Kudos Token

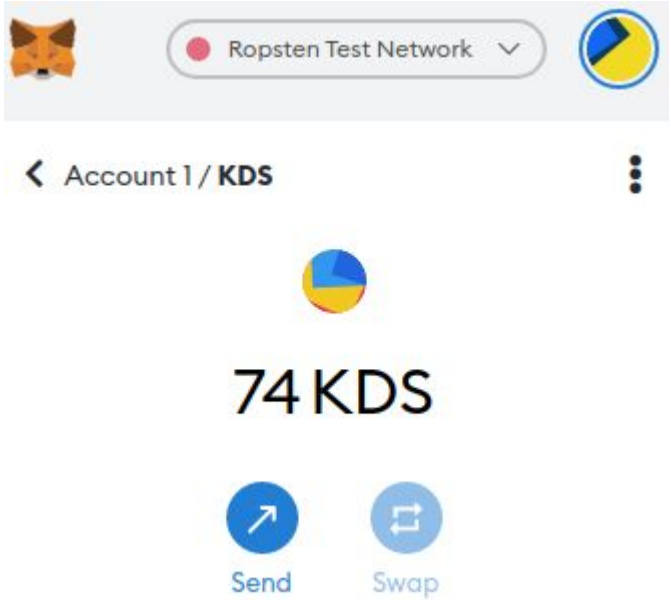
Yannick & Robin

Inhalt

- Übersicht
- Demo
- Technologien
- Architektur
- Implementation
  - Contract
  - UI
- Requirements
- Diskussion

Kudos (KDS)

- ERC20 Token
- Web-UI für Transaktionen



Demo

Technologien & Werkzeuge

Contract

- Remix IDE
- Solidity
- Ropsten Testnet

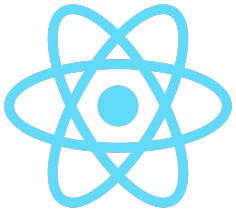
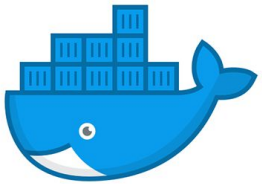
Web-UI

