BLOCKAIN CHALLENGE TASK: SUPPLY CHAIN TRACKING

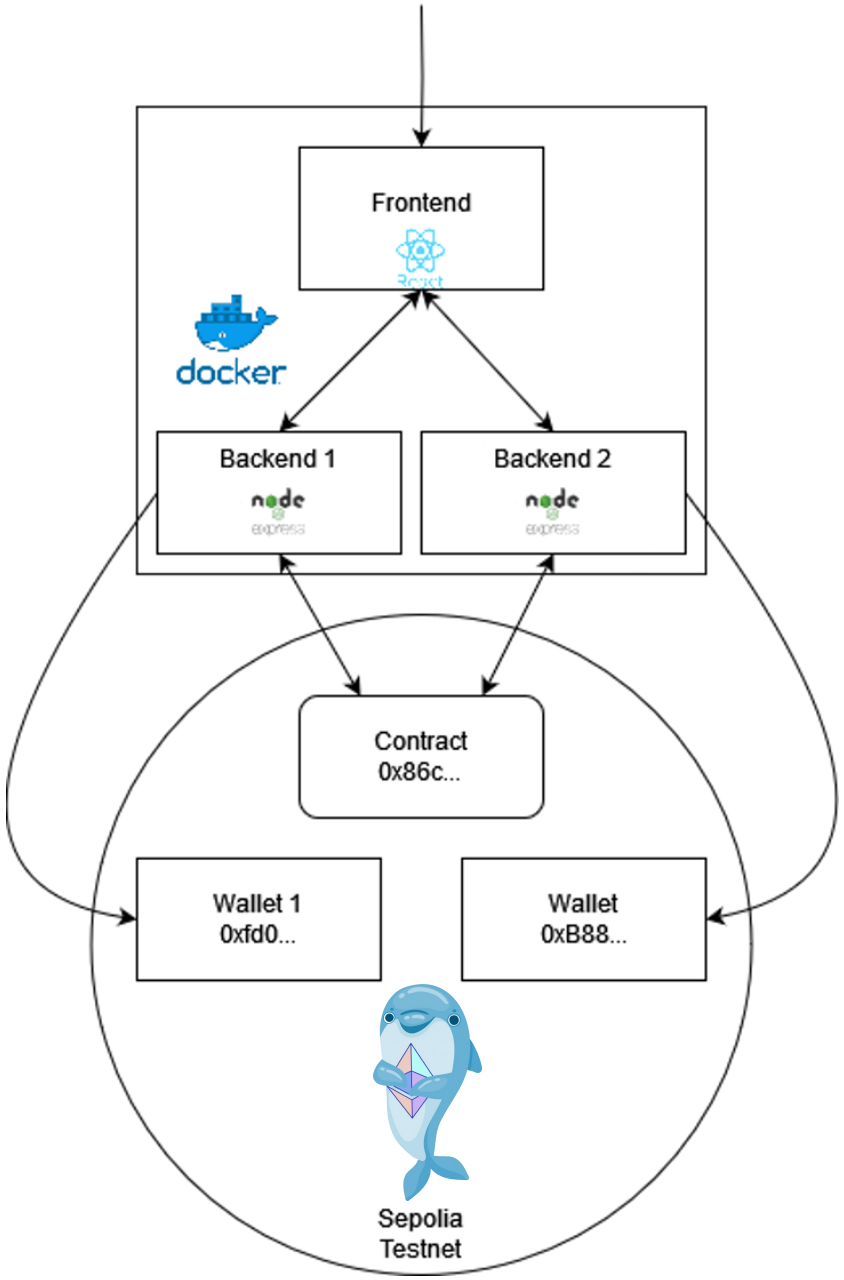
Leonie Däullary, Linus Flury, Kevin Kempf

