Crowdsale for:

J3KOIN

BRIDGING ASSETS AND COMMUNITIES

Offered by J3K

In this Presentation

Here's what we'll cover:

J3K

Making the inaccessible accessible

DAO

Decentralized Automonous Organization Designed to help ALL people establish and grow wealth

Secure

Developed for the Blockchain to increase trust between parties, reduce costs/complexity, save time, and break down barriers

Adaptable/Smart

Smart Contracts and Tokens built to change with the people who own them

EST 2022 J3KOIN BRIDGING ASSETS AND COMMUNITIES

Our Mission

"People-first DAO on a mission to help those we serve achieve financial freedom in a secure, smart, and efficient way.”

EST 2022
J3KOIN
BRIDGING ASSETS AND COMMUNITIES

Introducing

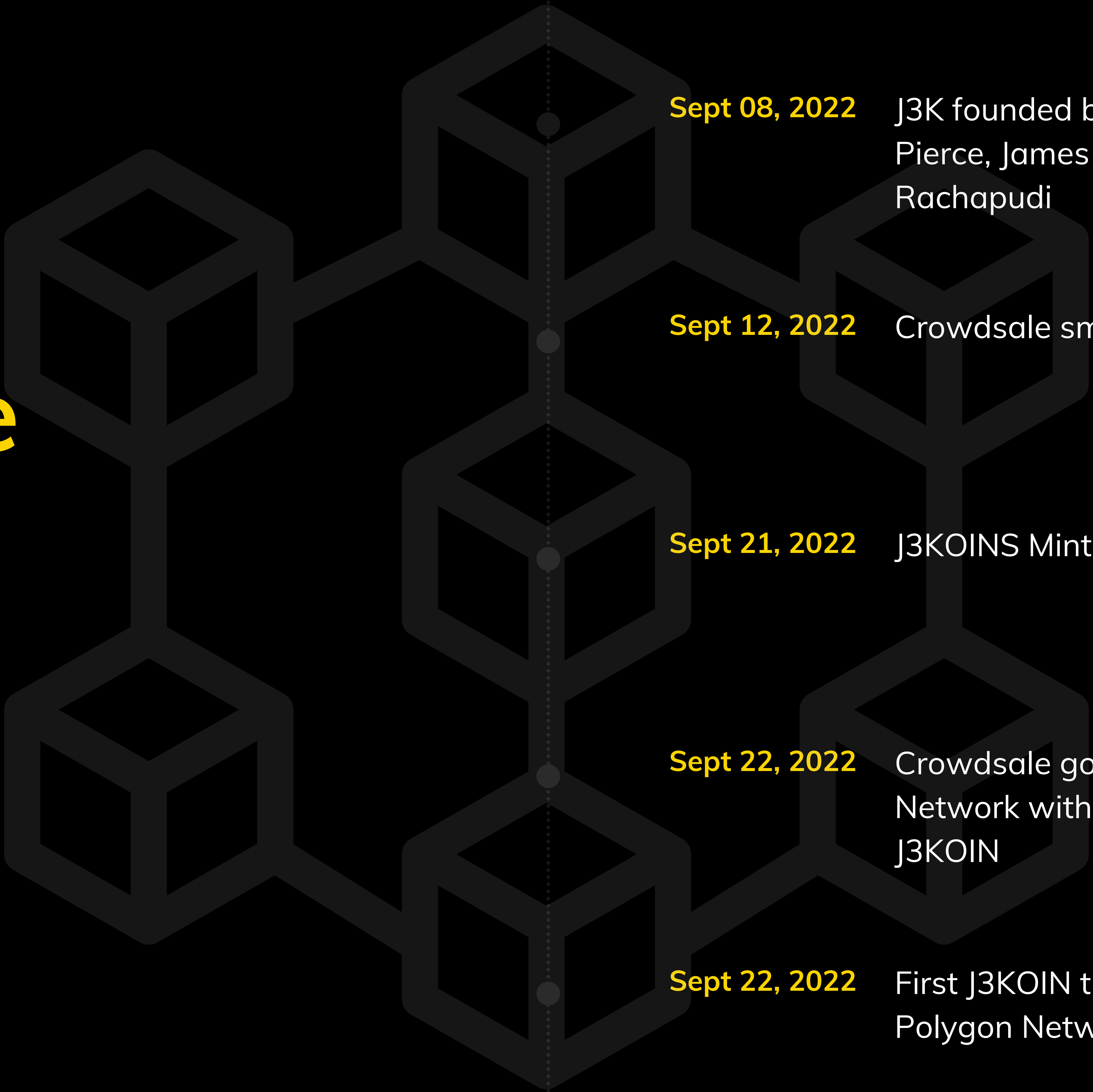
J3KOIN

Fungible token designed to help people globally bridge assets and communities

Solidity based Smart contracts utilizing blockchain and the polygon network.

