DSA Chat

Benjamin Daniels, Zarko Dragojevic & Micha Schena

Technologien pro Aufgabe

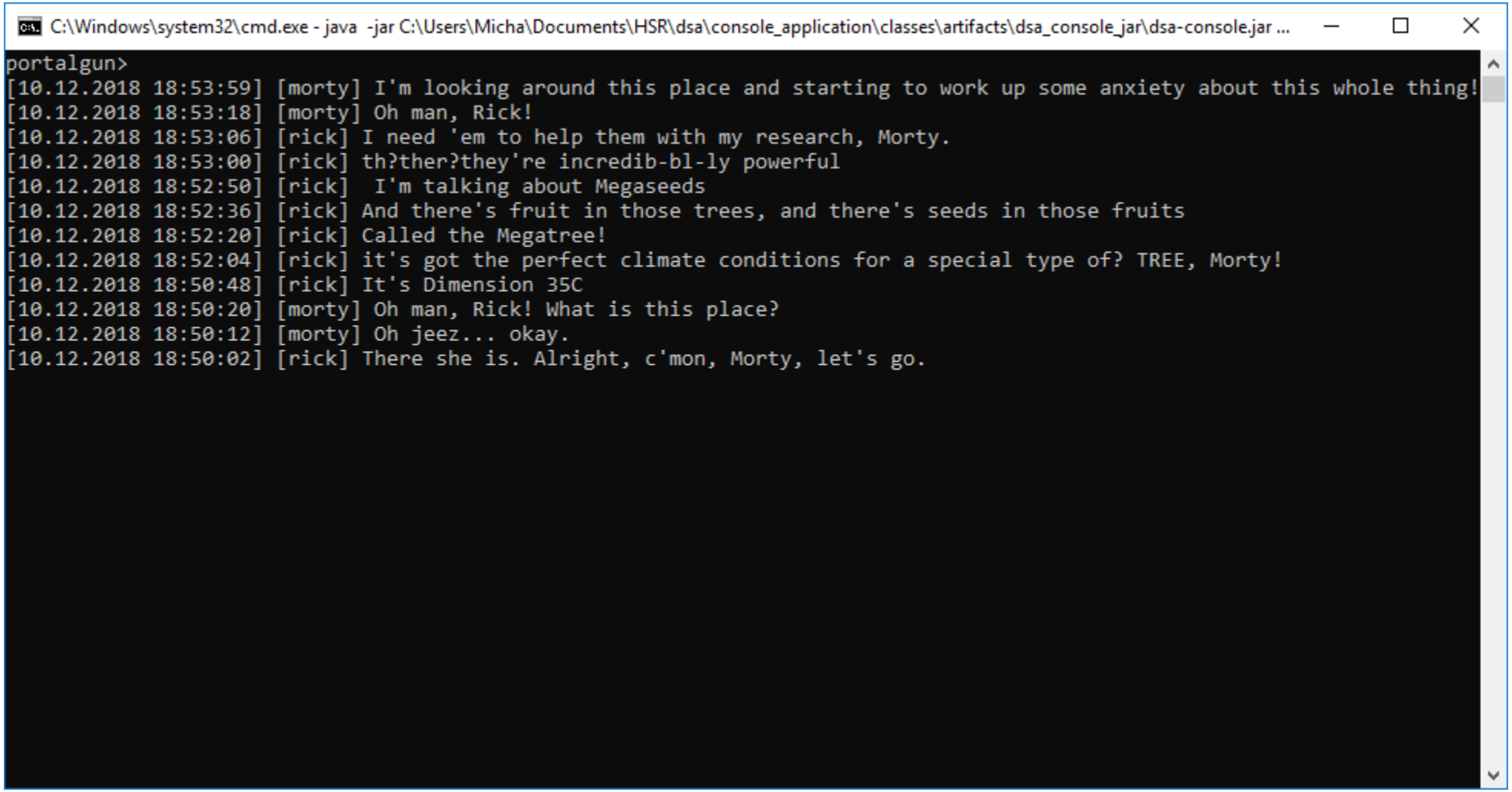
- Verteilte Chat-Applikation:

  TomP2P

- Eingeschriebener Brief:

  Smart Contract auf ETH + WEB3.js + Metamask Plugin

Zwischenstand



Architektur

- Backend:

  Spring Boot REST API + WebSockets

- Frontend:

  Angular mit Material Design + SockJS + StompJS + WEB3.js

Architektur TomP2P

- Bootstrap mit **peerId = Number160.createHash(name);**
- Userinfos (Name und Wallet) werden an Position **peerId** gespeichert:

peer.put(peerId).domainKey(USER_NAME_KEY).data(new Data(name))

- Userliste über **PeerBean**:

peer.peer().peerBean().peerMap().all()