Grundidee

Sendungen verfolgen mit Blockchain

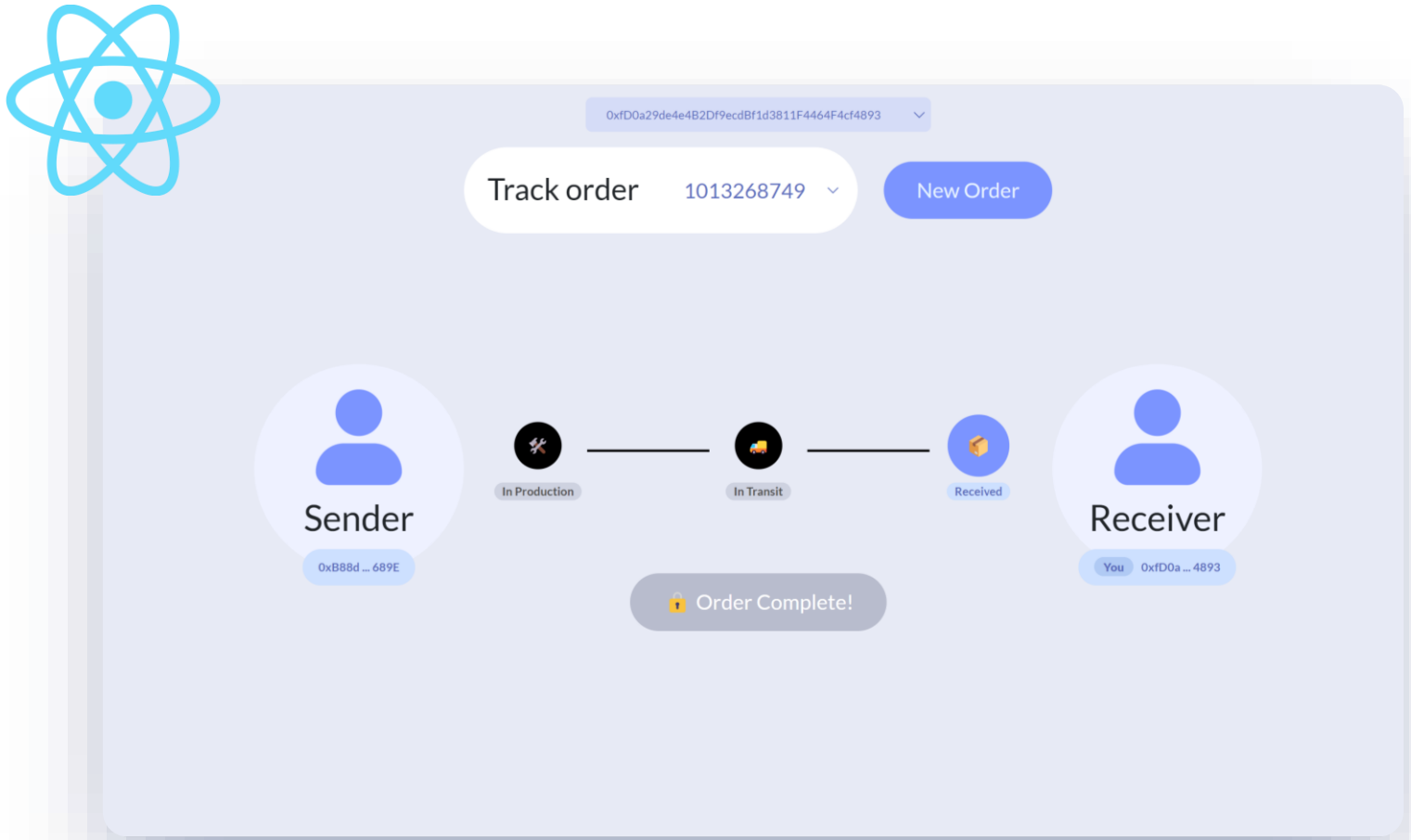
Sender updatet Status

Empfänger bestätigt Empfang



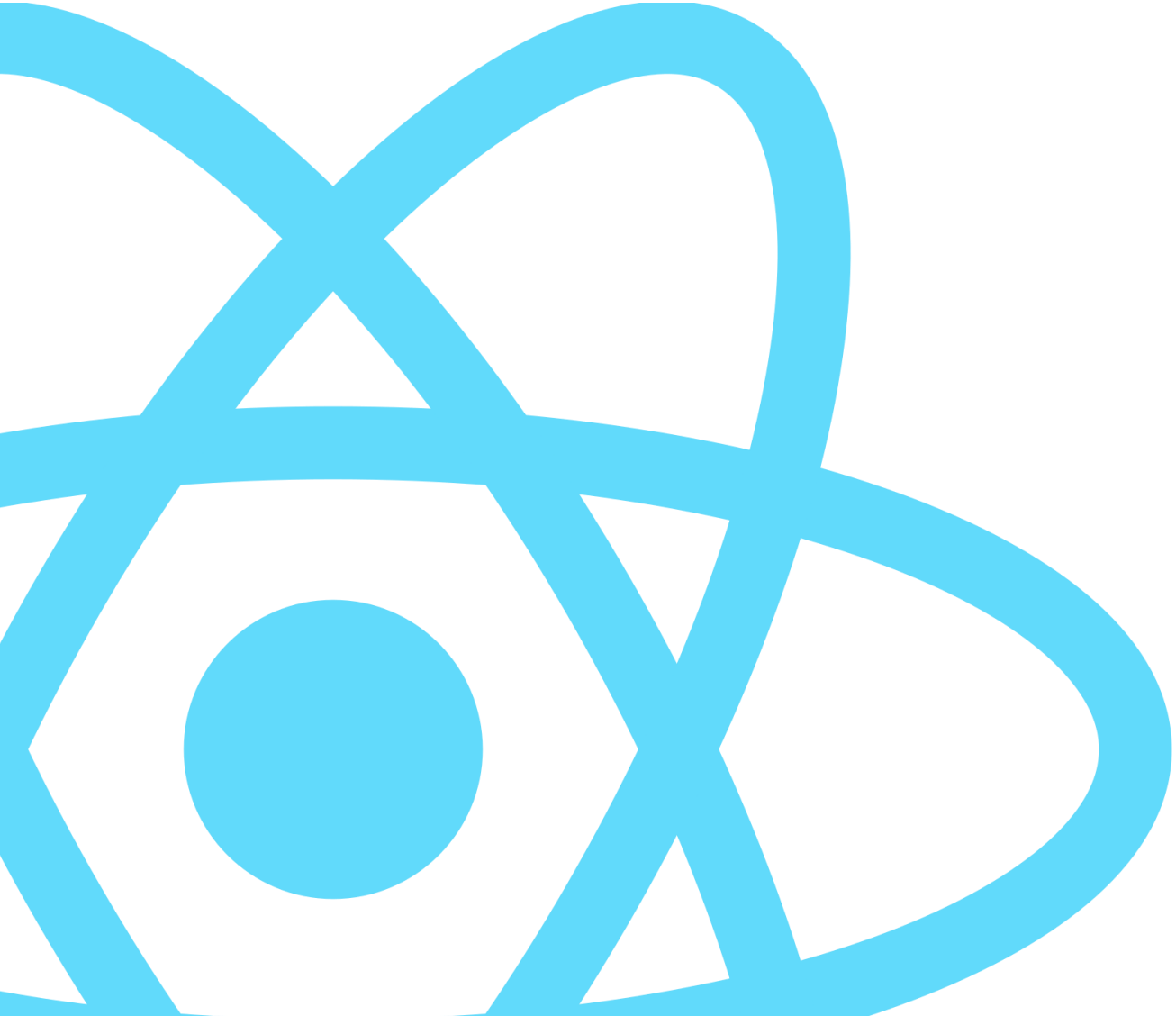
Setup

- Docker: Vereinheitlichung, Einfaches Instanziieren von Code-Identischen Backendinstanzen
