RustChain_RustCamp_Project Description

1. 项目背景

“RustChain”项目源于 RustCamp 学习活动，旨在通过一个轻量级区块链平台的开发实践，加深对区块链原理、加密算法、数据存储和分布式网络通信的理解。该项目不仅适合技术爱好者和学生进行学习实践，也为希望构建定制化区块链解决方案的开发者提供了一个基础框架。

2. 产品概述

“RustChain”是一款基于 Rust 语言开发的轻量级区块链平台，核心目标是实现一个安全、透明、易于定制与扩展的分布式账本系统。平台实现了区块构造与验证、交易处理、UTXO 模型、钱包管理以及 P2P 网络通讯等基本功能，并通过命令行（CLI）工具提供便捷的操作接口，既适用于区块链技术的学习实践，也可以作为进一步研发定制化解决方案的基础产品。

3. 核心功能与特点

3.1 区块链核心

- **区块结构与 PoW 共识：**
  - 实现了区块数据结构，包括时间戳、交易列表、前一区块哈希、当前区块哈希、随机数（nonce）和区块高度。
  - 利用 SHA256 算法和 Merkle 树构造区块数据摘要，通过工作量证明（Proof-of-Work）机制确保数据不可篡改。
- **链上数据存储：**
  - 采用 sled 数据库对区块数据进行持久化存储，通过键值对方式保存链顶信息（LAST）和各区块数据。

3.2 交易与 UTXO 模型

- **交易构造与签名验证：**
  - 支持普通交易与 coinbase 交易，使用 UTXO 模型确保账户余额与交易输入输出的一致性。
  - 基于 ed25519 算法实现交易签名和验证，防止恶意篡改。
- **UTXO 管理：**
  - 提供查找未花费输出、重建 UTXO 集合和更新机制，确保区块链中资产状态的实时性与准确性。

3.3 钱包管理

- **密钥与地址生成：**
  - 利用 ed25519 生成密钥对，并通过双重哈希（SHA256 后跟 Ripemd160）结合 bitcoincash_addr 库生成钱包地址。
  - 支持钱包的创建、存储、检索及地址管理，保证用户身份与资产安全。

3.4 P2P 网络与节点通信

- **节点发现与消息同步：**
  - 内置 TCP Socket 服务器，实现节点间的版本、区块、交易、地址等信息的广播与请求。
  - 通过自定义消息格式（固定长度命令码加数据载荷）确保数据在网络中的高效传输和解析。

3.5 CLI 命令行接口

- **功能全面：**
  - 提供创建区块链、钱包管理、余额查询、发送交易、打印区块链以及启动节点和矿工等多项命令。
  - 命令行工具便于开发者快速体验区块链各个功能模块，加速调试和学习进程。

4. 系统架构

“RustChain”采用模块化设计，各模块之间职责分明，主要包括：

- **Block 模块：** 实现区块数据结构、哈希计算及 PoW 共识机制。
- **Blockchain 模块：** 负责区块链状态管理、区块迭代、链上数据存储与查询。
- **Transaction 模块：** 定义交易结构、交易签名及验证逻辑。
- **UTXOSet 模块：** 管理未花费交易输出集合，提供余额查询及更新功能。
- **Wallets 模块：** 管理钱包生成、存储和地址管理，确保资产安全。
- **Server 模块：** 构建 P2P 网络，提供节点间通信、数据广播与同步机制。
- **Cli 模块：** 提供命令行接口，支持用户与系统交互，执行链上操作。

5. CLI 功能

5.1 使用说明

CLI 是本项目的核心交互入口，用户可以通过命令行执行以下操作：

主要命令

- printchain
  - **功能：** 打印当前区块链上所有区块的信息。
  - **示例：**

```
D:\rust_camp_project\rust_camp_project_blockchain>cargo run printchain
    Finished `dev` profile [unoptimized + debuginfo] target(s) in 0.50s
     Running `target\debug\rust_camp_project_blockchain.exe printchain`
Block {
    timestamp: 1741434797594,
    transactions: [
        Transaction {
            id: "4f9b81f16de57b390f685600dfd2c3af2c510f43c8c09056949917e73672d4a7",
            vin: [
                TXInput {
                    txid: "",
                    vout: -1,
                    signature: [],
                    pub_key: [
                        114,
                    ],
                },
            ],
        },
    ],
    prev_block_hash: "0000ba4f1ac5546fb51d4e391c89c0406f008a746ef3db066168f04d23d58d68",
    hash: "0000bb1e53e56549da92d5aafd77a04cf40ad2a363cfe3e7b17b36aa77325d36",
    nonce: 2347,
    height: 1,
}
```

- createwallet
  - **功能：** 创建一个新钱包，并输出生成的钱包地址。
  - **示例：**

```
D:\rust_camp_project\rust_camp_project_blockchain>cargo run createwallet
    Finished `dev` profile [unoptimized + debuginfo] target(s) in 0.39s
     Running `target\debug\rust_camp_project_blockchain.exe createwallet`
address: 3E8C9i5stXeofdYWD9zNhzMtBr2xfcZF8N
```