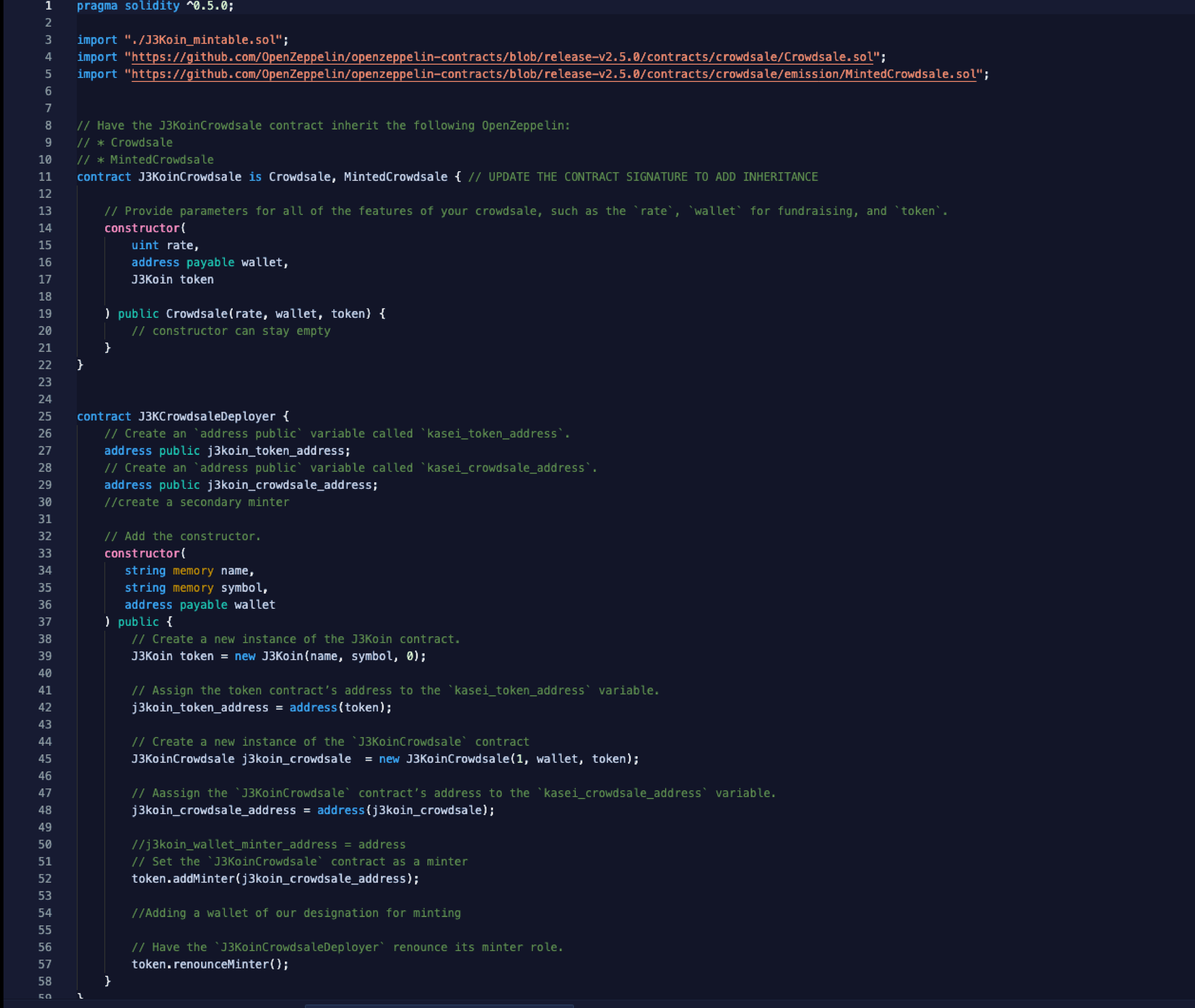
First Edition J3KOIN

The Crowdsale



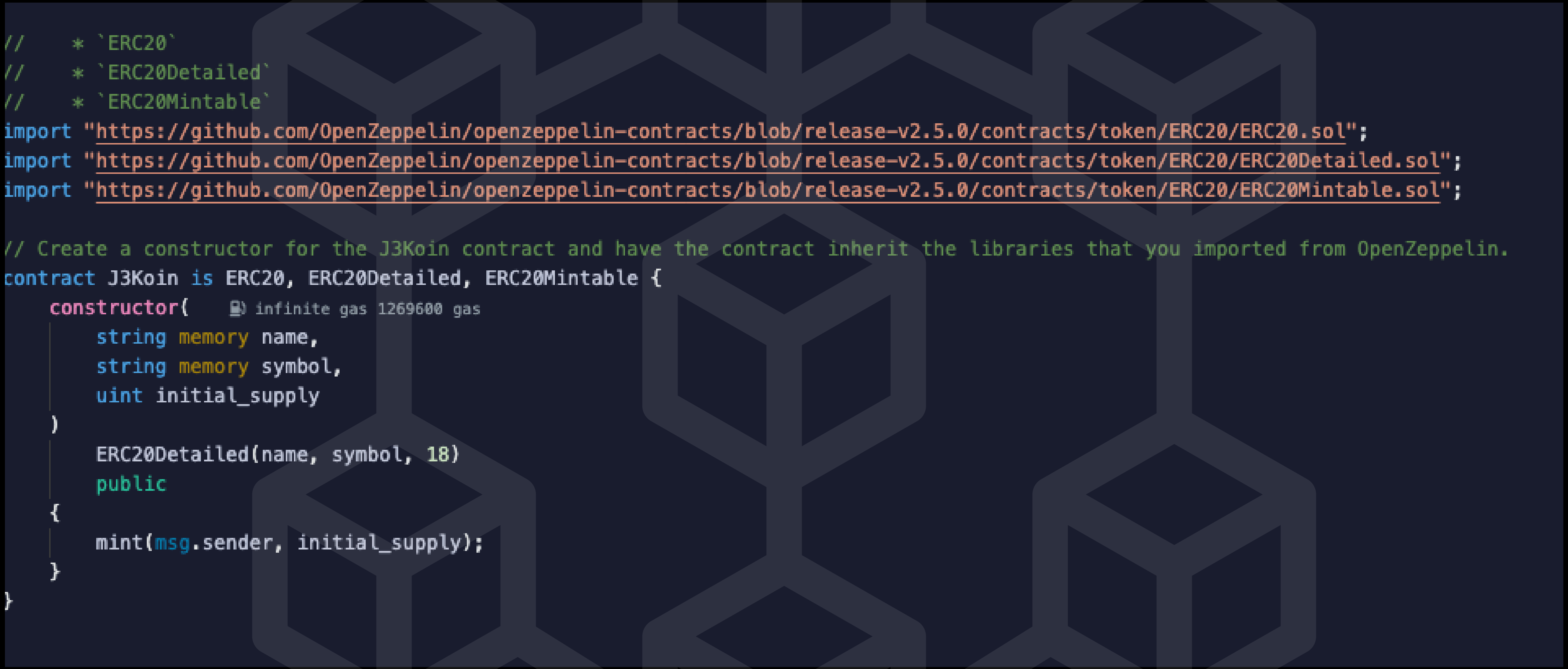
Sept 08, 2022	J3K founded by Jay Wiley, Jeremy Pierce, James Willis, and Kalyan Rachapudi
Sept 12, 2022	Crowdsale smart contract created
Sept 21, 2022	J3KOINS Minted by J3K
Sept 22, 2022	Crowdsale goes live on the Polygon Network with the first version of J3KOIN
Sept 22, 2022	First J3KOIN transaction on the Polygon Network