- Zwei BE Instanzen um zwei Parteien darzustellen
- Kommunikation FE/BE: HTTP Requests
- Kommunikation BE/Sepolia: ether.js



Frontend

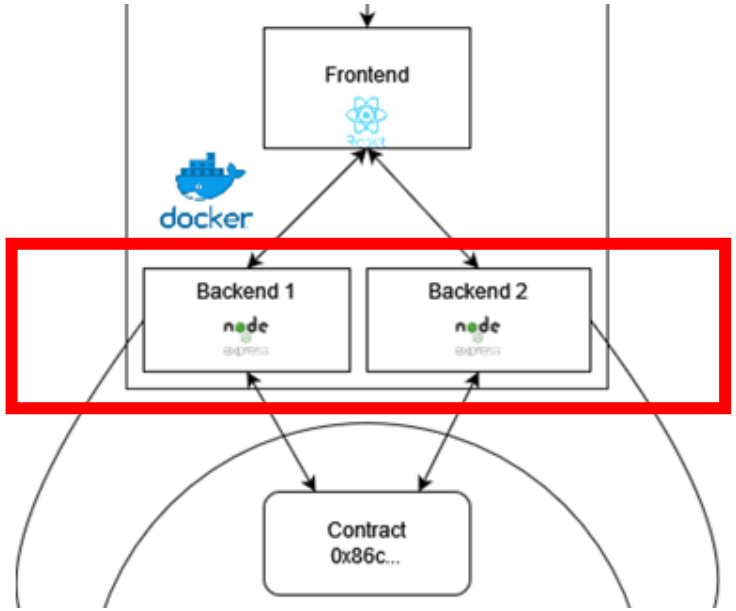
- React-basiert
- Zeigt den Status der Lieferung an
- Ermöglicht das Aktualisieren des Status
- Erlaubt das Erstellen neuer Bestellungen



