IDO流程

IDO业务流程

AllocationStaking & IDO

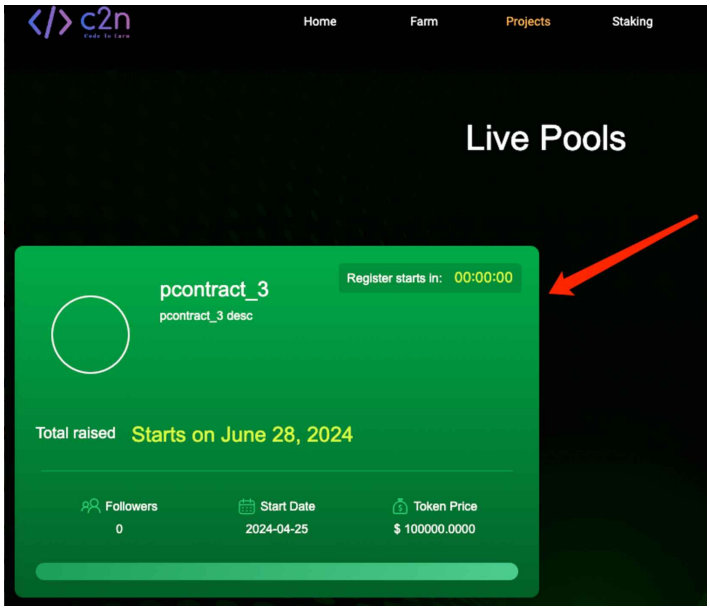
IDO（Initial DEX Offering）是代币发行的方式之一

ICO IEO IPO

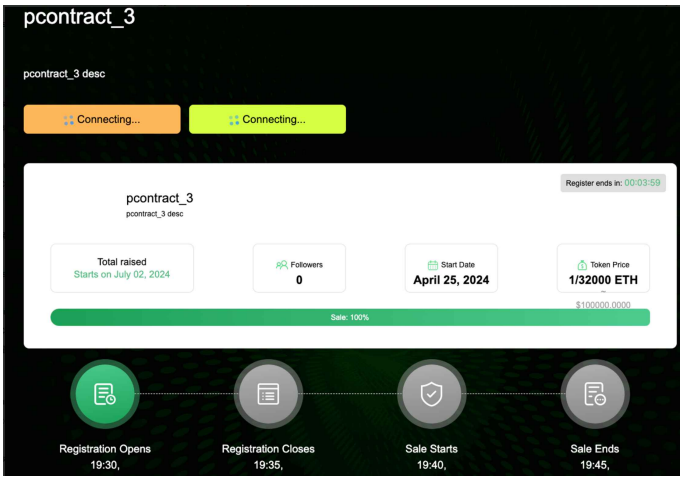
三个角色：

- 平台方
- 项目方
- 投资者

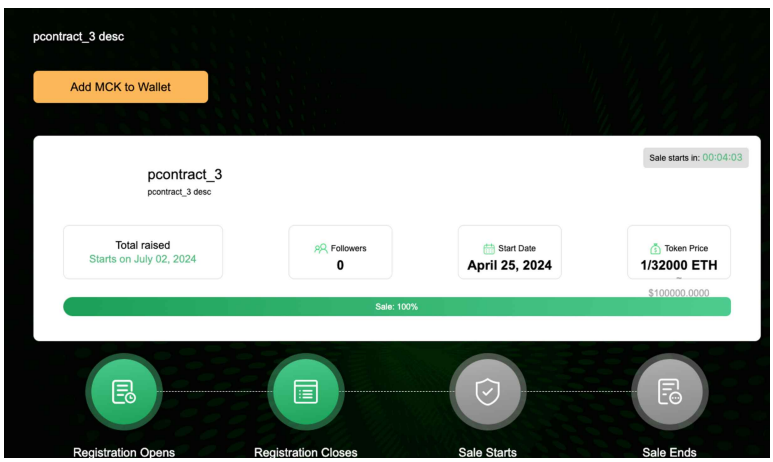
- 项目团队准备好白皮书、智能合约和代币分发计划等必要材料
  - 寻找交易平台发起IDO，并对即将开始的IDO进行各种渠道的推广
- 平台方审核项目方材料后，启动项目IDO
  - 投资者等待IDO流程启动（这期间投资者关注项目的公告和时间表，了解IDO的具体时间和参与方式）



