Blockchain-“Vier gewinnt” mit n-p2p Chat

Distributed Systems Advanced

Marco Keller, Martin Iten

Features

- Chat: Dezentralisierter Gruppenchat mit mehreren Peers
- Spiel: Vier-gewinnt mit 1 Gegenspieler

- Validierung der Spielzüge durch die Blockchain
- Mehrere Spiele parallel möglich

Technologien

NodeJS mit Express für den Signallingserver

Websocket-Verbindungen der Web-Clients zum Signallingserver

Javascript für das Frontend der Web-Clients

WebRTC für die einzelnen Peer-to-Peer-Verbindungen

Ethereum-Testnetz als Blockchain (Rinkeby-Testnetz)

WEB3-Library für die Verbindung der Web-Clients mit der Blockchain

Technologieentscheidungen

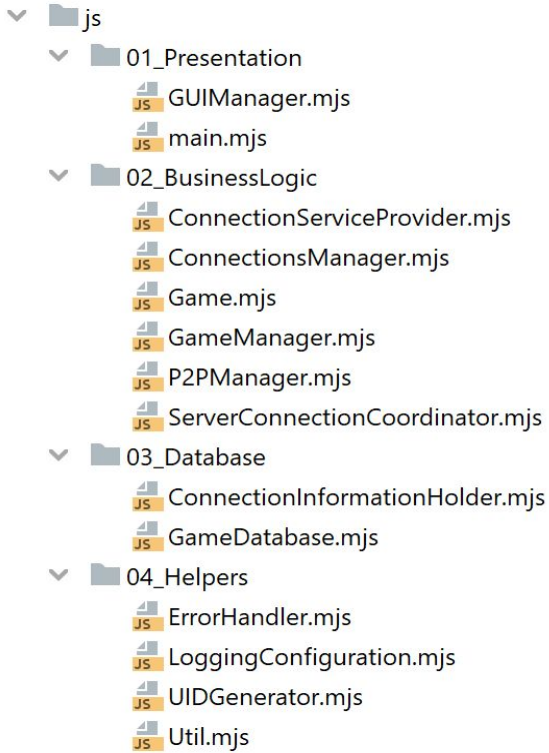
Wieso WebRTC?

- mittlerweile gibt es ziemlich solide Libraries für WebRTC und Firefox und Chrome unterstützen beide WebRTC
- Audio- und Videodatenkanäle können sehr einfach eingebaut werden

Wieso NodeJS und javascript?

- Studienarbeit auch in Node.js → Synergien
- Beispielprojekt von Thomas Bocek aus der Vorlesung auf Github ist auch mit NodeJS und javascript