Wieso React

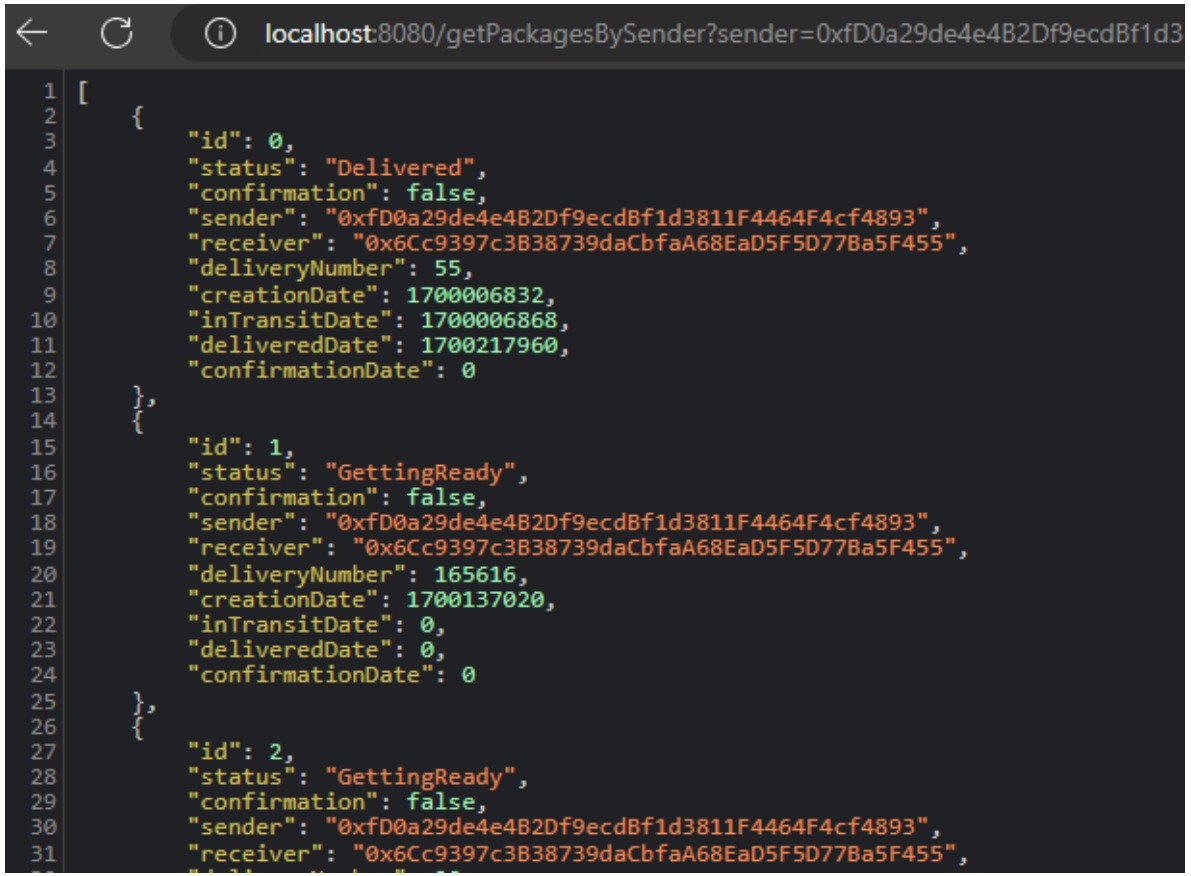
- Backend ist ebenfalls Node-basiert.
- Komponentenbasierte Architektur.
- Angular ist für große Projekte besser geeignet.
- Flache Lernkurve.
- Hat sich in früheren Projekten bewährt.

Backend



- NodeJS mit Express Framework,
- Einfache Abfrage über den Browser

Wieso Backend



- Das Backend verarbeitet alle Anfragen der Firma
- One Wallet
- Einfache Erweiterungs-möglichkeit (Login, Processing, Kontrolle über Transaction..)