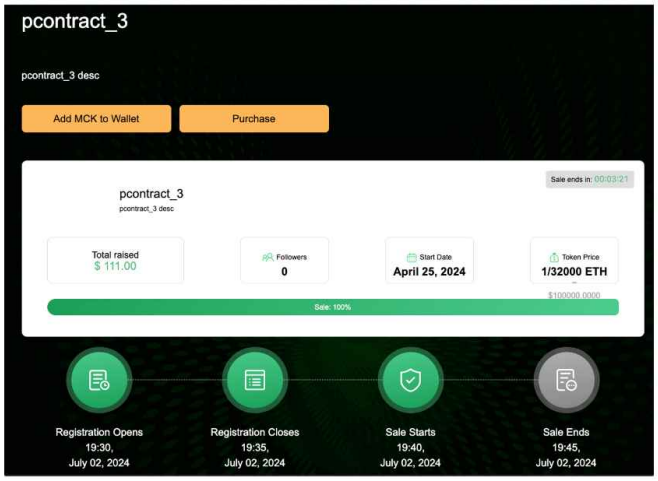
- 项目开始后进入注册流程（这时期投资者通过各种行为获得参与注册的权利）
  - 质押代币：Staking（我们项目使用的参与方式）
  - 资格审查：平台可能会进行KYC和AML审查，确保投资者的合法性（避免一些恶意融资行为）

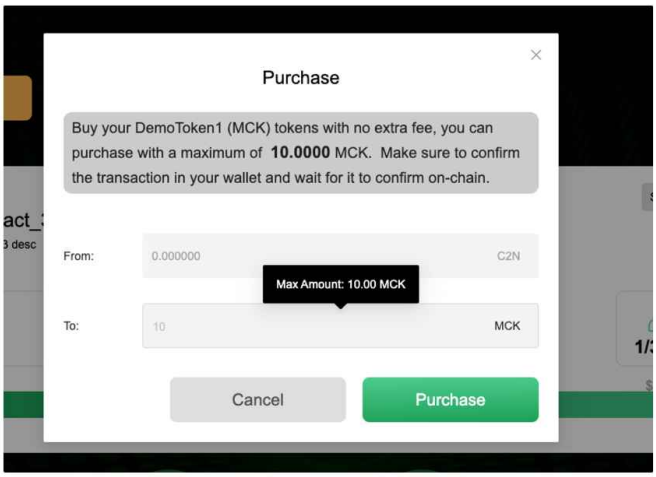


- 注册流程终止，投资者等待sale开始（到出售开始的时间之前，平台方进行各种投资者资格审查、运营等各种活动）

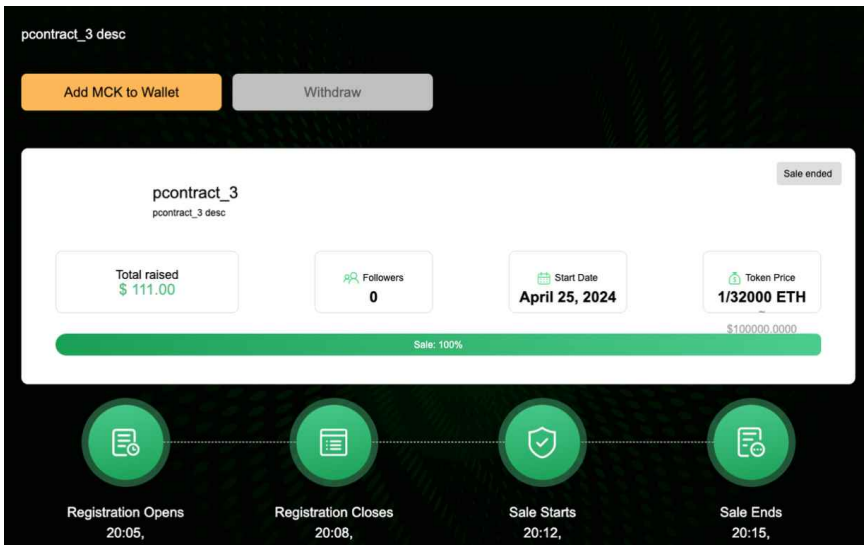


- sale流程开始，（投资者在此期间参与token购买，将自己的资金锁定在代币池中（一般是一些主流代币ETH、USDT等，我们项目使用的链上的nativecoin），这一步不直接发放token）