Crowdsale Contract



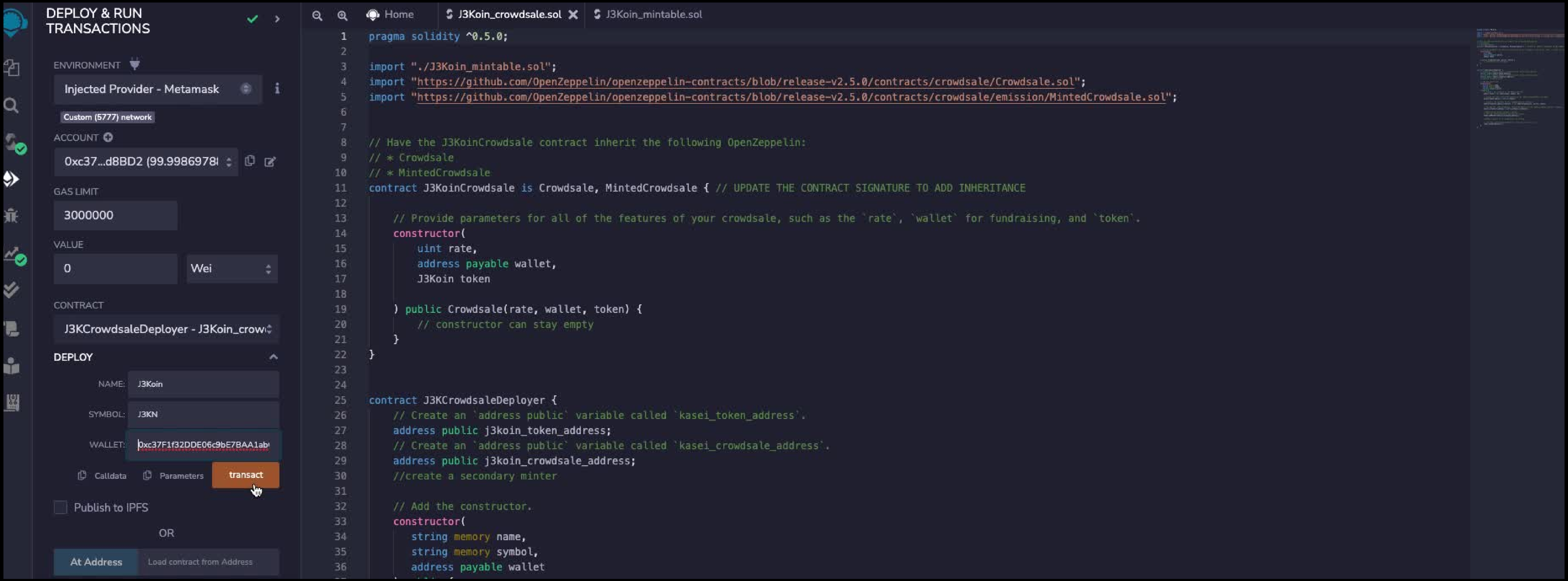
```solidity
pragma solidity ^0.5.0;

import "./J3Koin_mintable.sol";
import "https://github.com/OpenZeppelin/openzeppelin-contracts/blob/release-v2.5.0/contracts/crowdsale/Crowdsale.sol";
import "https://github.com/OpenZeppelin/openzeppelin-contracts/blob/release-v2.5.0/contracts/crowdsale/emission/MintedCrowdsale.sol";


// Have the J3KoinCrowdsale contract inherit the following OpenZeppelin:
// * Crowdsale
// * MintedCrowdsale
contract J3KoinCrowdsale is Crowdsale, MintedCrowdsale { // UPDATE THE CONTRACT SIGNATURE TO ADD INHERITANCE

    // Provide parameters for all of the features of your crowdsale, such as the `rate`, `wallet` for fundraising, and `token`.
    constructor(
        uint rate,
        address payable wallet,
        J3Koin token

    ) public Crowdsale(rate, wallet, token) {
        // constructor can stay empty
    }
}


contract J3KCrowdsaleDeployer {
    // Create an `address public` variable called `kasei_token_address`.
    address public j3koin_token_address;
    // Create an `address public` variable called `kasei_crowdsale_address`.
    address public j3koin_crowdsale_address;
    //create a secondary minter

    // Add the constructor.
    constructor(
       string memory name,
       string memory symbol,
       address payable wallet
    ) public {
        // Create a new instance of the J3Koin contract.
        J3Koin token = new J3Koin(name, symbol, 0);

        // Assign the token contract's address to the `kasei_token_address` variable.
        j3koin_token_address = address(token);

        // Create a new instance of the `J3KoinCrowdsale` contract
        J3KoinCrowdsale j3koin_crowdsale  = new J3KoinCrowdsale(1, wallet, token);

        // Aassign the `J3KoinCrowdsale` contract's address to the `kasei_crowdsale_address` variable.
        j3koin_crowdsale_address = address(j3koin_crowdsale);

        //j3koin_wallet_minter_address = address
        // Set the `J3KoinCrowdsale` contract as a minter
        token.addMinter(j3koin_crowdsale_address);

        //Adding a wallet of our designation for minting

        // Have the `J3KoinCrowdsaleDeployer` renounce its minter role.
        token.renounceMinter();
    }
```

Token Contract



```
//    * `ERC20`
//    * `ERC20Detailed`
//    * `ERC20Mintable`
import "https://github.com/OpenZeppelin/openzeppelin-contracts/blob/release-v2.5.0/contracts/token/ERC20/ERC20.sol";
import "https://github.com/OpenZeppelin/openzeppelin-contracts/blob/release-v2.5.0/contracts/token/ERC20/ERC20Detailed.sol";
import "https://github.com/OpenZeppelin/openzeppelin-contracts/blob/release-v2.5.0/contracts/token/ERC20/ERC20Mintable.sol";

// Create a constructor for the J3Koin contract and have the contract inherit the libraries that you imported from OpenZeppelin.
contract J3Koin is ERC20, ERC20Detailed, ERC20Mintable {
    constructor(    infinite gas 1269600 gas
        string memory name,
        string memory symbol,
        uint initial_supply
    )
        ERC20Detailed(name, symbol, 18)
        public
    {
        mint(msg.sender, initial_supply);
    }
}
```