Wieso NodeJS

- Die meisten Tutorials mit Javascript
- Einfach Interaktion dank Alchemy-SDK

Contract

- Create
- Read
- Update
- ~~Delete~~

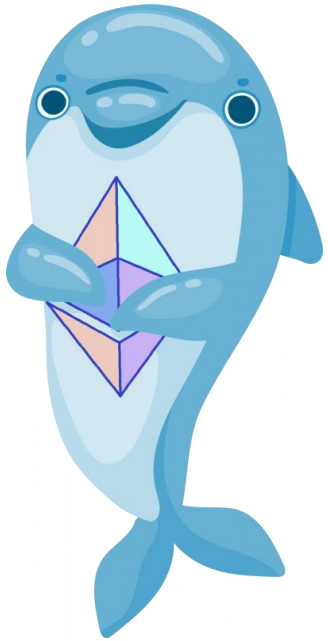
Update

Update

- Progressiv
- Status: Sender only (require)
- Bestätigung: Empänger only (require)

- Variante: Changelog pro Paket

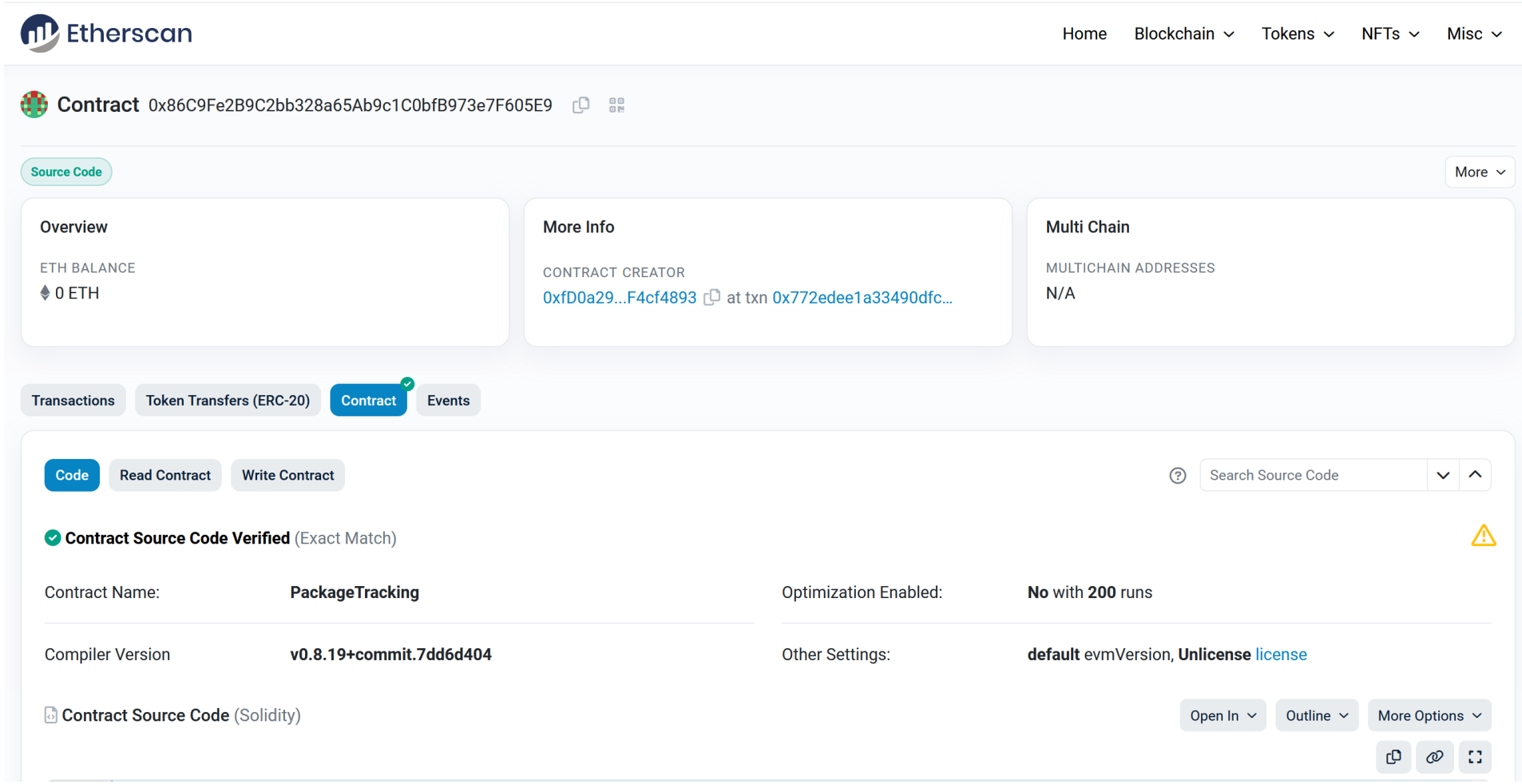
Solidity & Sepolia



- Smart-Contract geschrieben in Solidity
  - Behandelt in Vorlesung
  - Grosse Community
  - Einsteigerfreundlich
  - Alternativen: YUL, Vyper

- Sepolia: Ethereum Testnet
  - Vorgeschlagen durch Dozent
  - Frei zu verwenden
  - Alternative: Goerli, goETH aber wegen Scarcity nicht verwendet

Etherscan verified



https://sepolia.etherscan.io/address/0x86C9Fe2B9C2bb328a65Ab9c1C0bfB973e7F605E9#code