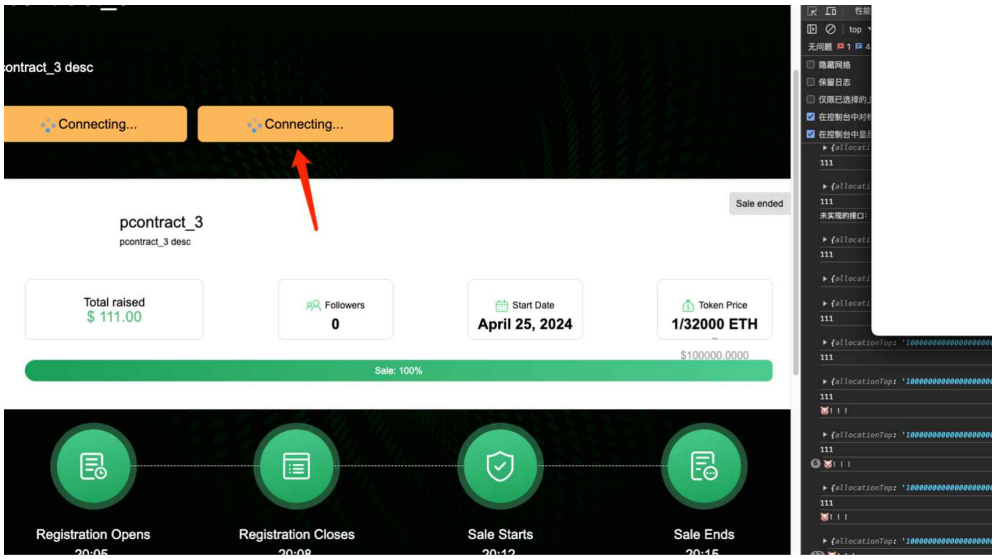




- 交易结束后等待TGE（TGE，代币生成事件），投资者在这之前暂时还不能取走购买的token（这期间平台方准备下面一些操作）
  - 审查交易记录，确定投资人白名单。
  - 准备代币生成事件（TGE）



- 等TGE发生后，用户可以取走购买的代币（mock_token）这一步包括
  - 平台将（Mock_Token）转入代币池（deposit）。（平台方一般不会全部发行，会留存一部分代币到LP中运作）
  - 投资者在TGE后从代币池中提取代币，用于各种交易和投资活动



项目流程

IDO流程整理

- 按照 `c2n-be` 中README的描述部署后台到容器中（需要了解docker的使用），`OWNER_PRIVATE_KEY` 注意填写后面执行部署合约账号的private key。
- 部署脚本

命令写在了makefile里直接运行make ido即可

```
1         ido:
2                 @npx hardhat compile
3                 @npx hardhat run --network local
  scripts/deployment/deploy_boba_token.js
4                 @npx hardhat run --network local
  scripts/deployment/deploy_c2n_token.js
5                 @npx hardhat run --network local
  scripts/deployment/deploy_airdrop_c2n.js
6                 @npx hardhat run --network local
  scripts/deployment/deploy_singletons.js
7                 @npx hardhat run --network local
  scripts/deployment/deploy_mock_token.js
```