Logische Architektur

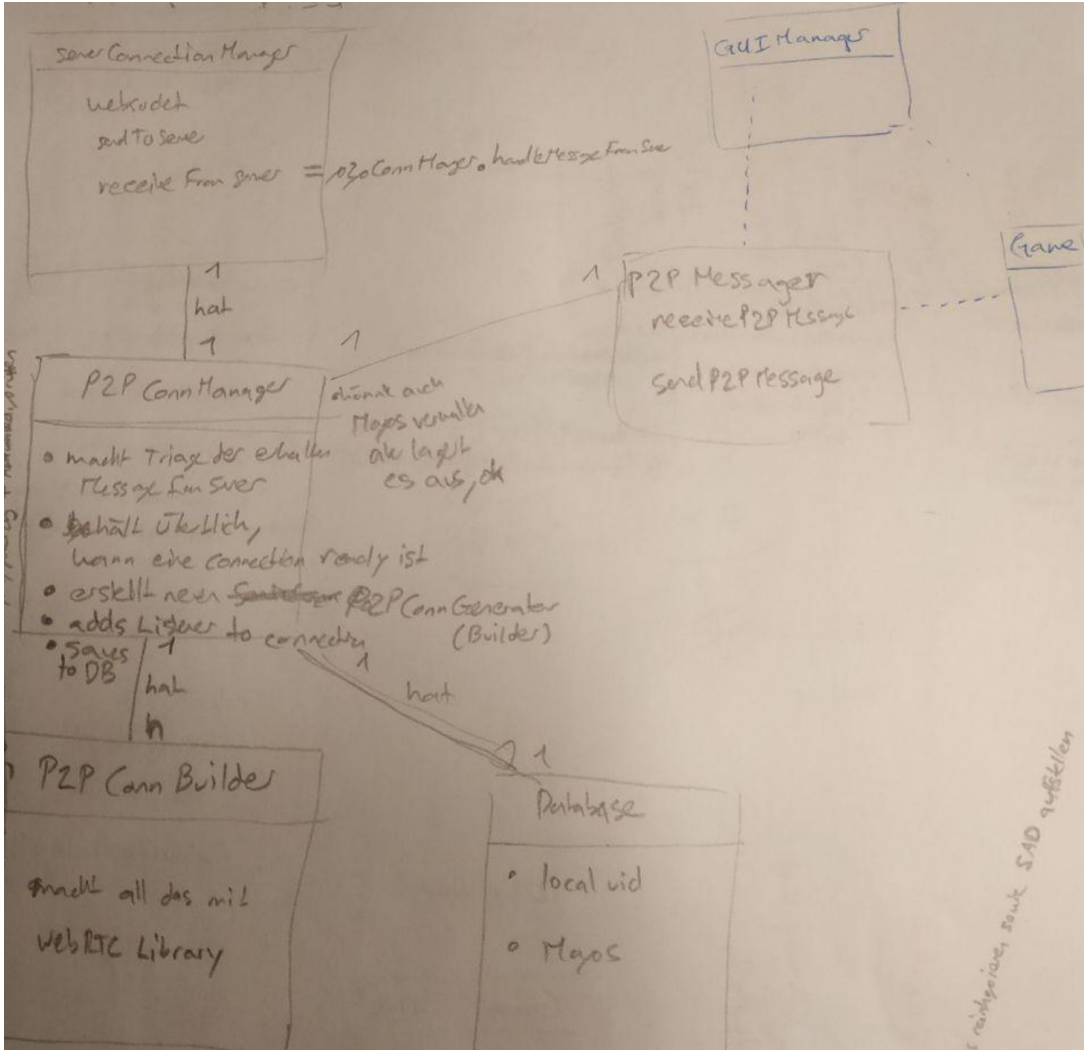


3 Schichten:

View / Presentation / GUI

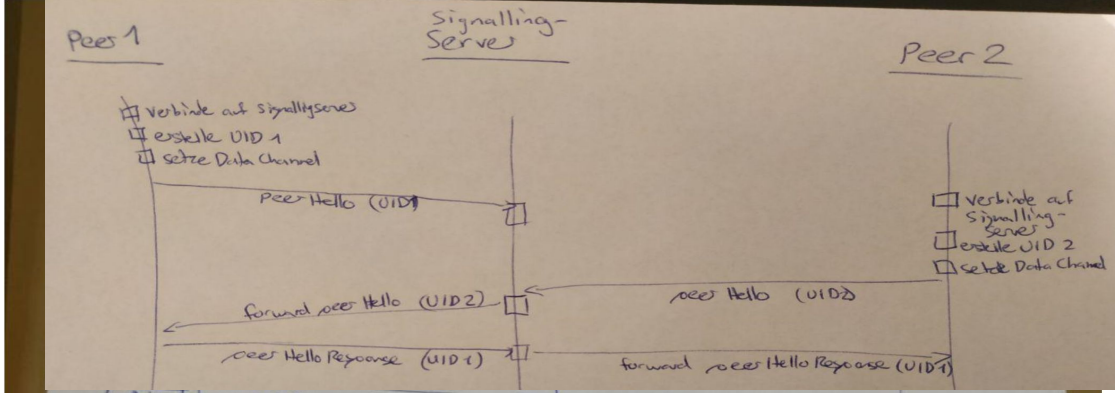
Controller / Business Logic

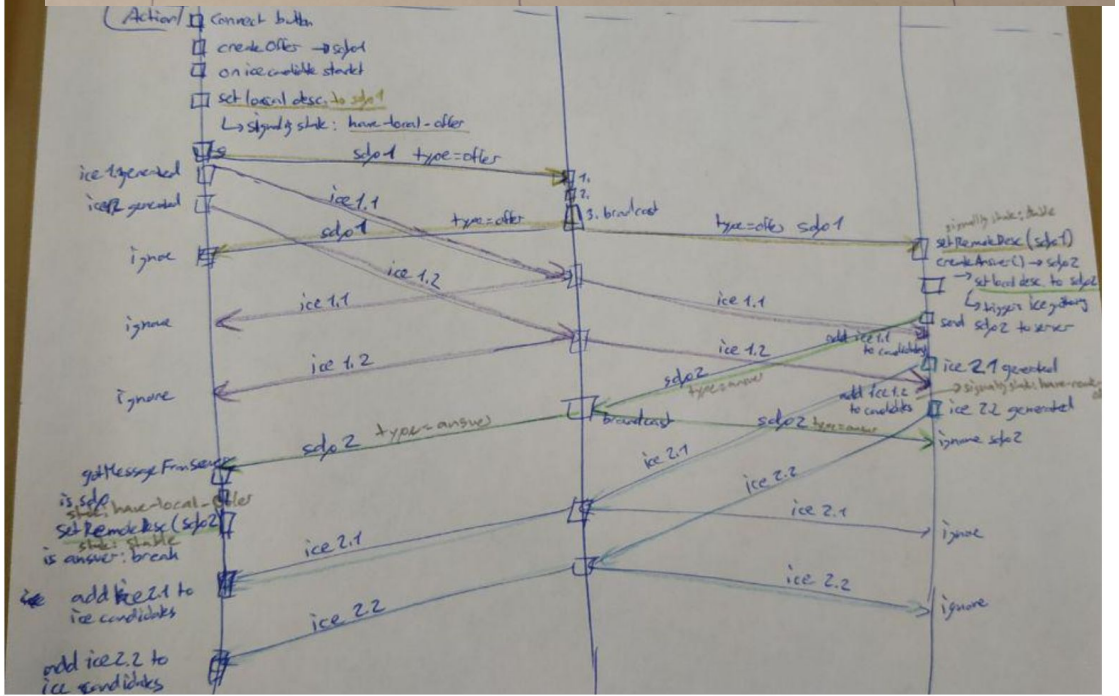
Model / In-memory database


ServerConnectionManager
sendToServe
receive From Server = p2pConnManager.handleMessageFromServer
GUIManager
1
hat
1
1
P2P Messager
receiveP2PMessage
sendP2PMessage
Game
P2P ConnManager
Mapos verwaltet alle Logik es aus, ok
macht Triage der erhaltenen Message from Server
erstellt neue P2P ConnGenerator (Builder)
adds Listener to connection
saves to DB
1
hat
n
1
hat
1
P2P Conn Builder
Database
local uid
Mapos
WebRTC Library

Architektur:

WebRTC-Verbindungsaufbau





Architektur: Spielzüge

Spielzüge werden lokal geprüft und zur globalen Validierung an die Blockchain gesendet

Spielzüge werden nicht nur an die Blockchain geschickt, sondern auch direkt via p2p-Datenkanal an den Gegenspieler

→ erlaubt bessere User Experience durch schnelleres Update

Spiellogik und Darstellung: Spielfeldgrösse und benötigte Reihenlänge kann mit der Änderung von 3 Variablen im Code angepasst werden

Schwierigkeiten

- WebRTC: eine **stabile** Verbindung zu mehr als 2 Peers erstellen war nicht einfach. SDP und ICE kommen teilweise nicht in der von uns erwarteten Reihenfolge an.
- Blockchain-Integration: Um die WEB3-Library einzubinden und Funktionen aus unserem Smart Contract aufzurufen benötigten wir recht viel Zeit (> 4 Stunden)
- Wir erhalten Blockchain-Events teilweise mehr als 1 Mal (in der Regel erhalten wir denselben Event 3 Mal) - offenbar ein WEB3-Library-Problem