Kosten

Etherscan:

Latest 8 from a total of 8 transactions

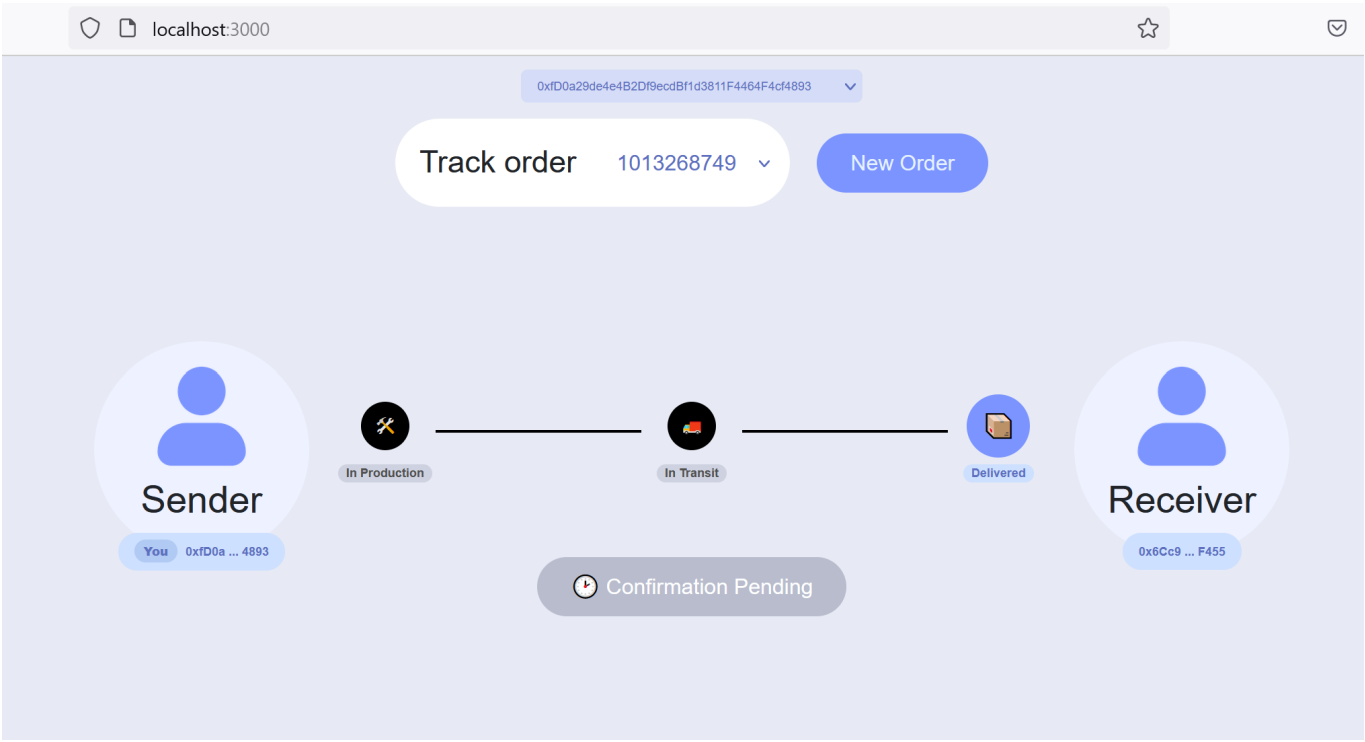
Transaction Hash	Method	Block	Age	From		To	Value	Txn Fee
0x3ce722633de0fe28b...	Update Packa...	4813963	5 mins ago	0xfD0a29...F4cf4893	IN	0x9b07c1...D900a143	0 ETH	0.00008337
0x8c57105034d102ba...	Create Package	4813962	6 mins ago	0xfD0a29...F4cf4893	IN	0x9b07c1...D900a143	0 ETH	0.00027087
0x03fc0d66a3781f428...	Update Packa...	4813956	7 mins ago	0xB88d77...1922689E	IN	0x9b07c1...D900a143	0 ETH	0.00008436
0x7f1bcd878e4dbc234...	Update Packa...	4813954	7 mins ago	0xB88d77...1922689E	IN	0x9b07c1...D900a143	0 ETH	0.00008362
0x5989ff167b5ddd79c...	Create Package	4813949	8 mins ago	0xB88d77...1922689E	IN	0x9b07c1...D900a143	0 ETH	0.00027042