- 配置staking信息
  - c2n-fe/src/config/index.js

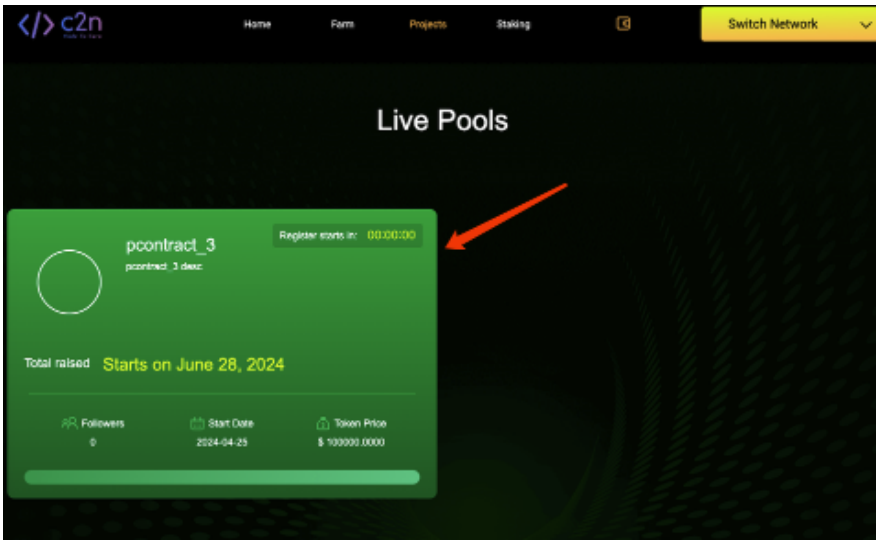
```
    process.env.NEXT_PUBLIC_EARNED_TOKEN_ADDRESS;

// staking address
export const stakingPoolAddresses = [
    {
        chainId: 11155111,
        stakingAddress: "0x6C336a43bC47648Dac96b1419958B8a4e78E05C1",
        depositTokenAddress: "0x4E71E941878CE2afEB1039A0FE16f5eb557571C8",
        earnedTokenAddress: "0x4E71E941878CE2afEB1039A0FE16f5eb557571C8",
    },
    {
        chainId: 31337,
        stakingAddress: "0x2279B7A0a67DB372996a5FaB50D91eAA73d2eBe6",
        // TODO
        depositTokenAddress: "0x5FbDB2315678afecb367f032d93F642f64180aa3", // 填C2N-Token的地址
        earnedTokenAddress: "0x5FbDB2315678afecb367f032d93F642f64180aa3", // 填C2N-Token的地址
    },
];

export const API_DOMAIN = process.env.NEXT_PUBLIC_SERVER_DOMAIN;
```

配置上图三个内容

stakingAddress改为AllocationStakingProxy (地址在contract-address.json的AllocationStakingProxy)

depositTokenAddress和earnedTokenAddress改为C2N-TOKEN的地址 (地址在contract-address.json的C2N-TOKEN)

- c2n-contracts/deployments/contract-addresses.json【配置内容从这里找，ido流程总共只需要用到以下两项】
  - C2N-TOKEN
  - AllocationStakingProxy
- 配置质押token和mock_token信息
  - 将mock-token的地址(地址在contract-address.json的MOCK-TOKEN)配置到c2n-contracts/scripts/configs/saleConfig.json的tokenAddress里
  - 修改.env.development中的地址为【配置】流程中设定的depositTokenAddress和earnedTokenAddress【就是C2N-TOKEN地址】
    - **NEXT_PUBLIC_STAKED_TOKEN_ADDRESS**
    - **NEXT_PUBLIC_EARNED_TOKEN_ADDRESS**
- 跑sale脚本：这一步将项目信息添加到数据库，完成后live pools中可以看到项目信息



项目状态

```
function judgePoolStatus() {
  const now = Date.now();
  if (!mileStones) {
    return;
  }

  if (now < mileStones.registrationTimeStarts) {
    // not started
    setStatus(0);
  } else if (now < mileStones.registrationTimeEnds) {
    // in registration
    setStatus(1);
  } else if (now < mileStones.saleStart) {
    // before sale
    setStatus(2);
  } else if (now < mileStones.saleEnd) {
    // in sale
    setStatus(3);
  } else if (now < mileStones.unlock) {
    // sale ends
    setStatus(4);
  } else if (now >= mileStones.unlock) {
    // sale ends
    setStatus(5);
  }
```

```
// [redacted]: -1:not ready, 0: not started, 1: in registration,
// 2: after registration and before participation,
// 3: in sale,
// 4: sale ended,
const [status, setStatus] = useState(-1);
```