- Messaging über **SendDirect:**

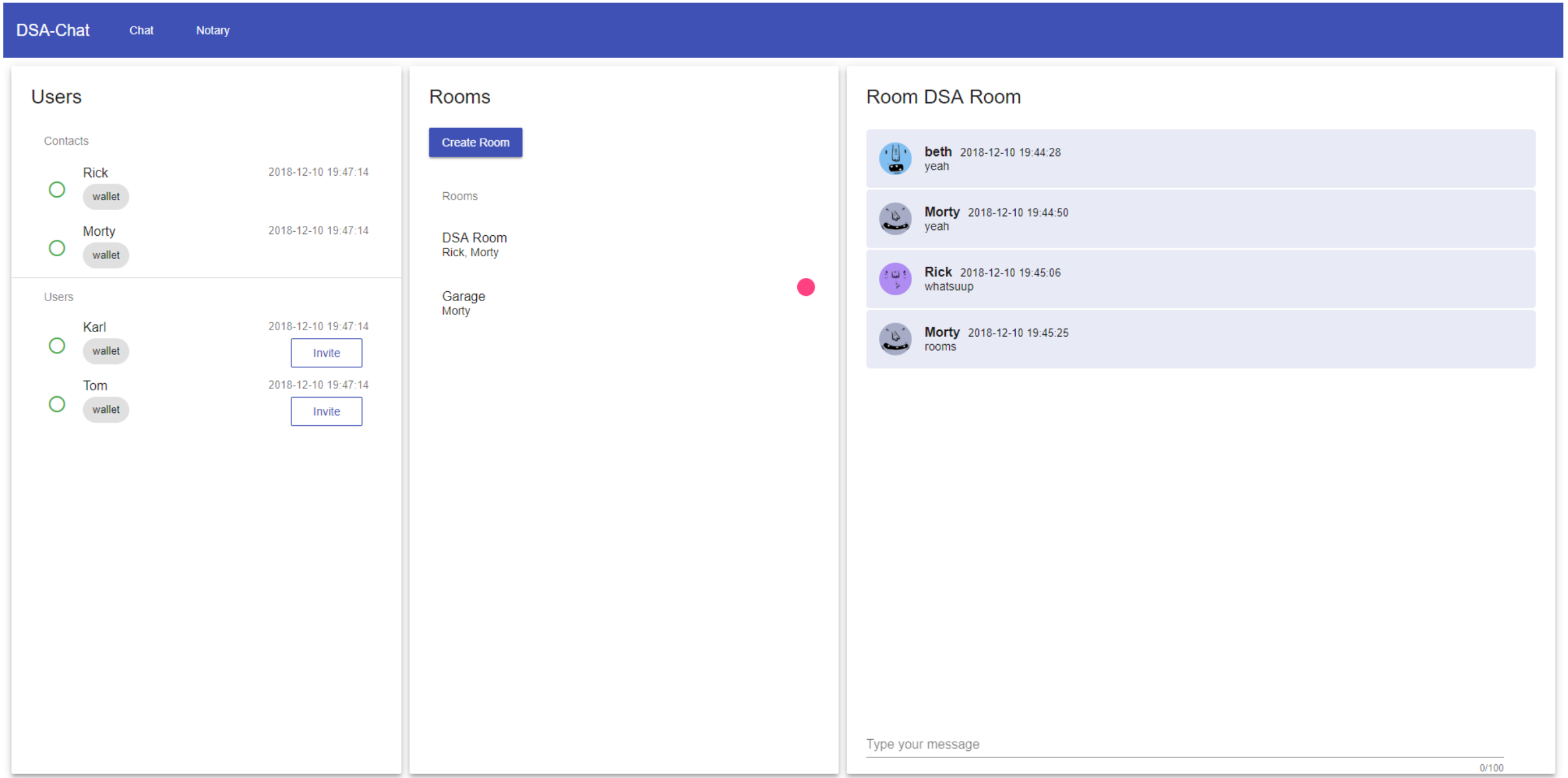
peer.peer().sendDirect(receiverAddress).object(message)

Architektur Ethereum Smart Contract

- WEB3.js
- Metamask
- Smart Contract mit Solidity auf Ethereum Blockchain

```
contract NotaryService {
    struct Letter {
        address sender;
        address receiver;
        string message;
        bool isRead;
        bool isAccepted;
        bool isValue;
    }

    mapping(bytes32 => Letter) letters;
    bytes32[] letterIds;

    function getLetterIds() public constant returns (bytes32[]) {
        return letterIds;
    }

    function getLetter(bytes32 id) public constant returns (address sender, address receiver, string message, bool isRead, b
        return (letters[id].sender, letters[id].receiver, letters[id].message, letters[id].isRead, letters[id].isAccepted);
    }

    function createLetter(bytes32 id, address receiver, string message) public {
        require(!letters[id].isValue);
        letters[id] = Letter(msg.sender, receiver, message, false, false, true);
        letterIds.push(id);
    }

    function acceptLetter(bytes32 id, bool isAccepted) public {
        require(letters[id].receiver == msg.sender && !letters[id].isRead);
        letters[id].isRead = true;
        letters[id].isAccepted = isAccepted;
    }
}
```

Demonstration



Fragen?