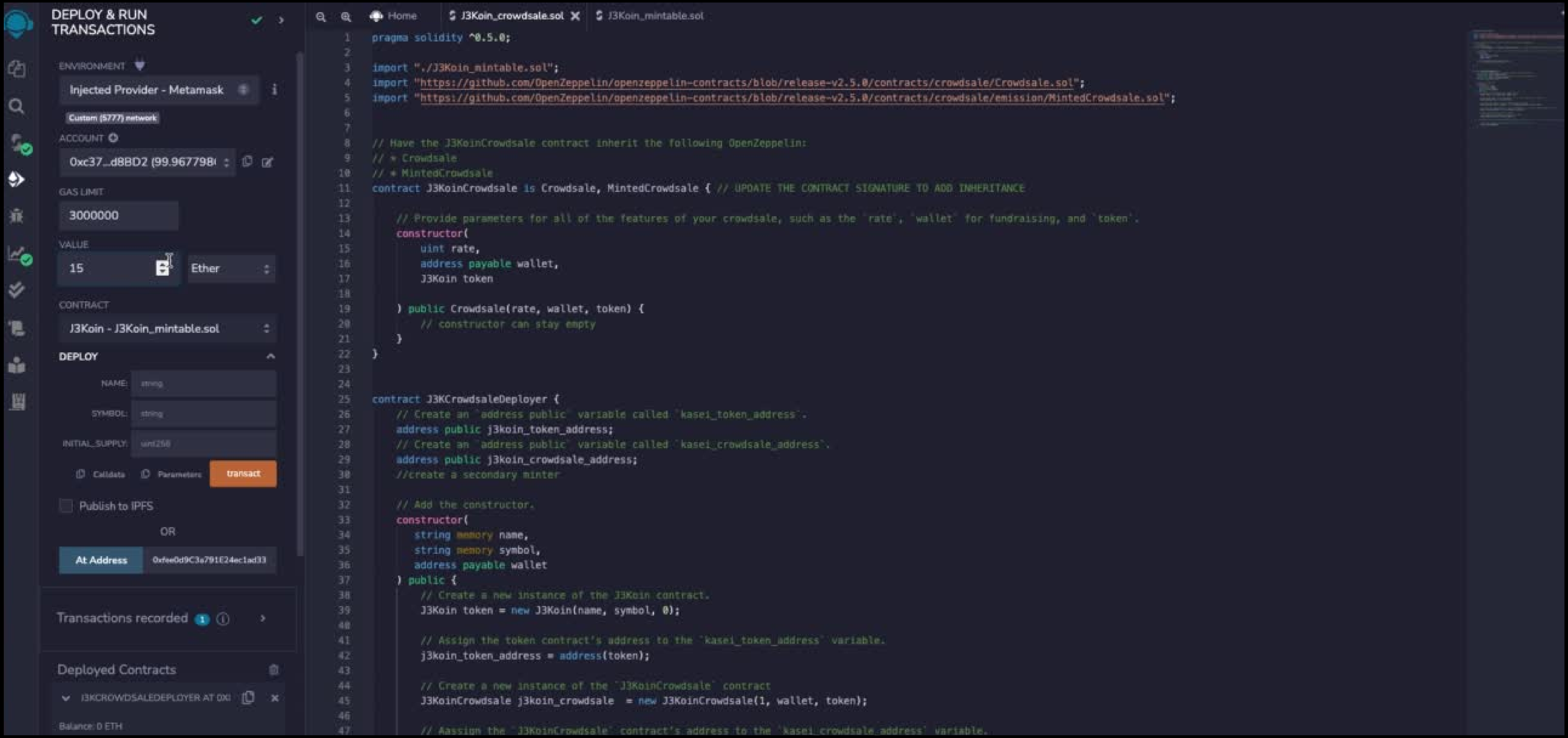
Deployed J3K Smart Contract



ENVIRONMENT: Injected Provider - Metamask
Custom (5777) network
ACCOUNT: 0xc37...d8BD2 (99.9986978
GAS LIMIT: 3000000
VALUE: 0 Wei
CONTRACT: J3KCrowdsaleDeployer - J3Koin_crow
DEPLOY
NAME: J3Koin
SYMBOL: J3KN
WALLET: 0xc37F1f32DDE06c9bE7BAA1ab
Publish to IPFS

```
pragma solidity ^0.5.0;

import "./J3Koin_mintable.sol";
import "https://github.com/OpenZeppelin/openzeppelin-contracts/blob/release-v2.5.0/contracts/crowdsale/Crowdsale.sol";
import "https://github.com/OpenZeppelin/openzeppelin-contracts/blob/release-v2.5.0/contracts/crowdsale/emission/MintedCrowdsale.sol";


// Have the J3KoinCrowdsale contract inherit the following OpenZeppelin:
// * Crowdsale
// * MintedCrowdsale
contract J3KoinCrowdsale is Crowdsale, MintedCrowdsale { // UPDATE THE CONTRACT SIGNATURE TO ADD INHERITANCE

    // Provide parameters for all of the features of your crowdsale, such as the `rate`, `wallet` for fundraising, and `token`.
    constructor(
        uint rate,
        address payable wallet,
        J3Koin token

    ) public Crowdsale(rate, wallet, token) {
        // constructor can stay empty
    }
}


contract J3KCrowdsaleDeployer {
    // Create an `address public` variable called `kasei_token_address`.
    address public j3koin_token_address;
    // Create an `address public` variable called `kasei_crowdsale_address`.
    address public j3koin_crowdsale_address;
    //create a secondary minter

    // Add the constructor.
    constructor(
       string memory name,
       string memory symbol,
       address payable wallet
```

