

# HackChain

基于 Monad 区块链的去中心化黑客松管理平台

Decentralized Hackathon Management Platform

# 核心功能特性



## 链上管理

在 Monad 测试网上创建、管理和追踪黑客松活动的完整生命周期。



## NFT 门票

发放基于 ERC721 的唯一 NFT 门票，支持二维码验证与签到。



## 实时同步

通过 WebSocket 实时监听链上事件，实现前端数据的毫秒级更新。



## 赞助商系统

透明的链上赞助追踪机制，确保活动资金流向清晰可查。

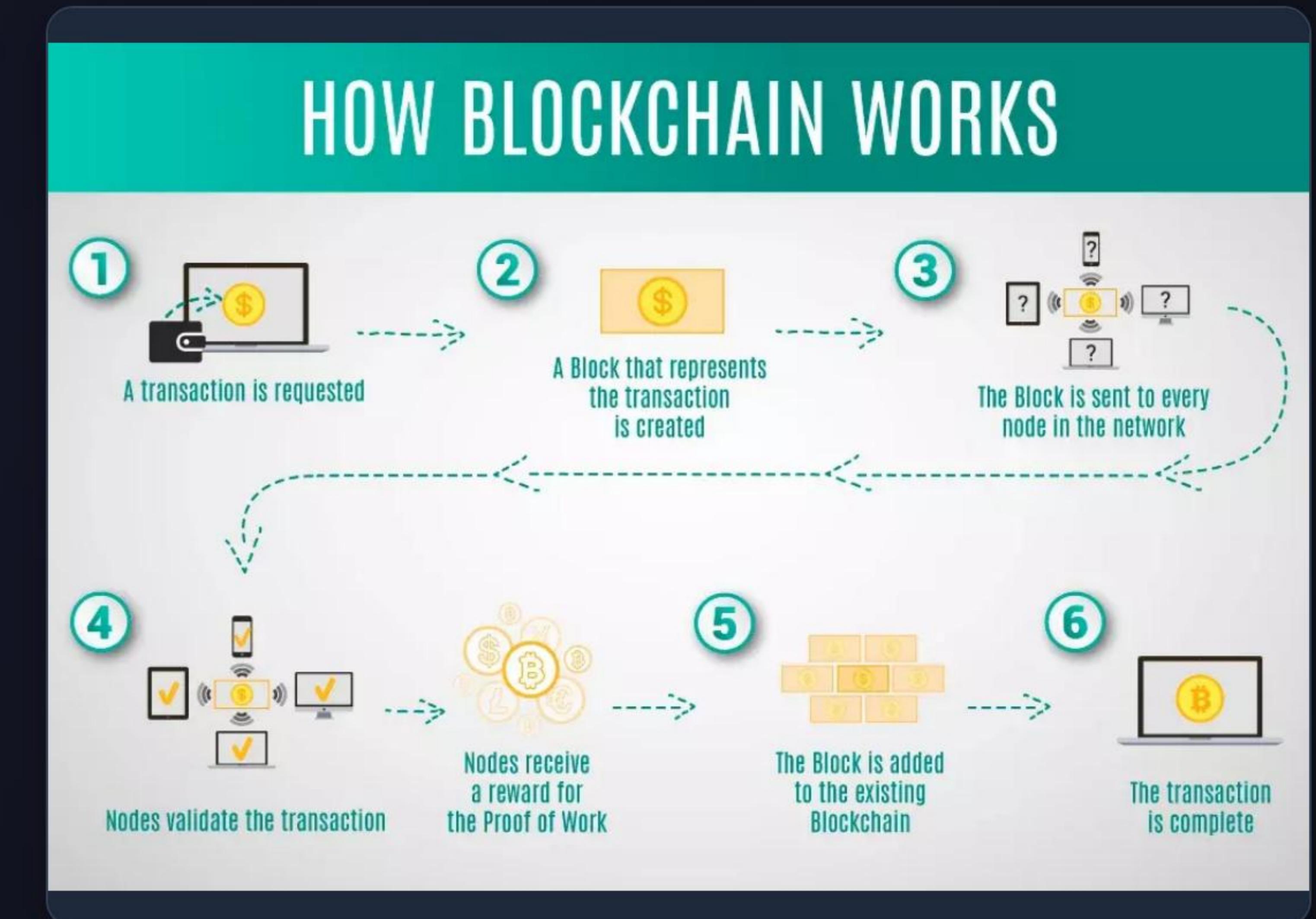
# 系统架构

 前端 React + Vite 构建的用户界面，通过 Ethers.js 与区块链交互。

 后端 Go (Gin) 处理业务逻辑，维护 WebSocket 服务：连接并同步链上数据。

 区块链 Monad 测试网上的智能合约，负责核心逻辑与资产管理。