Kosten:

- Neues Package: 0.00027 ETH, ~0.51 CHF
- Update Package: 0.000084 ETH, ~0.16 CHF

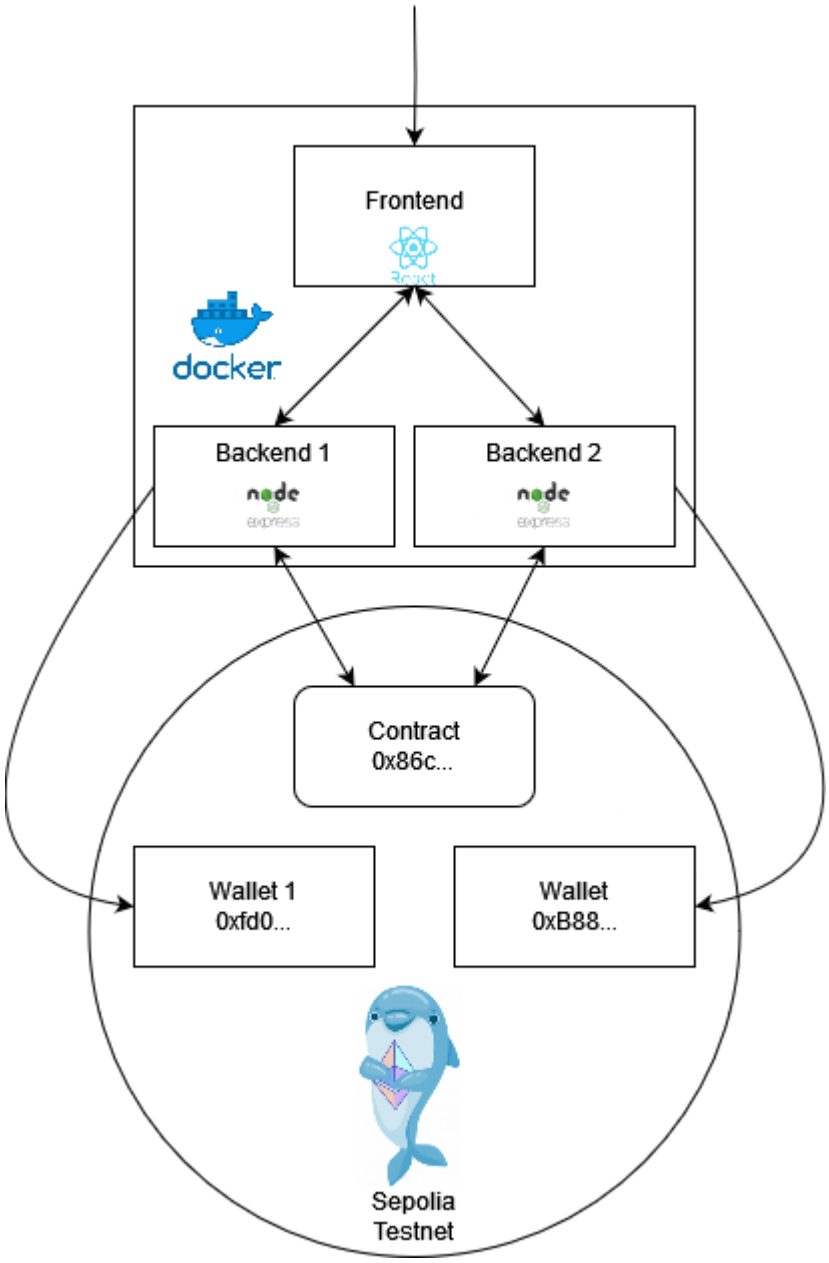
Challenge Task - Blockchain 2023

This is the repository for the challenge task of the lecture Blockchain at OST Rapperswil in FS 2023. It is a application to store and retrieve data on the Sepolia Ethereum testnet, whose main purpose is to show our understanding of the covered topics of the lecture. Hence the focus was not on the available functions, but instead on smartcontracts, wallets and their interaction.



Setup

We are using the following setup:



The access to our application is through the frontend, from there request get sent to two different instances of our backend API, each representing a different party with their own wallet. The backend services interact with the sepolia testnet using the ethers.js library.

Frontend

Our React-based Blockhain tracking app's frontend offers features such as displaying delivery status, enabling status updates, and facilitating the creation of new orders.

The choice of React is driven by its component-based architecture, ease of learning, and proven success in previous projects. The backend is Node-based, ensuring a cohesive technology stack. Choosing React over Angular was influenced by React's flat learning curve and proven track record, as Angular is generally deemed more suitable for larger projects..

We define our http request target in api.js, where we send api calls to the backend services. The following requests are sent from the frontend:

Endpoint	HTTP Method	Description
`/getPackagesByReceiver`	GET	Get packages associated with a specific receiver.
`/getPackagesBySender`	GET	Get packages associated with a specific sender.
`/getAllPackagesForWallet`	GET	Get all packages for a given wallet.
`/createPackage`	POST	Create a new package with specified details.
`/updatePackageStatus`	POST	Update the status of a specific package.
`/confirmPackage`	POST	Confirm the delivery of a specific package.

Backend Services

We are using node.js webservers with express framework for our backend services. We chose that setup because there are many tutorials available for javascript and alchemy has a great framework to interact with the smartcontract. Alchemy also provides a Sepolia Testnet (Ethereum).

We handle the HTTP requests with the express framework, which we translate into functioncalls onto the smartcontract using ether.js. The backend services are using identical code, their Sepolia credentials are provided to them seperately using Docker (environment-) parameters.

