NFT-MARTKETPLACE

技术栈

智能合约开发:

- Solidity: 用于编写智能合约的高级编程语言，是以太坊智能合约的标准语言。
- Hardhat或Truffle: 用于智能合约的编译、测试、部署和调试的开发框架。

前端开发:

- React.js: 用于构建用户界面的JavaScript库，常配合Redux用于状态管理。
- web3.js或ethers.js: 连接前端应用和以太坊区块链，允许与智能合约交互。
- Bootstrap或Material-UI: 提供UI组件库，加速前端开发。

后端开发 (假设项目包含后端服务):

- Java/Spring Boot: 作为后端服务的开发语言和框架，处理非区块链逻辑，如用户管理。
- Web3j: Java库，用于与以太坊区块链交互。
- PostgreSQL或MongoDB: 存储用户数据和NFT元数据的数据库系统。
- RabbitMQ/Kafka: 消息队列，处理前端、后端和智能合约之间的异步通信。

DevOps:

- Docker: 容器化应用，简化部署和运行环境的配置。
- GitHub Actions 或 Jenkins: 自动化CI/CD流程。

架构

该项目可能采用分层架构，包括：

1. 智能合约层：负责处理NFT的铸造、转移、购买和销售等核心逻辑。
2. 后端服务层：处理应用的业务逻辑，如用户账户管理、通知服务和数据分析。
3. 前端展示层：用户与应用交互的界面，提供浏览、发布、购买NFT等功能。
4. 区块链网络层：以太坊区块链网络，存储智能合约和执行交易。
5. 数据存储层：用于存储应用数据（用户信息、交易记录等）和NFT元数据。

需求

1. NFT铸造和展示：用户能够创建（铸造）新的NFT，并在市场上展示。
2. NFT购买和销售：用户可以购买市场上的NFT，或将自己的NFT列入市场销售。

3. 拍卖和报价：支持拍卖模式，允许用户对NFT出价。
4. 钱包集成：集成以太坊钱包（如MetaMask），使用户能够进行交易。
5. 用户账户系统：注册、登录、用户资料管理等功能。
6. 搜索和筛选：允许用户基于NFT的种类、价格等条件搜索和筛选。
7. 交易历史和统计：显示用户的交易历史和市场统计数据。

该项目的实现需要跨多个技术栈和领域的深入知识，从智能合约的开发和安全性保障，到用户友好的前端界面设计，再到后端服务的稳定和高效运行。此外，还需要考虑区块链应用的特殊性，如交易确认时间、Gas费用优化等。

📌 **按照以下任务实现该大作业（ 两周 ）**

任务一：项目初始化和智能合约开发环境设置

目标：设置项目框架和智能合约开发环境。

步骤：

1. 克隆仓库并熟悉项目结构。
2. 安装Node.js和npm。
3. 安装Truffle或Hardhat作为智能合约开发和测试框架。
4. 初始化项目（如果仓库中未提供初始化配置）。

任务二：开发NFT智能合约

目标：开发ERC-721或ERC-1155标准的NFT智能合约。

步骤：

1. 定义NFT的属性和元数据结构。
2. 实现NFT的铸造（Minting）功能。
3. 实现NFT的转移功能。
4. 编译和测试智能合约。

任务三：开发市场智能合约

目标：开发处理NFT买卖逻辑的市场智能合约。

步骤：

1. 实现列出NFT销售和拍卖的功能。
2. 实现购买和出价功能。

3. 实现撤销和结束拍卖的功能。
4. 编译和测试智能合约。

任务四：智能合约部署

目标：将开发的智能合约部署到测试网络。

步骤：

1. 使用Ganache创建本地以太坊测试网络。
2. 配置Truffle或Hardhat进行部署。
3. 部署NFT和市场智能合约到测试网络。
4. 验证合约是否正常部署和运行。

任务五（前端）：前端开发基础

目标：搭建和开发NFT市场的前端界面基础。

步骤：

1. 选择前端框架（如React或Vue）。
2. 创建页面结构，包括首页、市场浏览、NFT详情等。
3. 实现与以太坊钱包（如MetaMask）的连接。

任务六（前端）：前端与智能合约交互

目标：实现前端界面与智能合约的交互。

步骤：

1. 使用web3.js或ethers.js与部署的智能合约交互。
2. 实现NFT的展示、铸造和购买等功能。
3. 实现市场的NFT列表展示和购买逻辑。

任务七（后端）：开发后端服务

目标：构建后端服务来支持NFT市场的额外功能，如用户账户管理、市场活动统计、邮件通知等。

技术栈：Java、Spring Boot、数据库（如PostgreSQL或MongoDB）、Web3j（与以太坊智能合约交互的Java库）

步骤：

1. 搭建基础后端框架：
    - 使用Spring Initializr初始化一个Spring Boot项目。

- 添加必要的依赖，如 `spring-boot-starter-web` 用于REST API开发， `spring-boot-starter-data-jpa` 用于数据库交互， `web3j-spring-boot-starter` 用于与以太坊区块链交互。

2. 实现用户管理：
   - 设计用户模型，并使用JPA实体来表示。
   - 实现用户注册、登录、资料编辑等基本功能。
   - 考虑使用Spring Security添加认证和授权机制。
3. 市场活动和统计：
   - 实现一个服务，用于跟踪和记录市场上的活动，如NFT的铸造、购买和销售记录。
   - 使用Web3j订阅智能合约事件，将相关数据保存到数据库。
   - 开发REST API，提供市场统计信息，如交易量和活跃用户等。
4. 邮件通知服务：
   - 实现一个简单的邮件通知服务，当用户的NFT被购买或他们的出价被接受时发送通知。
   - 使用Spring Mail集成邮件发送功能。
5. 测试：
   - 编写单元测试和集成测试，确保后端服务的稳定性和可靠性。
   - 使用Postman或Swagger测试REST API的功能。
6. 部署：
   - 配置项目以在容器化环境（如Docker）中运行。
   - 考虑将服务部署到云服务器，如AWS或Heroku。

进阶：

- 实现更复杂的功能，如用户钱包管理、NFT藏品展示等。
- 集成第三方服务，如IPFS用于存储NFT元数据。

任务八：测试和优化

目标：对整个NFT市场项目进行测试和优化。

步骤：

1. 对智能合约进行单元测试和集成测试。
2. 在前端进行用户界面和用户体验测试。
3. 识别和修复可能的安全漏洞。
4. 优化交易的Gas成本和前端加载速度。

项目地址

https://github.com/TechPlanB/NFT-MARKETPLACE