/Probleme etc.