 数据层：MySQL 存储索引数据，优化查询性能。



# 技术栈

## 前端 (Frontend)

**Framework:** React 19 + Vite

**UI Library:** Material-UI (MUI)

**Web3:** Ethers.js v6

**State:** TanStack Query

## 后端与合约 (Back/Chain)

**Server:** Go 1.21+ (Gin Framework)

**Database:** MySQL 8.0+ (GORM)

**Contract:** Solidity 0.8.27 (Hardhat)

**Network:** Monad Testnet

# 智能合约体系

# Hackathon.sol

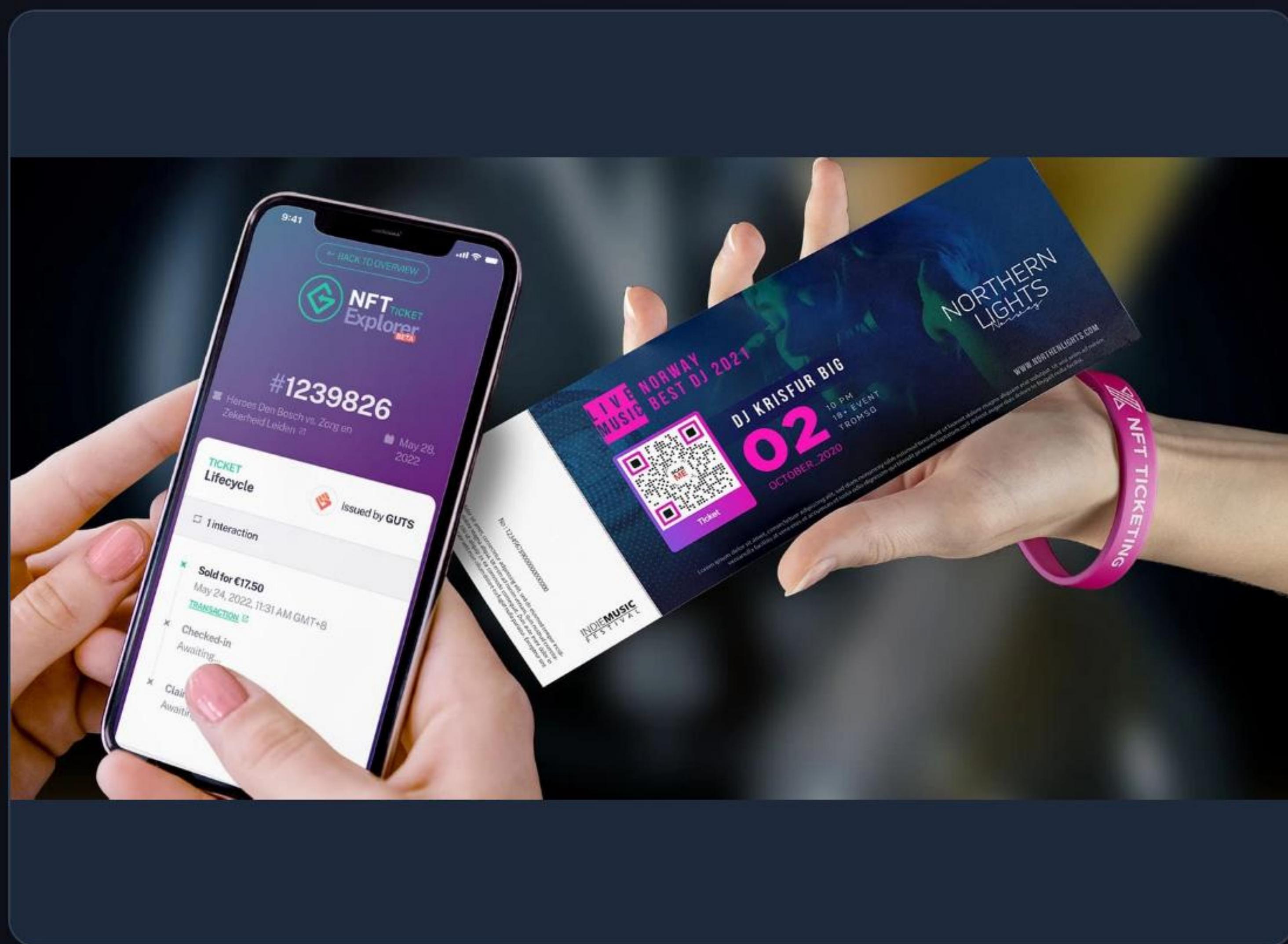
核心管理合约，负责活动的创建、更新、取消，以及参与者注册和赞助商管理逻辑。触发标准事件供后端索引。