Purchasing J3KOIN



DEPLOY & RUN TRANSACTIONS

ENVIRONMENT: Injected Provider - Metamask

Custom (5777) network

ACCOUNT: 0xc37...d8BD2 (99.9677981

GAS LIMIT: 3000000

VALUE: 15 Ether

CONTRACT: J3Koin - J3Koin_mintable.sol

DEPLOY

NAME: string

SYMBOL: string

INITIAL_SUPPLY: uint256

Publish to IPFS

OR

At Address: 0xfee0d9C3a791E24ec1ad33

Transactions recorded 1

Deployed Contracts

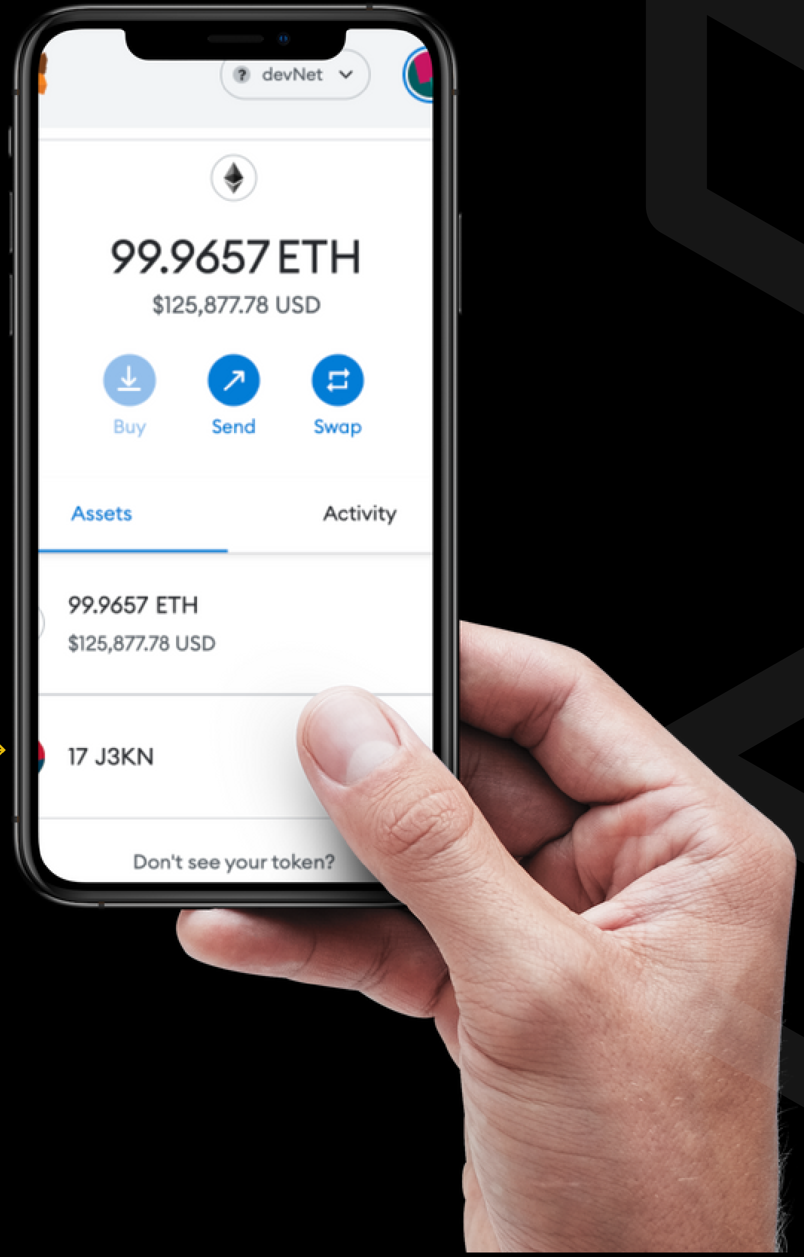
J3KCROWDSALEDEPLOYER AT 0X

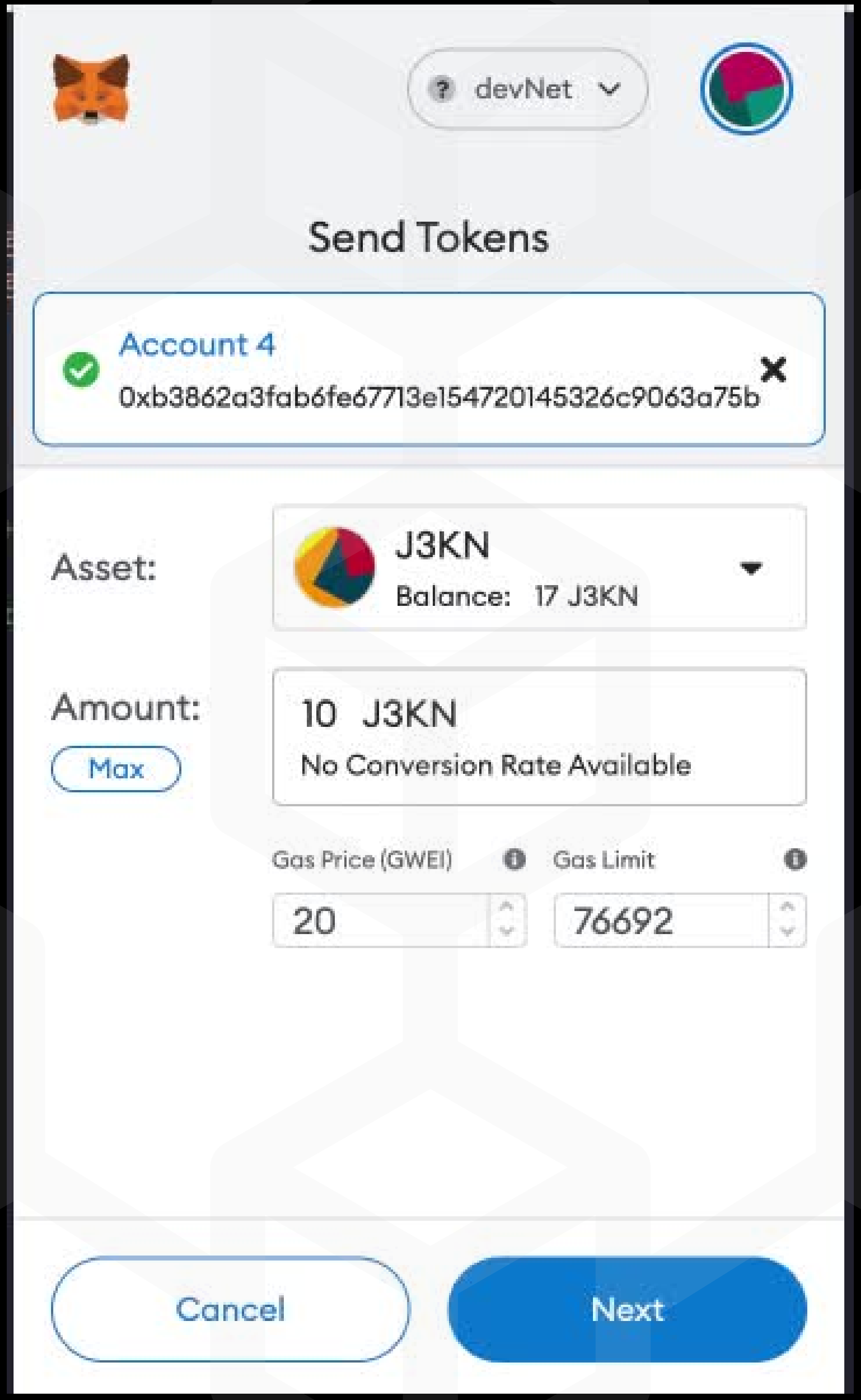
Balance: 0 ETH

```
pragma solidity ^0.5.0;

import "./J3Koin_mintable.sol";
import "https://github.com/OpenZeppelin/openzeppelin-contracts/blob/release-v2.5.0/contracts/crowdsale/Crowdsale.sol";
import "https://github.com/OpenZeppelin/openzeppelin-contracts/blob/release-v2.5.0/contracts/crowdsale/emission/MintedCrowdsale.sol";


// Have the J3KoinCrowdsale contract inherit the following OpenZeppelin:
// * Crowdsale
// * MintedCrowdsale
contract J3KoinCrowdsale is Crowdsale, MintedCrowdsale { // UPDATE THE CONTRACT SIGNATURE TO ADD INHERITANCE

    // Provide parameters for all of the features of your crowdsale, such as the `rate`, `wallet` for fundraising, and `token`.
    constructor(
        uint rate,
        address payable wallet,
        J3Koin token

    ) public Crowdsale(rate, wallet, token) {
        // constructor can stay empty
    }
}


contract J3KCrowdsaleDeployer {
    // Create an `address public` variable called `kasei_token_address`.
    address public j3koin_token_address;
    // Create an `address public` variable called `kasei_crowdsale_address`.
    address public j3koin_crowdsale_address;
    //create a secondary minter

    // Add the constructor.
    constructor(
       string memory name,
       string memory symbol,
       address payable wallet
    ) public {
        // Create a new instance of the J3Koin contract.
        J3Koin token = new J3Koin(name, symbol, 0);

        // Assign the token contract's address to the `kasei_token_address` variable.
        j3koin_token_address = address(token);

        // Create a new instance of the `J3KoinCrowdsale` contract
        J3KoinCrowdsale j3koin_crowdsale  = new J3KoinCrowdsale(1, wallet, token);

        // Aassign the `J3KoinCrowdsale` contract's address to the `kasei_crowdsale_address` variable.
```