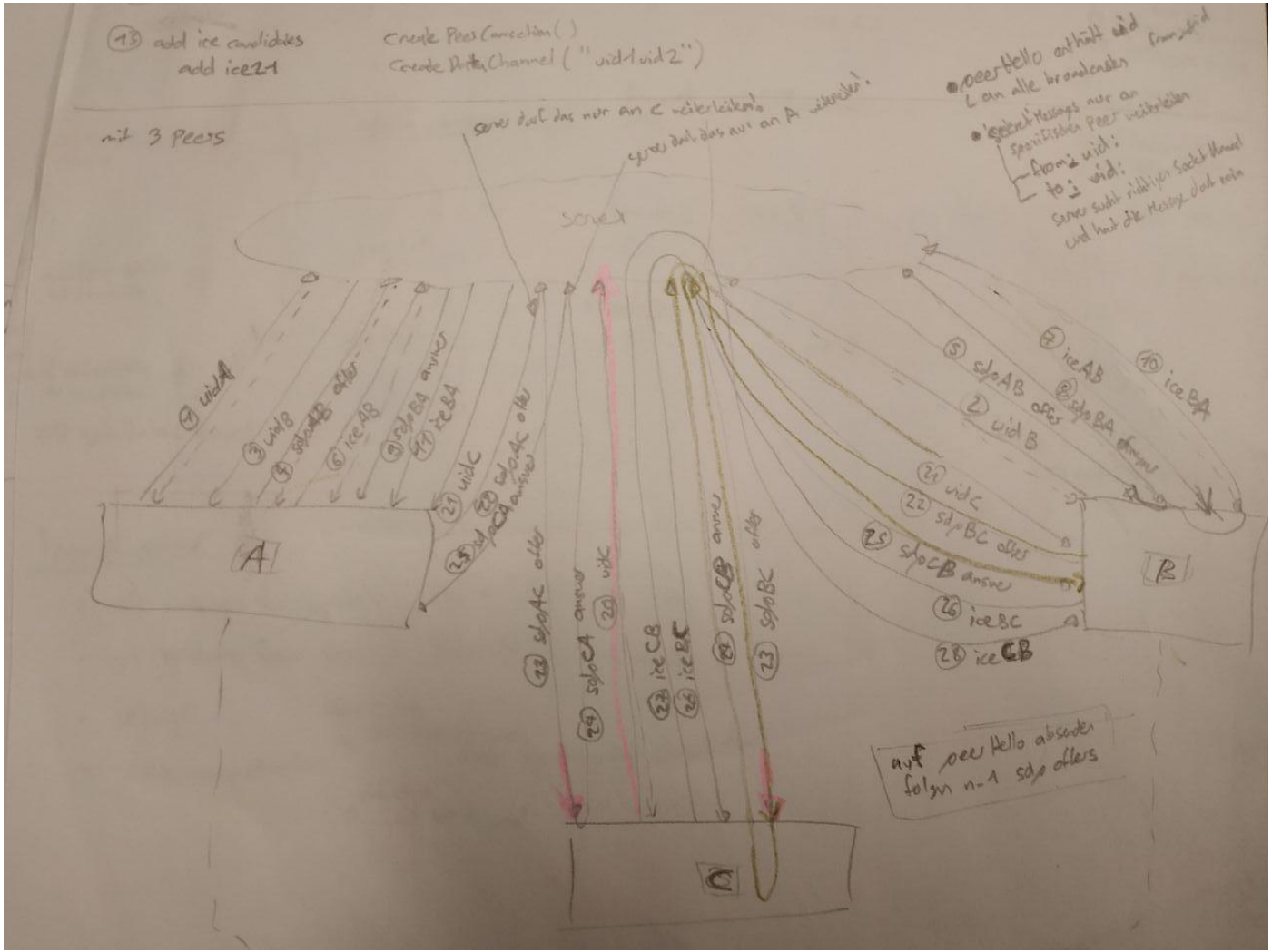
Learnings / Takeaways

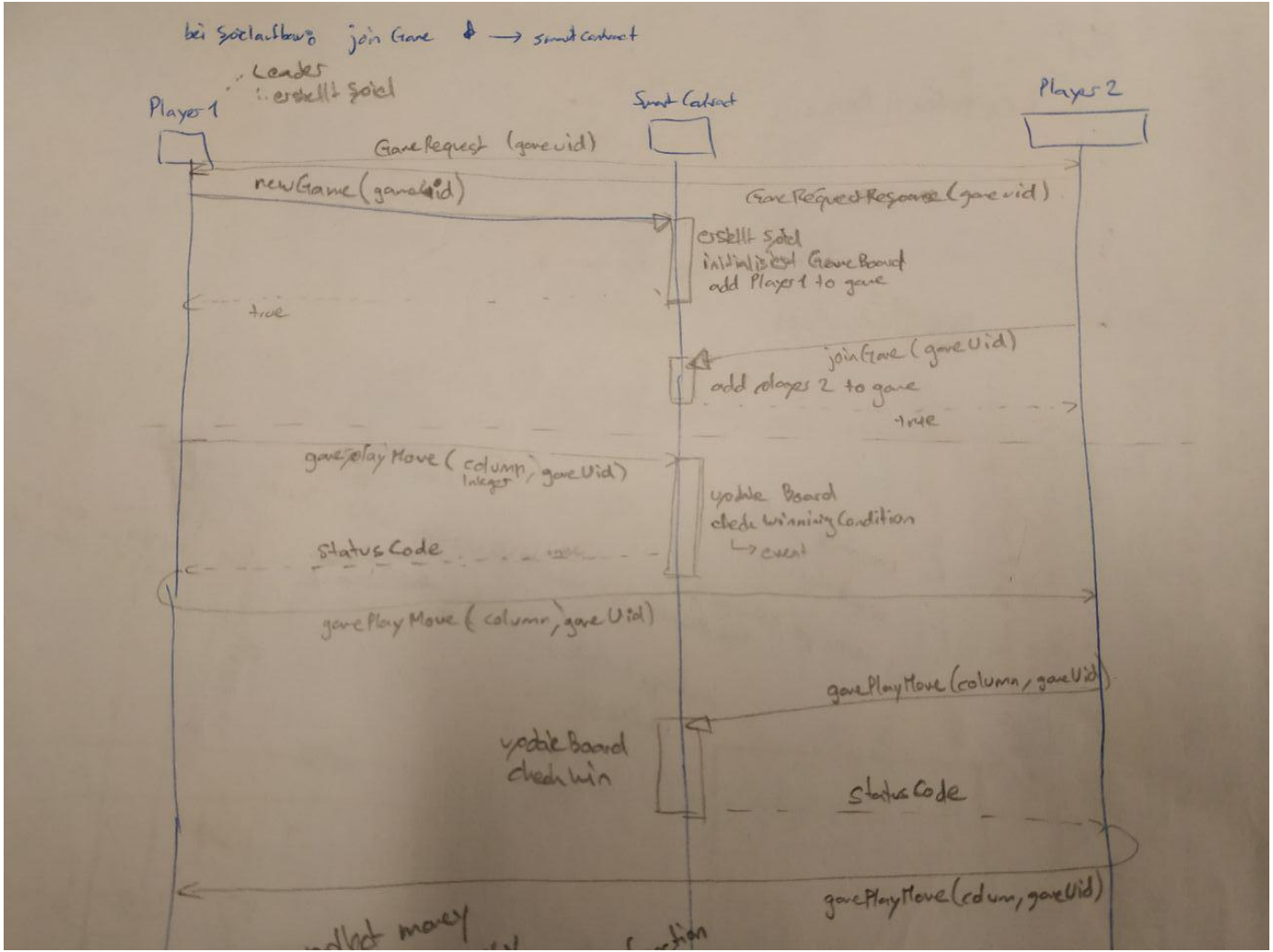
- Grenzen von Blockchain in der Praxis erfahren: Vier-gewinnt in der Blockchain zu spielen macht nicht so wirklich Spass: Der Spielfluss fehlt, vor allem zu Spielbeginn nervig
- Solidity (Ethereum) sieht zwar ähnlich aus wie javascript, aber im Detail unterscheidet es sich merklich: z. B. kein String-Array, keine Bit-Shifting-Operationen
- Weil komplexe Operationen (for-loops etc.) in der Blockchain teuer sind, ist man gezwungen nach Alternativen zu suchen → fördert das konzeptionelle Denken

Anhang

1.) Verbindungsaufbau WebRTC im Detail

2.) Spielzüge in der Blockchain


add ice candidates
add ice21
Create PeerConnection()
Create DataChannel ("uid1uid2")
mit 3 Peers
Server darf das nur an C weiterleiten
Server darf das nur an A weiterleiten
peerHello enthält uid
an alle broadcasten
gezielt Nachrichten nur an spezifischen Peer weiterleiten
from: uid:
to: uid:
Server sucht richtigen Socket-Channel und hat die Nachricht
Server
uidA
uidB
sdpAB offer
iceAB
sdpBA answer
iceBA
uidC
sdpAC offer
sdpCA answer
A
sdpAC offer
sdpCA answer
uidC
iceCB
iceBC
sdpCB answer
sdpBC offer
C
sdpAB offer
uidB
iceAB
sdpBA answer
iceBA
uidC
sdpBC offer
sdpCB answer
iceBC
iceCB
B
auf peerHello absenden
folgen n-1 sdp offers


bei Spielaufbau join Game → smart contract
Leader
: erstellt Spiel
Player 1
Smart Contract
Player 2
GameRequest (gameUid)
newGame (gameUid)
GameRequestResponse (gameUid)
erstellt Spiel
initialisiert GameBoard
add Player 1 to game
true
joinGame (gameUid)
add player 2 to game
true
gamePlayMove (column, gameUid)
updateBoard
checkWinningCondition
→ event
statusCode
gamePlayMove (column, gameUid)
gamePlayMove (column, gameUid)
updateBoard
checkWin
statusCode
gamePlayMove (column, gameUid)