EVENT TICKET RESELLING

BLOCKCHAIN PROJECT WITH THE AIM TO MITIGATE TICKET SCAMMERS

CONTENT

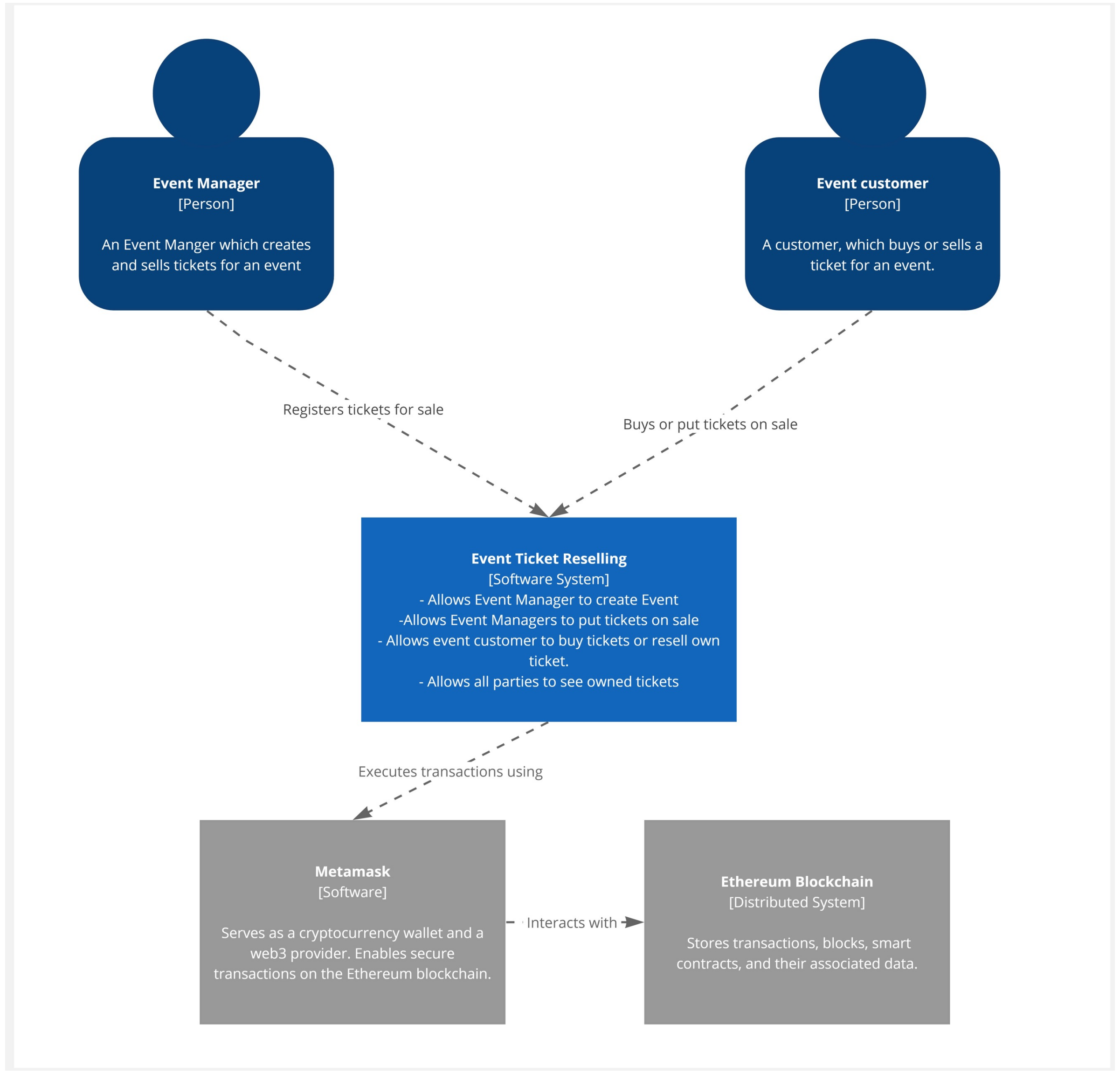
IDEA

- Building a transparent ticket selling and reselling platform
- Opportunity to buy ticket as Non-Fungible Token (NFT)
- Mitigate scammers
  - Proof to own the ticket
  - Proof that ticket is still valid
  - Guarantee that ticket will be transferred after buying (contract)