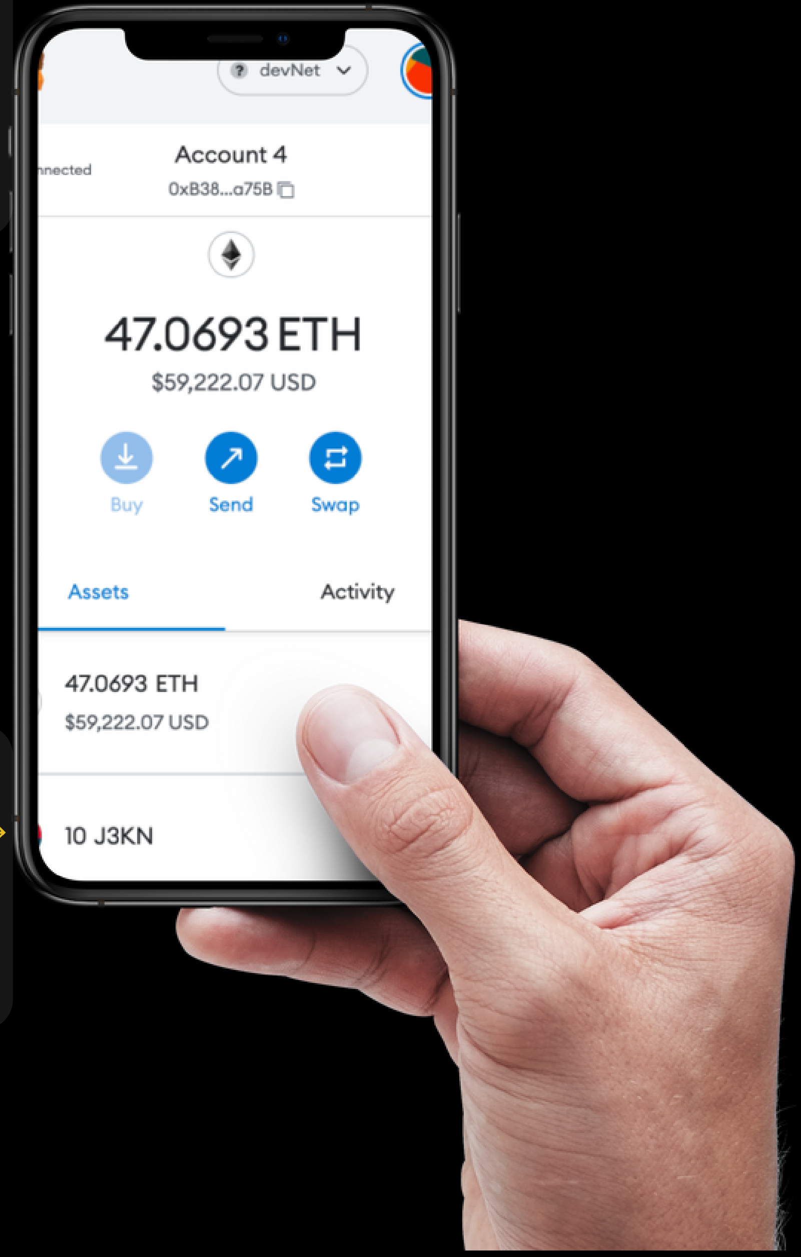
Account Transfer

Account Balance

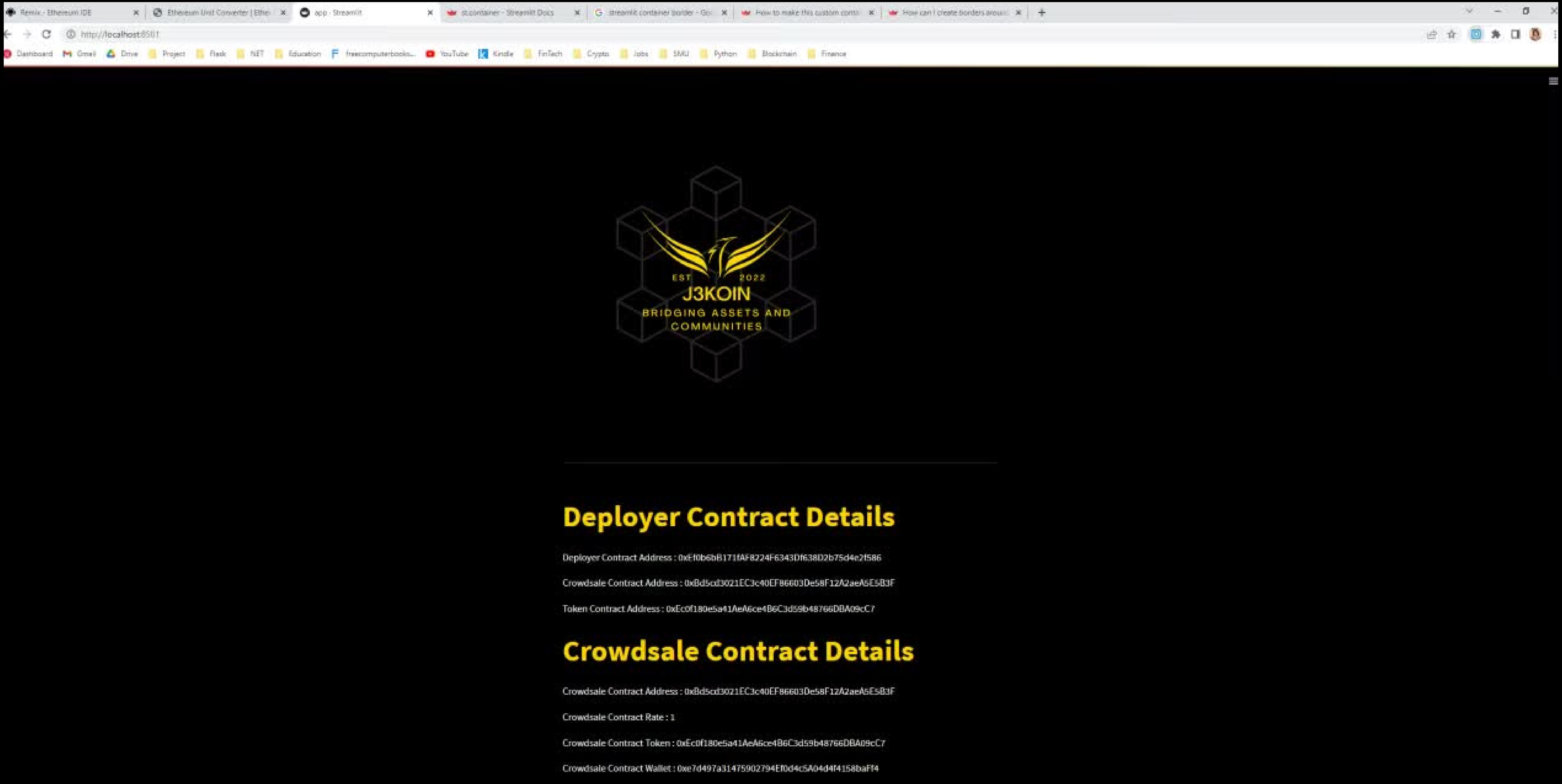




Account Recieve



J3K Client Experience



Deployer Contract Details

Deployer Contract Address : 0xEf0b6bB171fAF8224F6343Df638D2b75d4e2f586

Crowdsale Contract Address : 0xBd5cd3021EC3c40EF86603De58F12A2aeA5E5B3F

Token Contract Address : 0xEc0f180e5a41AeA6ce4B6C3d59b48766DBA09cC7

Crowdsale Contract Details

Crowdsale Contract Address : 0xBd5cd3021EC3c40EF86603De58F12A2aeA5E5B3F

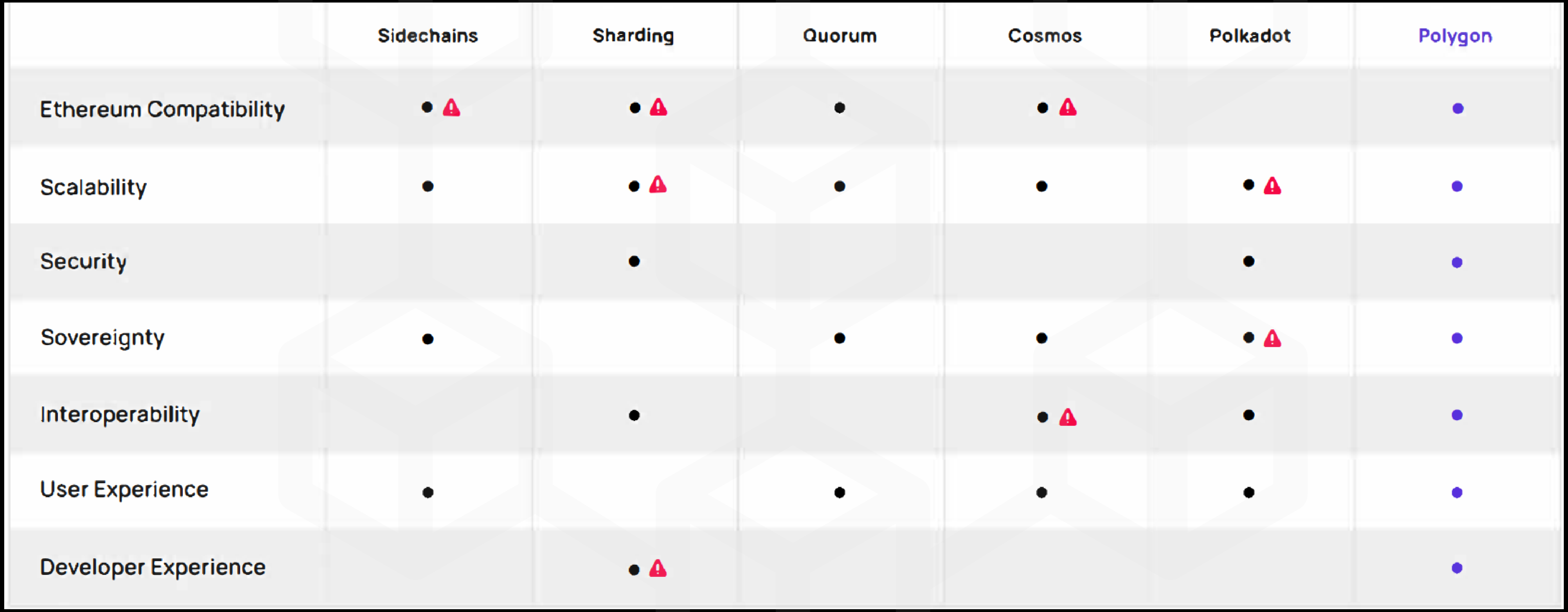
Crowdsale Contract Rate : 1