# NFTTicket.sol

基于 OpenZeppelin ERC721 标准。为每位注册参与者铸造唯一 Token。Metadata 包含活动关键信息，具备防篡改特性。

```
sicToken {  
    mapping (address => uint256) internal allowed;  
  
    address to another  
    address which you want to send tokens from  
    address which you want to transfer to  
    address count of tokens to be transferred  
    _from, address _to, uint256 _value) public returns (bool) {  
        if (_from != msg.sender) {  
            require(_from == msg.sender);  
            _from].sub(_value);  
            _to].add(_value);  
            allowed[_from][msg.sender].sub(_value);  
            = allowed[_from][msg.sender].sub(_value);  
        }  
        return true;  
    }  
}  
// Transfer tokens on behalf of msg.sender.  
// You can use both the old and new syntax here, this will  
// work as well:  
function transferFrom(address sender, address to, uint256 value) public returns (bool) {  
    if (sender != msg.sender) {  
        require(sender == msg.sender);  
        sender].sub(value);  
        to].add(value);  
        allowed[sender][msg.sender].sub(value);  
        = allowed[sender][msg.sender].sub(value);  
    }  
    return true;  
}
```

# NFT 门票与签到流程



## 去中心化身份验证

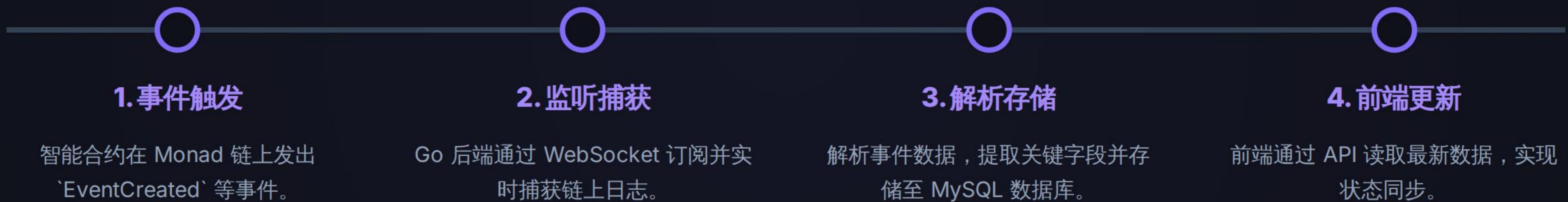
**唯一标识:** Token ID 由 Event ID + 参与者地址哈希生成。

**二维码生成:** 前端根据 Token ID 生成专属二维码。

**移动端扫码:** 组织者使用移动端扫描器读取二维码。

**链上确认:** 验证所有权后，合约记录签到状态，防止重复使用。

# 实时数据同步机制



# API 接口与集成

Method	Endpoint	Description
GET	/api/events	获取所有已创建的黑客松活动列表
GET	/api/events/:id/participants	获取指定活动的注册参与者信息
GET	/api/events/:id/tickets	查询该活动下发放的所有 NFT 门票状态
GET	/api/stats	获取区块链同步状态与系统统计信息

# 安全与可靠性设计



**所有权控制:** 仅活动组织者有权修改活动详情或执行签到操作。



**防重复注册:** 智能合约层强制限制每位用户每个活动仅能注册一次。



**数据完整性:** 关键业务逻辑（票务、资金）全链上执行，不可篡改。



**连接保活:** 后端 WebSocket 实现心跳检测与自动重连，防止事件丢失。

# Q & A

HackChain: Build on Monad

---



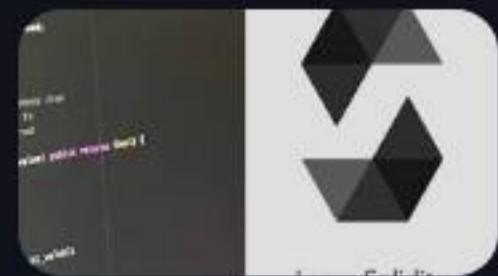
[github.com/zhiming817/HackChain](https://github.com/zhiming817/HackChain)

# Image Sources



[https://d32myzxfyl12w.cloudfront.net/assets/images/article\\_images/bb5cbccab13a4a4e651a942f9abfee83b26e8ac9.webp?1702474556](https://d32myzxfyl12w.cloudfront.net/assets/images/article_images/bb5cbccab13a4a4e651a942f9abfee83b26e8ac9.webp?1702474556)

Source: [mlsdev.com](https://mlsdev.com)



[https://miro.medium.com/1\\*\\_KbmufKq3eXr8iX2YQXB1g.jpeg](https://miro.medium.com/1*_KbmufKq3eXr8iX2YQXB1g.jpeg)

Source: [medium.com](https://medium.com)



[https://media.qrtiger.com/blog/2023/07/qr-code-in-ticketsjpg\\_800.webp](https://media.qrtiger.com/blog/2023/07/qr-code-in-ticketsjpg_800.webp)

Source: [www.qrcode-tiger.com](https://www.qrcode-tiger.com)



[https://img.freepik.com/free-vector/gradient-connection-background\\_23-2150518080.jpg](https://img.freepik.com/free-vector/gradient-connection-background_23-2150518080.jpg)

Source: [www.freepik.com](https://www.freepik.com)