# Uto OS项目设计文档

## 1. 概述

### 1.1 目的

本文档详细描述了UTO网络与Uto OS项目的系统设计，包含架构设计、模块划分、数据模型、接口定义、技术选型等，旨在为开发团队提供清晰的指导和参考。

### 1.2 范围

本文档涵盖系统的所有主要功能模块，针对每个模块的设计进行详细描述，包括前端、后端、区块链节点管理、钱包管理、安全性保障等。

### 1.3 术语

* **L1节点**：处理区块链主网事务的节点。
* **L2节点**：提供扩展功能，如数据存储、智能缓存的节点。
* **L3节点**：处理高性能计算和数据处理的节点。

## 2. 系统架构

### 2.1 总体架构

UTO网络与Uto OS项目采用分层架构，主要分为以下几层：

* **用户层**：包括前端用户界面和API接口。
* **服务层**：处理业务逻辑，包含钱包管理、节点管理、授权管理等功能。
* **区块链层**：基于以太坊源码实现的区块链网络，负责账本维护、交易处理、智能合约执行等。
* **数据层**：存储系统数据，包括交易数据、用户数据、节点状态等。

### 2.2 组件架构

系统由多个核心组件组成，每个组件负责特定的功能：

* **前端组件**：负责UI展示和用户交互，使用React或Vue.js构建。
* **后端服务**：通过微服务架构实现，使用Spring Boot或Node.js构建。
* **区块链节点**：使用以太坊源码改造，包含L1、L2、L3节点。
* **数据库**：使用MySQL、PostgreSQL存储用户和交易数据，使用Redis进行缓存。

## 3. 模块设计

### 3.1 区块链浏览器模块

#### 3.1.1 功能描述

提供钱包和区块查询、网络统计与监控功能，支持用户查看交易历史、余额变化、区块详情等。

#### 3.1.2 设计细节

* **UI层**：提供查询表单、结果展示页面。
* **服务层**：提供区块链数据查询API，集成到区块链节点中，通过RPC接口获取数据。
* **数据层**：利用索引优化查询效率，存储历史查询记录以供用户快速访问。

### 3.2 节点管理模块

#### 3.2.1 功能描述

管理L1、L2、L3节点的运行状态、性能指标、数据分发等。

#### 3.2.2 设计细节

* **监控模块**：每个节点定期上报状态数据，使用Prometheus进行收集和监控。
* **配置模块**：管理员可以通过UI界面配置节点参数，配置文件存储在ETCD中。
* **通信模块**：节点之间通过gRPC协议进行通信，确保高效数据传输。

### 3.3 钱包管理模块

#### 3.3.1 功能描述

支持多资产管理、交易记录查询和安全性保障。

#### 3.3.2 设计细节

* **加密模块**：使用AES-256加密保护用户私钥，结合多重签名机制增加安全性。
* **交易模块**：集成以太坊钱包API，支持ERC-20代币的转账和查询。
* **备份与恢复**：用户可通过助记词恢复钱包，采用分布式存储方案确保数据安全。

### 3.4 授权管理模块

#### 3.4.1 功能描述

允许用户授权第三方应用访问账户数据，并自定义权限范围。

#### 3.4.2 设计细节

* **OAuth2.0认证**：使用OAuth2.0标准进行第三方授权认证。
* **权限管理**：用户在授权时选择权限范围，系统将权限配置存储在数据库中，并在每次调用时进行权限检查。
* **日志记录**：记录每一次授权和访问操作，用户可以查询授权历史。

### 3.5 提现功能模块

#### 3.5.1 功能描述

支持用户将Uto代币兑换成人民币，并通过智能合约进行代币燃烧与黄金兑换。

#### 3.5.2 设计细节

* **兑换合约**：在智能合约中实现代币兑换与燃烧逻辑，保证透明性。
* **支付网关**：集成第三方支付网关，将兑换后的人民币打款到用户账户。
* **交易确认**：系统确认区块链上的兑换交易后，触发人民币支付流程。

### 3.6 资金证明查询模块

#### 3.6.1 功能描述

实时展示储备资金的证明情况，包括总量及资产分配。

#### 3.6.2 设计细节

* **智能合约集成**：通过调用智能合约查询储备资金总量。
* **数据展示**：前端通过图表展示各类资产的分配比例和变动情况。
* **历史记录**：记录每次资金查询操作，用户可以查看历史查询记录。