Crowdsale Contract Token : 0xEc0f180e5a41AeA6ce4B6C3d59b48766DBA09cC7

Crowdsale Contract Wallet : 0xe7d497a31475902794Ef0d4c5A04d4f4158baFf4

Polygon Network

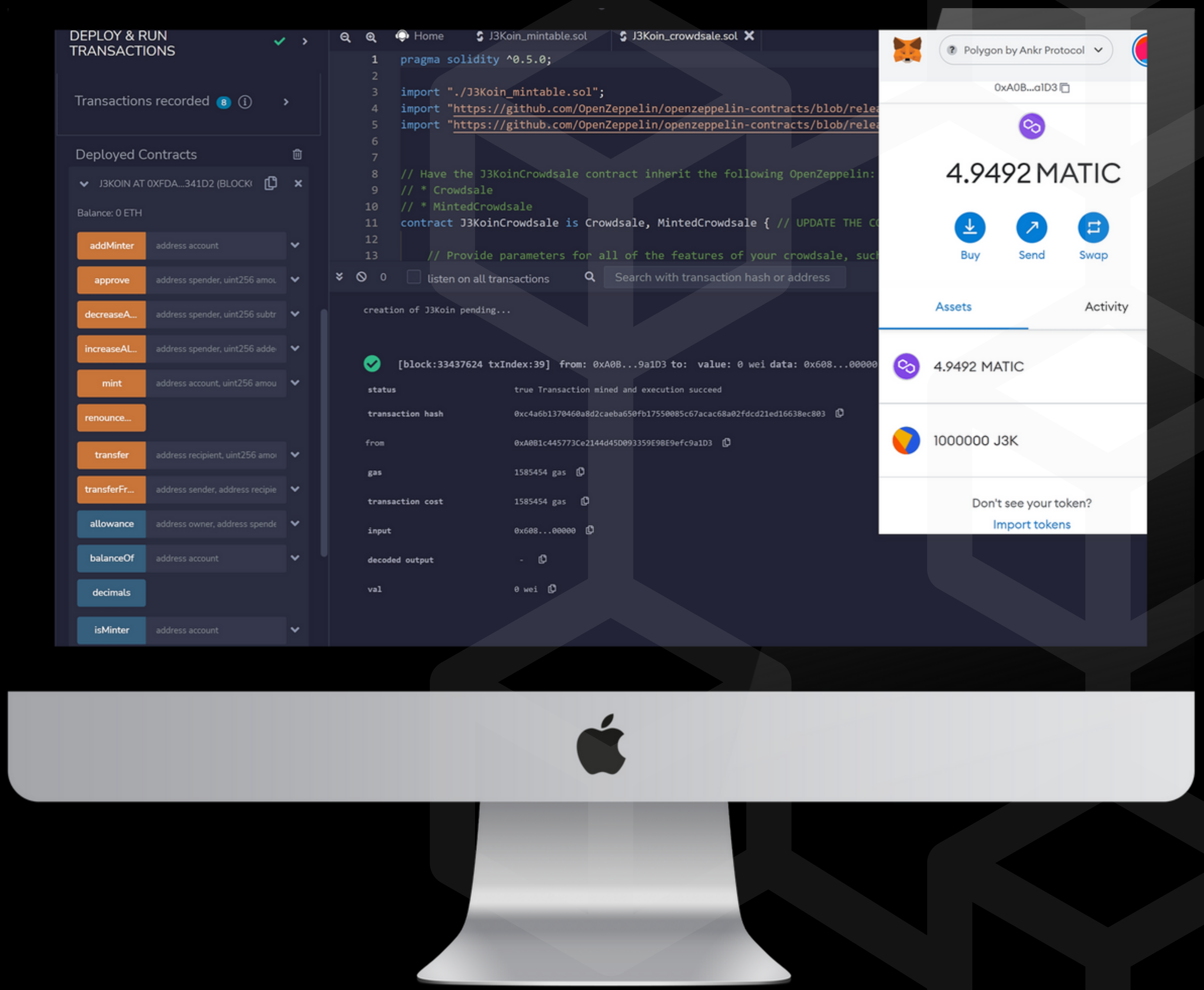
Polygon combines the best of Ethereum and other blockchain platforms without the additional limitations, giving J3Koin those same benefits.



	Sidechains	Sharding	Quorum	Cosmos	Polkadot	Polygon
Ethereum Compatibility	● ⚠	● ⚠	●	● ⚠		●
Scalability	●	● ⚠	●	●	● ⚠	●
Security		●			●	●
Sovereignty	●		●	●	● ⚠	●
Interoperability		●		● ⚠	●	●
User Experience	●		●	●	●	●
Developer Experience		● ⚠				●