TECHSTACK / DESIGN DECISIONS

Application	NEXT.js TS npm Vercel viem wagmi REACT HOOKS FOR ETHEREUM WAGMI.SH moralis WS
Smart Contract	OpenZeppelin sourcify.eth Etherscan
Other Tooling	QD miro T

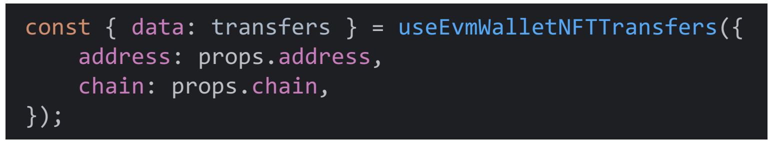
ARCHITECTURE


Event Manager
[Person]
An Event Manger which creates and sells tickets for an event
Event customer
[Person]
A customer, which buys or sells a ticket for an event.
Registers tickets for sale
Buys or put tickets on sale
Event Ticket Reselling
[Software System]
- Allows Event Manager to create Event
-Allows Event Managers to put tickets on sale
- Allows event customer to buy tickets or resell own ticket.
- Allows all parties to see owned tickets
Executes transactions using
Metamask
[Software]
Serves as a cryptocurrency wallet and a web3 provider. Enables secure transactions on the Ethereum blockchain.
Interacts with
Ethereum Blockchain
[Distributed System]
Stores transactions, blocks, smart contracts, and their associated data.

IMPLEMENTATION - APPLICATION

moralis

Reading data from the blockchain

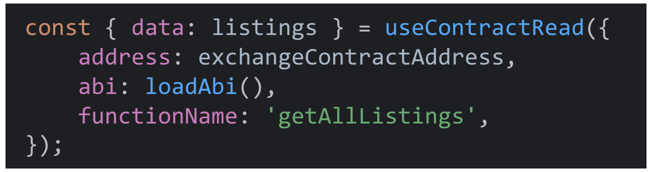


```
const { data: transfers } = useEvmWalletNFTTransfers({
    address: props.address,
    chain: props.chain,
});
```

wagmi

React Hooks for Ethereum

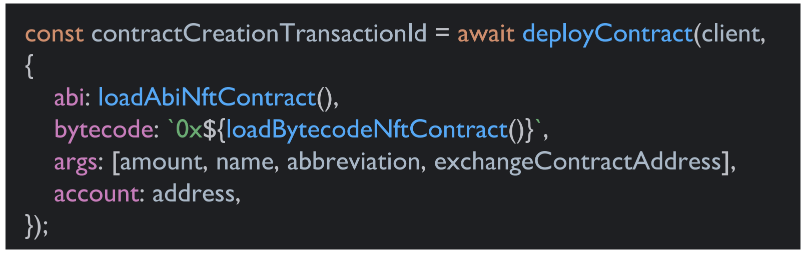
Use read/write methods from the contracts



```
const { data: listings } = useContractRead({
    address: exchangeContractAddress,
    abi: loadAbi(),
    functionName: 'getAllListings',
});
```

viem

Deploy contracts onto the blockchain



```
const contractCreationTransactionId = await deployContract(client,
{
  abi: loadAbiNftContract(),
  bytecode: `0x${loadBytecodeNftContract()}`,
  args: [amount, name, abbreviation, exchangeContractAddress],
  account: address,
});
```