这一步可以反复执行

1. 设置saleConfig.js，主要修改项目时间
    a. 修改unix时间戳
        i. registrationStartAt：半分钟后（中间操作一下脚本传数据到后台就行了）
        ii. registrationLength：100（100s，这一步完成register）
        iii. delayBetweenRegistrationAndSale：10（可以设置为10，因为没有操作，设置非0值）
        iv. saleRoundLength：200（购买时间200s）
        v. TGE：大于registrationStart：对流程无影响（Token Generation Events）
    b. 修改saleConfig的tokenAddress（MockToken的地址）
    c. saleOwner，admin的地址，和后台填的私钥是同一个地址账号
    d. 得到存入后台的json
        i. `make sales`

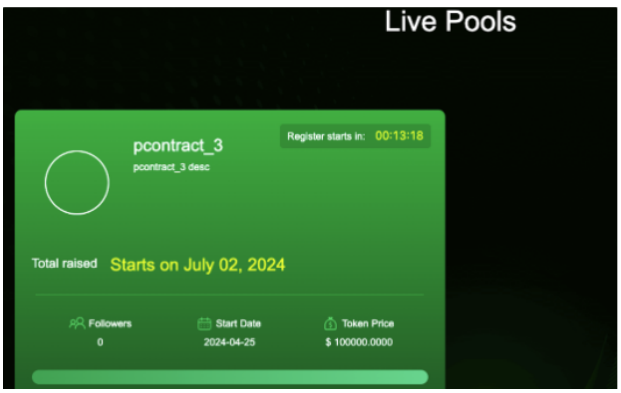
```
  registrationStart: 1719918588,
  registrationEnd: 1719919188,
  saleStartTime: 1719919788
}
{"saleAddress":"0x11dAD4616B0CA8Aed5A75A9E4b9B72d0bd3eDBD0","saleToken":"0xc6e7DF5E7b4f2A278906862b61205850344D4e7d","saleOwner":"0xf39Fd6e51aad88F6F4ce6aB8827279cffFb92266","tokenPriceInEth":"100000000000","totalTokens":"100000000000000000000000000","saleEndTime":1719920388,"tokensUnlockTime":1719919288,"registrationStart":1719918588,"registrationEnd":1719919188,"saleStartTime":1719919788}
zhangjingzheng@zhangjingzhengs-Mac-Studio c2n-contracts %
```

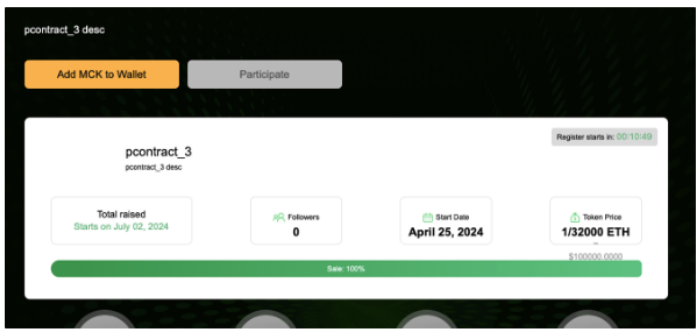
e. 传到后台服务器（按照c2n-be中readme的描述，将其中json字符串替换为上面内容，url填localhost:8080）

```
sh generate_update_data.sh '{"saleAddress":"0x11dAD4616B0CA8Aed5A75A9E4b9B72d0bd3eDBD0","saleToken":"0xc6e7DF5E7b4f2A278906862b61205850344D4e7d","saleOwner":"0xf39Fd6e51aad88F6F4ce6aB8827279cffFb92266","tokenPriceInEth":"100000000000","totalTokens":"100000000000000000000000000","saleEndTime":1719920388,"tokensUnlockTime":1719919288,"registrationStart":1719918588,"registrationEnd":1719919188,"saleStartTime":1719919788}' localhost:8080
```