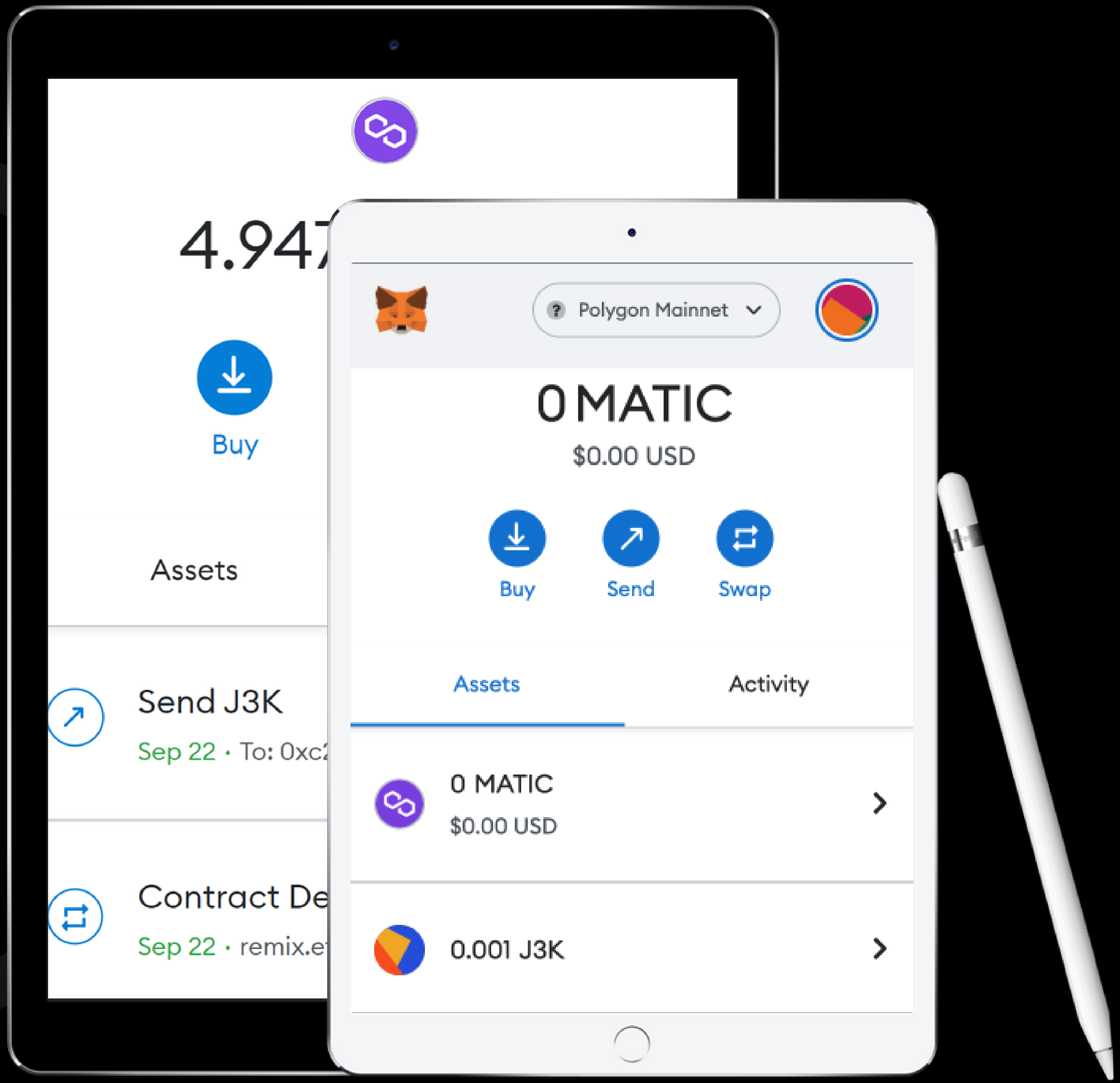
Modular Security **Interoperability** **Scalability** **Low Transaction Fees**

J3KOIN IS LIVE!!!



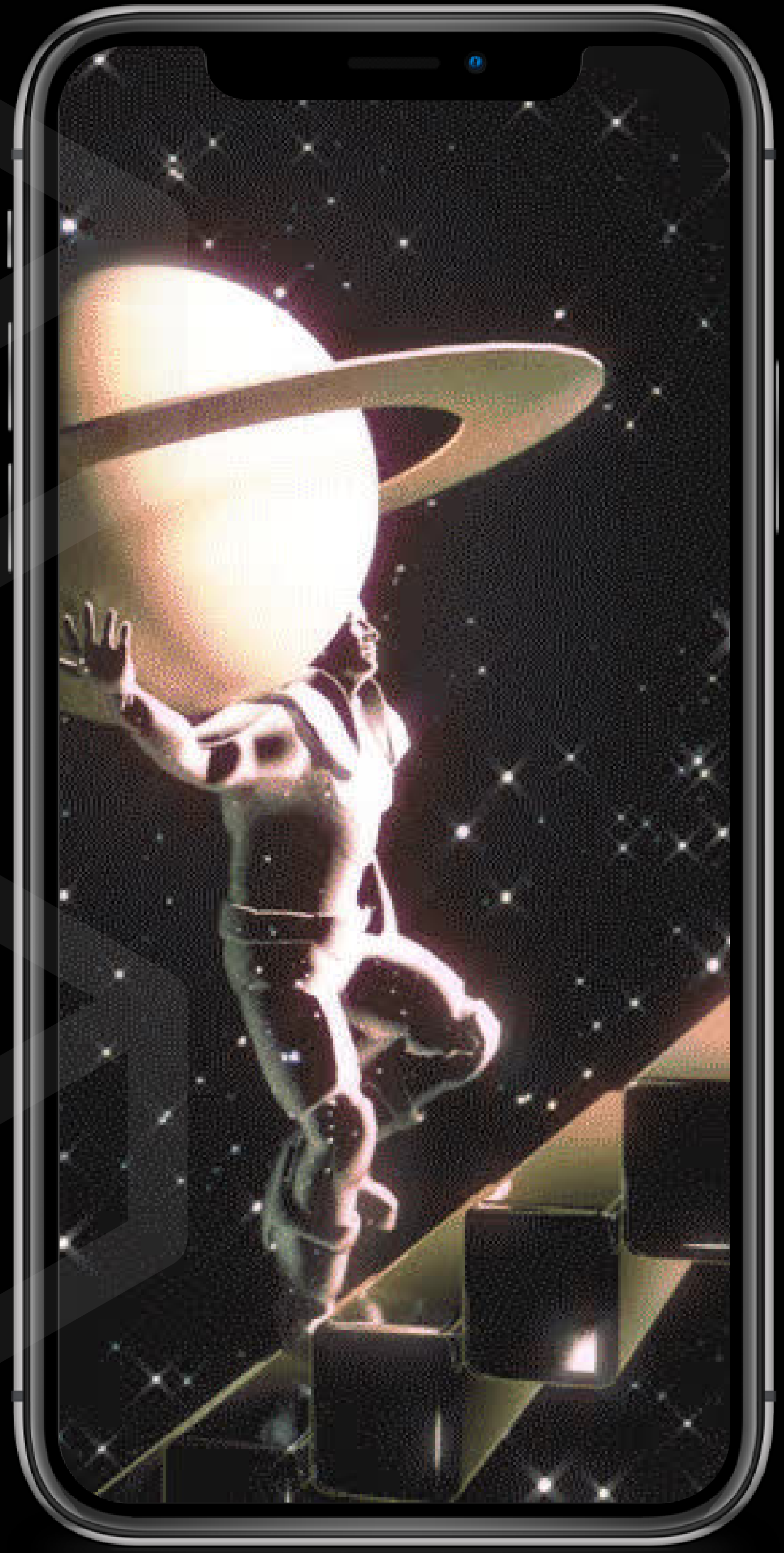
```
pragma solidity ^0.5.0;

import "./J3Koin_mintable.sol";
import "https://github.com/OpenZeppelin/openzeppelin-contracts/blob/relea
import "https://github.com/OpenZeppelin/openzeppelin-contracts/blob/relea


// Have the J3KoinCrowdsale contract inherit the following OpenZeppelin:
// * Crowdsale
// * MintedCrowdsale
contract J3KoinCrowdsale is Crowdsale, MintedCrowdsale { // UPDATE THE CO

    // Provide parameters for all of the features of your crowdsale, such
```

```
creation of J3Koin pending...

[block:33437624 txIndex:39]  from: 0xA0B...9a1D3 to:  value: 0 wei data: 0x608...00000
status              true Transaction mined and execution succeed
transaction hash    0xc4a6b1370460a8d2caeba650fb17550085c67acac68a02fdcd21ed16638ec803
from                0xA0B1c445773Ce2144d45D093359E9BE9efc9a1D3
gas                 1585454 gas
transaction cost    1585454 gas
input               0x608...00000
decoded output      -
val                 0 wei
```



The Future

1. Develop a J3K Dex
2. Enhanced Smart Contract design
3. Client Experience Improvements
4. 3-5 minute How To educational videos



Do you have any questions?

We are glad to help!

EST 2022

J3KOIN

BRIDGING ASSETS AND COMMUNITIES