IMPLEMENTATION - CONTRACT

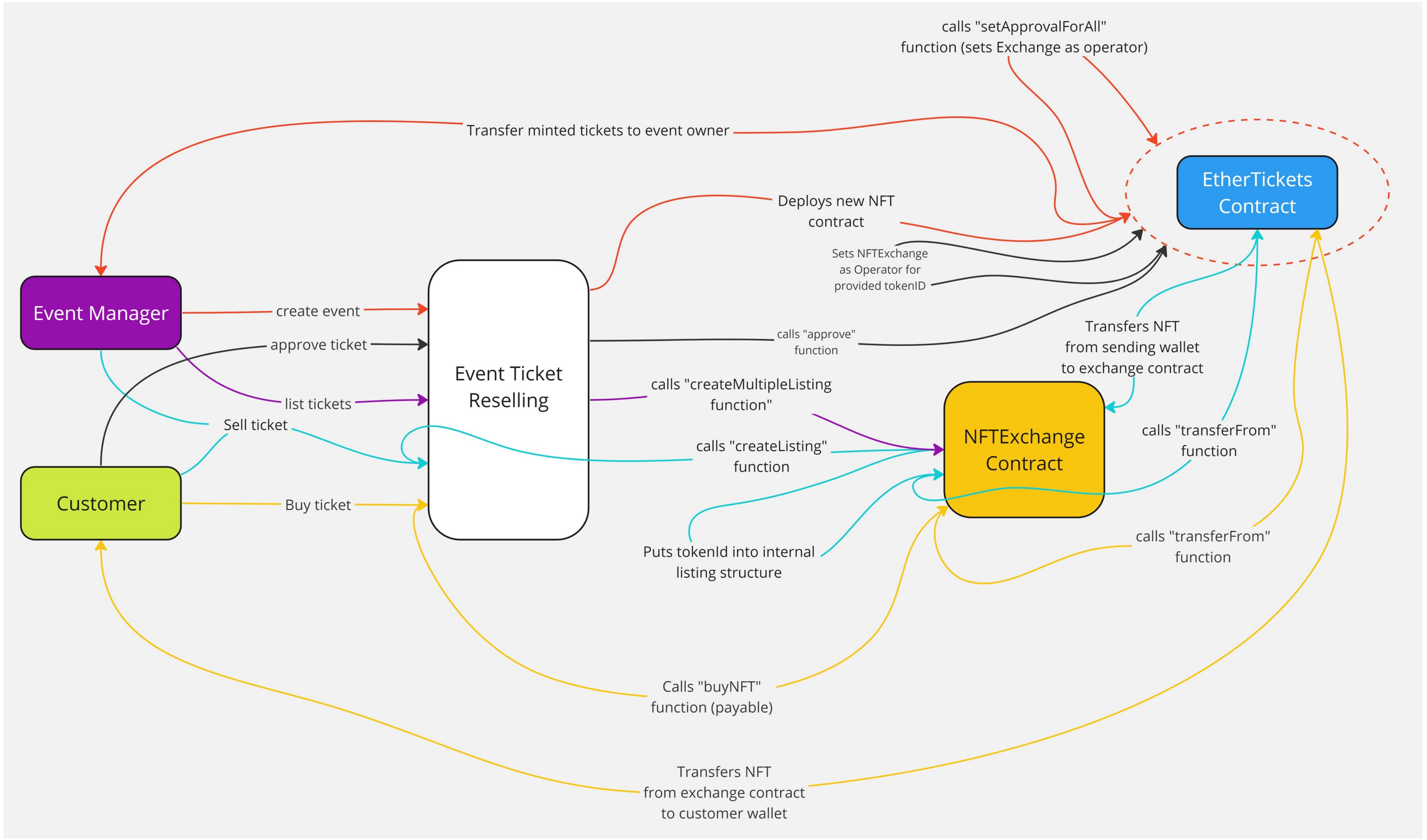
NFTContract "EtherTickets"

- Based on ERC721, ERC721Enumerable, ERC721Burnable and Ownable
- Minting of tickets
- Ownership transfers
- Source OpenZeppelin

Exchange "NFTExchange"

- Based on IERC721 and Ownable
- Used to sell Tickets (NFT's)
- Payment handling
- Source Hackermoon.com

IMPLEMENTATION - WORKFLOW


calls "setApprovalForAll" function (sets Exchange as operator)
Transfer minted tickets to event owner
EtherTickets Contract
Deploys new NFT contract
Sets NFTExchange as Operator for provided tokenID
Event Manager
create event
approve ticket
list tickets
Sell ticket
Event Ticket Reselling
calls "approve" function
Transfers NFT from sending wallet to exchange contract
calls "createMultipleListing function"
calls "createListing" function
NFTExchange Contract
calls "transferFrom" function
Customer
Buy ticket
Puts tokenId into internal listing structure
calls "transferFrom" function
Calls "buyNFT" function (payable)
Transfers NFT from exchange contract to customer wallet

LIVE-DEMO

CONCLUSION

Further improvements

- Feature to validate tickets for customer/event entrance
- Include only our EtherTickets-NFTs in ticket view
- Include reference to further event informations, which are stored off-chain

Retrospective

- Overall fun project with good learning curve
- Idea was maybe to complex/big for this challenge task
- Using next.js frontend framework without prior knowledge was not a good idea
- Ticket reselling is not very user friendly (since you have two Metamask Pop-Ups), but we weren't able to implement Delegatecall in the contract properly

THANK YOU