- React
- web3.js
- Etherscan API

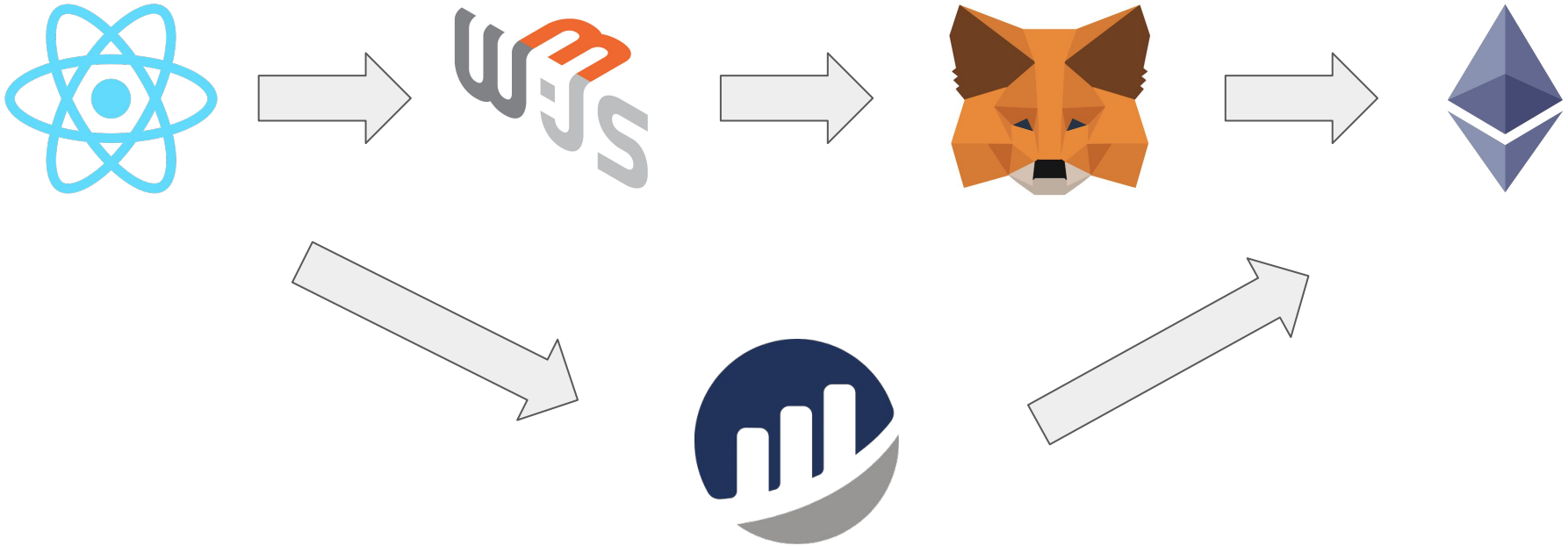
Deployment

- Ethereum Blockchain
- Docker
- Traefik




Architektur



Contract Implementation

ERC20 Interface

```
interface IERC20 {
    function name() external view returns (string);
    function symbol() external view returns (string);
    function decimals() external view returns (uint8);
    function totalSupply() external view returns (uint256);

    function balanceOf(address owner) external view returns (uint256 balance);
    function transfer(address to, uint256 value) external returns (bool success);
    function transferFrom(address from, address to, uint256 value) external returns (bool success);
    function approve(address spender, uint256 value) external returns (bool success);
    function allowance(address owner, address spender) external view returns (uint256 remaining);
```

Transfer Methode

```
function transfer(address to, uint256 tokens) public returns (bool success) {
    require(tokens <= balances[msg.sender]);

    balances[msg.sender] = balances[msg.sender].sub(tokens);
    balances[to] = balances[to].add(tokens);

    emit Transfer(msg.sender, to, tokens);
    return true;
}
```

UI Implementation

Initialisierung

```
const initializeWeb3 = () => {
  if (window.ethereum) {
    window.web3 = new Web3(window.ethereum);
    window.ethereum.enable();
  } else {
    alert("Please install MetaMask to use the Kudos (KDS) Wallet UI");
  }
};

const initializeContract = () => {
  window.kds = new window.web3.eth.Contract(kdsInterface, kdsContractAddress);
};
```

ERC20 Interface

```
}, {
  "constant": false,
  "inputs": [{"name": "to", "type": "address"}, {"name": "tokens", "type": "uint256"}],
  "name": "transfer",
  "outputs": [{"name": "success", "type": "bool"}],
  "payable": false,
  "stateMutability": "nonpayable",
  "type": "function"
}, {
```

Transfer Methode

```
const transfer = () => {
  window.kds.methods
    .transfer(toAddress, amount)
    .send({ from: window.ethereum.selectedAddress }, (err, res) => {
      if (err) {
        setTransactionStatus(
          <span style={{ color: "tomato" }}>Failed to send transaction</span>
        );
      } else {
        setTransactionStatus(
          <a target="_blank" rel="noreferrer" href={"https://ropsten.etherscan.io/tx/" + res}>
            Sent, view on Etherscan
          </a>
        )
      }
    })
};
```

Requirements

1. Full decentralization and P2P mechanisms or scalable mechanisms
   - ➢ Skalierbar mittels traefik
2. Graphisches (webbasiertes) Frontend
   - ➢ React App
3. Läuft auf einer öffentlichen Blockchain
   - ➢ Ethereum

Fazit

- Von 0 auf 1
  - Keine Erfahrung mit Solidity, web3.js, Ethereum oder React
- Viel gelernt
  - Solidity
  - React
- Einfache Ethereum-Tokens mit Solidity implementieren ist relativ angenehm

Diskussion