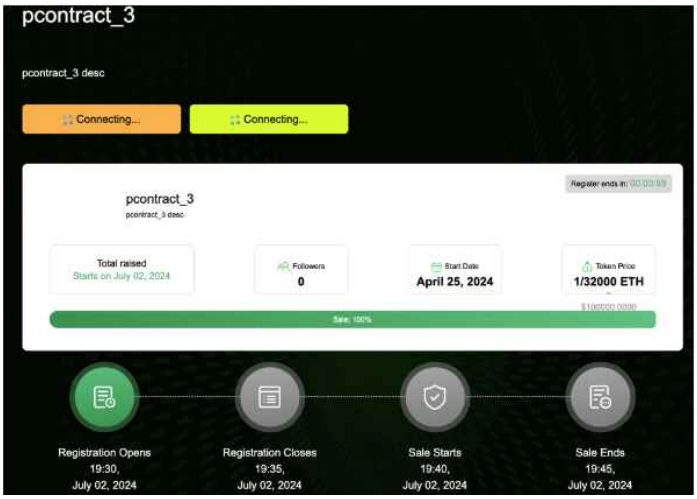
- 初始化项目发行的token数量（将mock_token打入sale地址）
  - `make deposit`

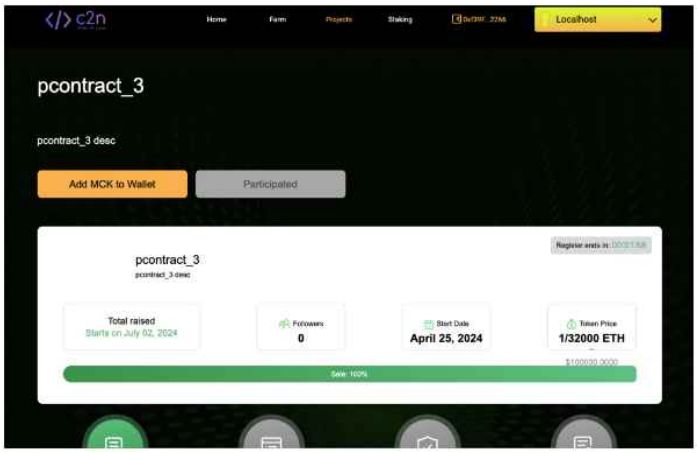
2. 等待注册开始（点击pools，进入等待界面）