Each backend service represents one party with one wallet. They are available under the ports :8080 and :8081.

A helperscript is provided to deploy a new contract.

For additional information in regards to code layout and seperate usage of the backend you can read up on the Backend README.

Smart Contract

The contract is written in solidity and available under contracts. At it's core it is a List of package objects, with the following attributes:

```
struct Package {
    uint256 id;
    PackageStatus status;
    bool confirmation;
    address sender;
    address receiver;
    uint256 deliveryNumber;
    uint256 creationDate;
    uint256 inTransitDate;
    uint256 deliveredDate;
    uint256 confirmationDate;
}
```

Upon creation, sender and receiver must be entered. Those adresses restrict the access to update functions. Only the wallet with the identical address to the sender is allowed to update the status, and only the receiver is allowed to confirm the receival of the delivery. This is achieved through 'require' checks on the message.sender address in the update functions.

For the getPackagesBy* view functions, all attributes are returned. With that, there is full transparancy on the update history, as with every status check the according date is set. There is no error correcting, as we only allow forward updates. Once a status is set, only further status along the chain can be set. A future implementation could allow for an alternativ state like 'cancelled' for error correction. However no data shall be removed.

Sepolia

We have deployed our smartcontract on sepolia, as this was the testnet recommended by the lecturer and we found it working well and be relatively easy to use. We set up a total of 4 wallets, however in this project we are only using 2 main wallets for interaction and one spare wallet to show the functionality of our filter on the frontend. We are accessing the Sepolia Testnet through alchemy under the following baseurl: https://eth-sepolia.g.alchemy.com/v2/

Usage

Run: Having Docker running and docker-compose available execute the following in the challenge-task directory:

```
docker-compose -f docker-compose.yml up
```

The service will be available under:

```
http://localhost:3000/
```

Cleanup:

```
docker-compose down -v
```

Authors

Leonie Däullary, Linus Flury, Kevin Kempf