### 3.7 网络协议支持模块

#### 3.7.1 功能描述

支持IPV4和IPV6双协议，并智能管理流量、配置带宽限制。

#### 3.7.2 设计细节

* **协议支持**：通过配置Nginx或HAProxy实现IPV4/IPV6双协议支持。
* **带宽管理**：使用Linux的TC（Traffic Control）工具精确配置带宽限制。
* **流量监控**：集成流量监控工具，实时监控并调整流量分配。

## 4. 数据模型

### 4.1 数据库设计

#### 4.1.1 用户表（Users）

* **字段**：UserID, UserName, PasswordHash, Email, CreatedAt, UpdatedAt

#### 4.1.2 钱包表（Wallets）

* **字段**：WalletID, UserID, Address, Balance, CreatedAt, UpdatedAt

#### 4.1.3 交易表（Transactions）

* **字段**：TransactionID, WalletID, Amount, TxHash, Status, CreatedAt

#### 4.1.4 授权表（Authorizations）

* **字段**：AuthorizationID, UserID, AppID, Scopes, CreatedAt, ExpiresAt

### 4.2 缓存设计

* **Redis**：用于存储短期的数据，如用户会话、临时授权信息等。
* **索引优化**：对高频查询的字段进行索引，提升查询性能。

## 5. 接口设计

### 5.1 RESTful API

#### 5.1.1 区块链查询API

* **GET /api/block/{blockNumber}**：查询指定区块的详细信息。
* **GET /api/wallet/{address}/transactions**：查询指定钱包的交易记录。

#### 5.1.2 节点管理API

* **GET /api/nodes/L1/status**：获取L1节点的运行状态。
* **POST /api/nodes/L2/config**：更新L2节点的配置。

### 5.2 WebSocket API

* **/ws/notifications**：实时推送系统通知，如交易确认、节点状态变化等。

## 6. 技术选型

### 6.1 后端技术

* **编程语言**：Java (Spring Boot), Node.js
* **数据库**：MySQL, PostgreSQL, Redis
* **区块链平台**：基于以太坊源码改造

### 6.2 前端技术

* **框架**：React.js 或 Vue.js
* **状态管理**：Redux 或 Vuex
* **UI组件库**：Ant Design 或 Element UI

### 6.3 安全技术

* **加密算法**：AES-256, RSA
* **身份认证**：OAuth2.0, JWT

## 7. 安全性设计

### 7.1 数据加密

* **静态数据加密**：存储在数据库中的敏感数据使用AES-256加密。
* **传输数据加密**：所有数据传输通过HTTPS和TLS加密。

### 7.2 多重签名

* 关键操作（如大额提现）采用多重签名机制，增加安全性。

### 7.3 安全审计

* 定期进行第三方安全审计，确保系统和智能合约的安全。

## 8. 部署设计

### 8.1 部署架构

* **Kubernetes**：使用Kubernetes

进行容器化部署，支持弹性伸缩。

* **CI/CD**：集成Jenkins或GitLab CI进行持续集成和交付。

### 8.2 日志与监控

* **日志管理**：集成ELK（Elasticsearch, Logstash, Kibana）进行日志收集和分析。
* **监控报警**：使用Prometheus和Grafana进行系统监控，配置自动报警机制。

## 9. 测试计划

### 9.1 单元测试

* 每个模块的核心功能都需编写单元测试，覆盖率要求90%以上。

### 9.2 集成测试

* 测试各模块之间的交互，确保系统整体功能正常运行。

### 9.3 性能测试

* 使用JMeter或Locust对系统进行压力测试，评估系统在高并发情况下的表现。

## 10. 未来扩展

### 10.1 支持更多区块链平台

* 考虑未来集成其他主流区块链平台，如Polkadot, Solana。

### 10.2 增加去中心化交易所（DEX）功能

* 支持在UTO网络上进行去中心化交易，增加系统的功能性。

### 10.3 引入人工智能分析模块

* 利用AI对交易数据进行分析，提供用户行为预测和市场趋势分析。