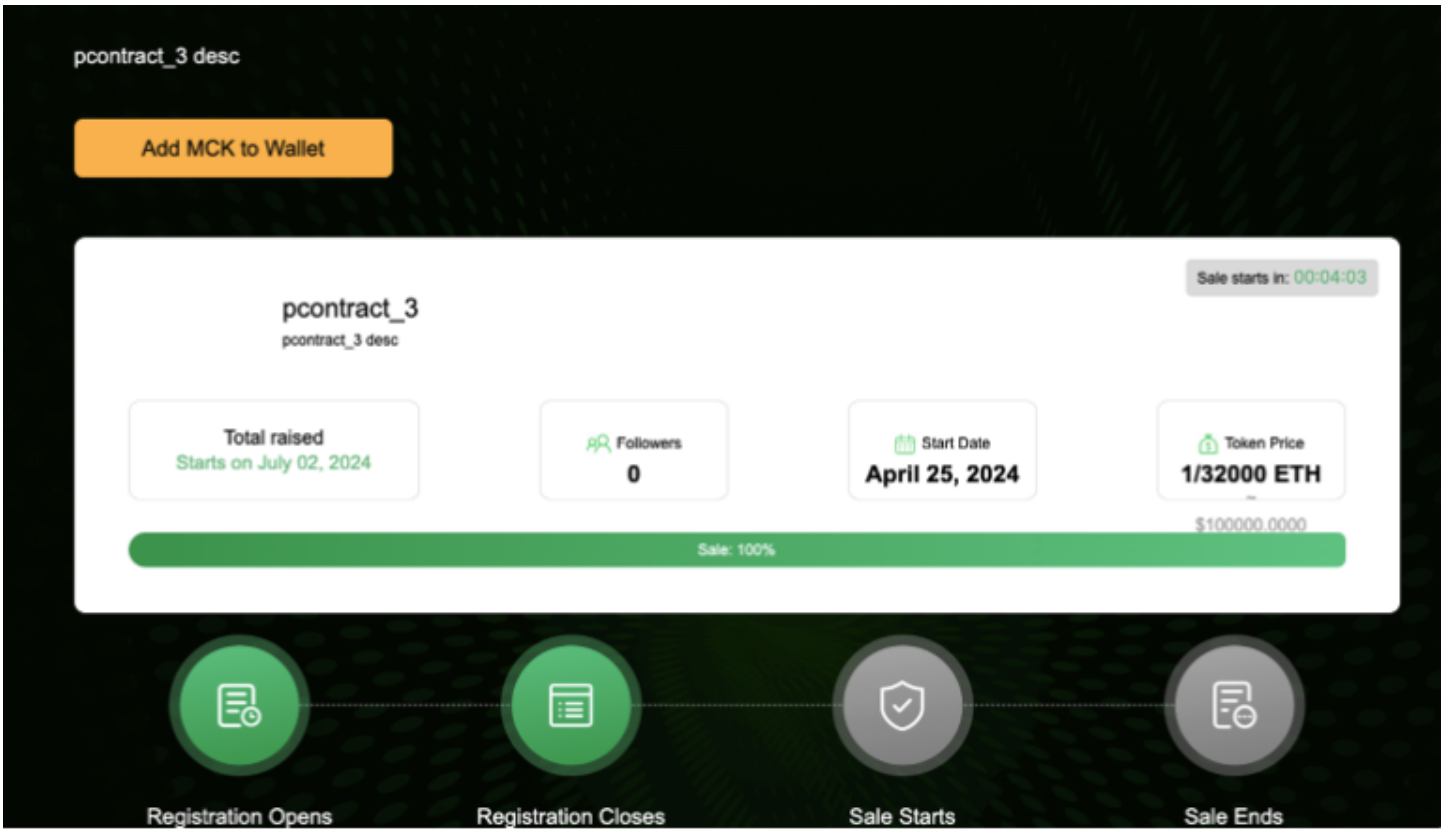


3. 注册开始（在Staking中有质押就能参与注册，没stake会激活stake按钮参与质押）

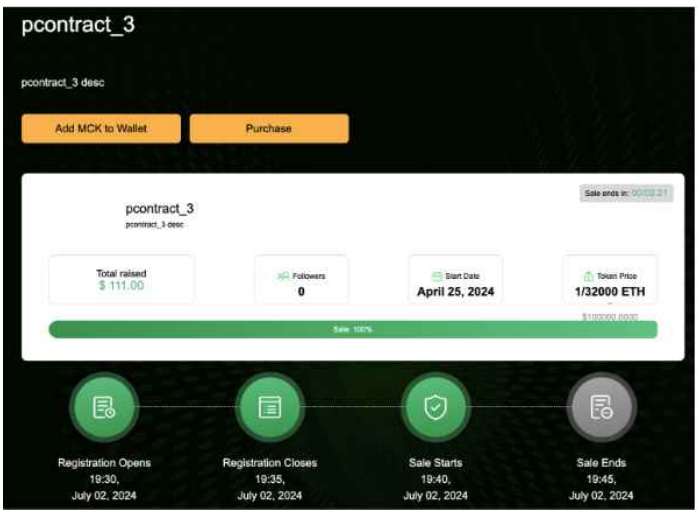


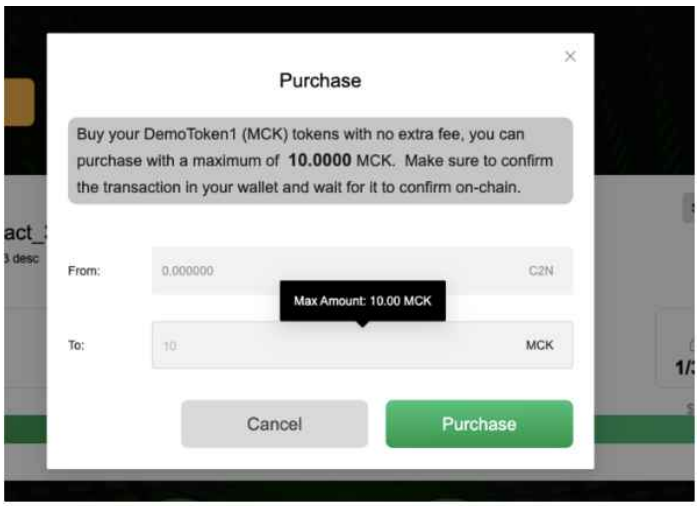


4. 等待sale开始（质押完成后等待sale开始）

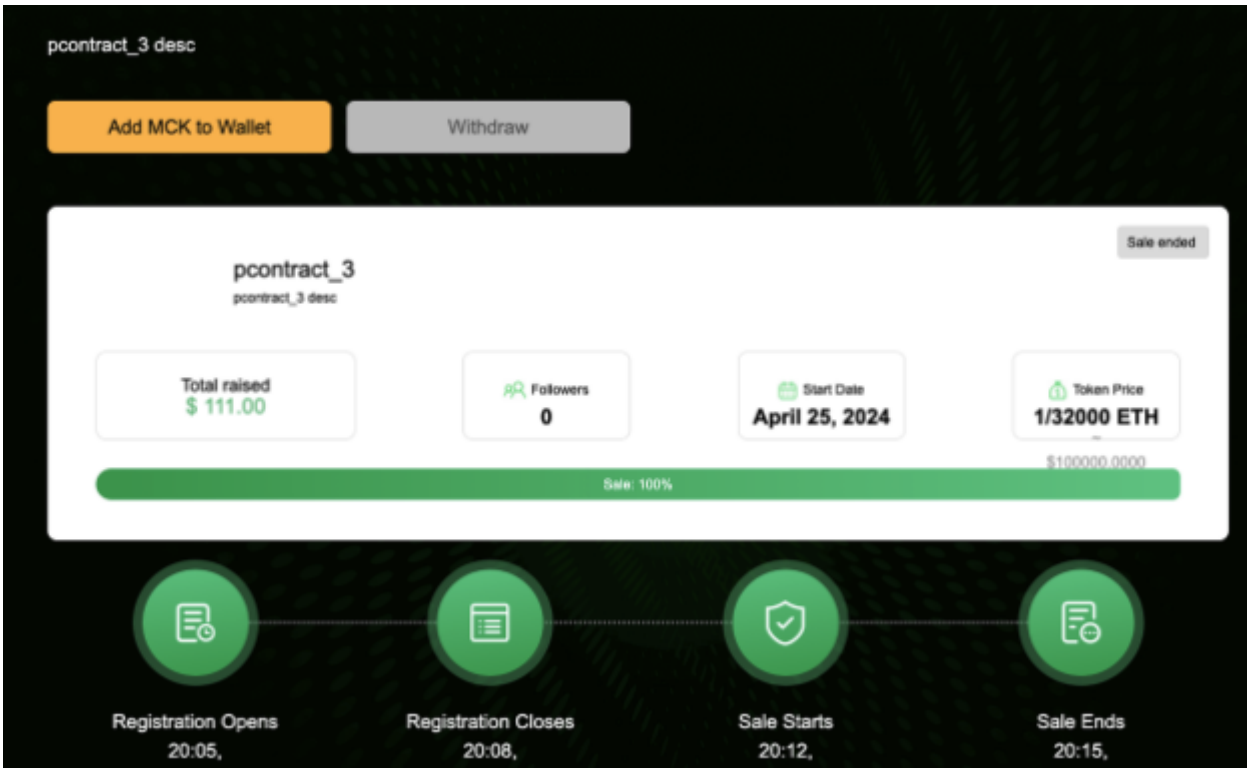


5. sale开始后可以购买mock_token（写死了10个MCK，这一步只将代币转进合约，mock_token还未到账）





6. 交易结束，还未解锁，不可取走mock_token



7. 解锁后可以取走mock_token