- listaddresses
  - **功能：** 列出所有已创建的钱包地址。
  - **示例：**

```
D:\rust_camp_project\rust_camp_project_blockchain>cargo run listaddresses
    Finished `dev` profile [unoptimized + debuginfo] target(s) in 0.33s
     Running `target\debug\rust_camp_project_blockchain.exe listaddresses`
addresses:
3CalSWUntNPsM1KbUavtVru2rzwaukB4ae
3CAoF3YN22pTtCmPEJDWTcuB1HocoZ5vdU
3E8C9i5stXeofdYWD9zNhzMtBr2xfcZF8N
3Mc6DzPe46KmBtSF89LZaWPAYXpunDMFSs
```

- reindex
  - **功能：** 重建 UTXO 集合，并输出当前 UTXO 集合中交易的数量。
  - **示例：**

```
D:\rust_camp_project\rust_camp_project_blockchain>cargo run reindex
    Finished `dev` profile [unoptimized + debuginfo] target(s) in 0.33s
     Running `target\debug\rust_camp_project_blockchain.exe reindex`
Done! There are 2 transactions in the UTXO set.
```

- getbalance <ADDRESS>
  - **功能：** 根据指定钱包地址查询余额。
  - **示例：**

```
D:\rust_camp_project\rust_camp_project_blockchain>cargo run getbalance 3CAoF3YN22pTtCmPEJDWTcuB1HocoZ5vdU
    Finished `dev` profile [unoptimized + debuginfo] target(s) in 0.32s
     Running `target\debug\rust_camp_project_blockchain.exe getbalance 3CAoF3YN22pTtCmPEJDWTcuB1HocoZ5vdU`
Balance: 10
```

- **create <ADDRESS>**
  - **功能：** 创建新的区块链，并将创世区块奖励发送到指定地址。
  - **示例：**

```
D:\rust_camp_project\rust_camp_project_blockchain>cargo run create 3Mc6DzPe46KmBtSF89LZaWPAYXpunDMFSs
    Finished `dev` profile [unoptimized + debuginfo] target(s) in 0.32s
     Running `target\debug\rust_camp_project_blockchain.exe create 3Mc6DzPe46KmBtSF89LZaWPAYXpunDMFSs`
create blockchain
```

- **send <FROM> <TO> <AMOUNT> [--mine]**
  - **功能：** 发起交易，将指定金额从源地址转移至目标地址；可选的 --mine 参数表示立即挖矿确认交易。
  - **示例：**

```
D:\rust_camp_project\rust_camp_project_blockchain>cargo run send 3Mc6DzPe46KmBtSF89LZaWPAYXpunDMFSs 3CAoF3YN22pTtCmPEJDWTcuB1HocoZ5vdU 5
    Finished `dev` profile [unoptimized + debuginfo] target(s) in 0.33s
     Running `target\debug\rust_camp_project_blockchain.exe send 3Mc6DzPe46KmBtSF89LZaWPAYXpunDMFSs 3CAoF3YN22pTtCmPEJDWTcuB1HocoZ5vdU 5`
success!
```

- **startnode <PORT>**
  - **功能：** 启动节点服务器，绑定到指定端口，参与 P2P 网络数据同步。
  - **示例：**

```
D:\rust_camp_project\rust_camp_project_blockchain>cargo run startnode 8080
    Finished `dev` profile [unoptimized + debuginfo] target(s) in 0.33s
     Running `target\debug\rust_camp_project_blockchain.exe startnode 8080`
```

```
D:\rust_camp_project\rust_camp_project_blockchain>netstat -an | find "8080"
  TCP    [::1]:8080             [::]:0                 LISTENING
  TCP    [::1]:8080             [::1]:62418            ESTABLISHED
  TCP    [::1]:8080             [::1]:62419            ESTABLISHED
  TCP    [::1]:62418            [::1]:8080             ESTABLISHED
  TCP    [::1]:62419            [::1]:8080             ESTABLISHED
```

- **startminer <PORT> <ADDRESS>**
  - **功能：** 启动矿工节点服务器，指定监听端口和矿工钱包地址，用于自动打包交易挖矿。
  - **示例：**

```
D:\rust_camp_project\rust_camp_project_blockchain>cargo run startminer 8081 3Mc6DzPe46KmBtSF89LZaWPAYXpunDMFSs
    Finished `dev` profile [unoptimized + debuginfo] target(s) in 0.37s
     Running `target\debug\rust_camp_project_blockchain.exe startminer 8081 3Mc6DzPe46KmBtSF89LZaWPAYXpunDMFSs`
```

```
D:\rust_camp_project\rust_camp_project_blockchain>netstat -an | find "8081"
  TCP    [::1]:8081             [::]:0                 LISTENING
```

5.2 CLI 内部实现

- **参数解析：**
  - 采用 clap 库对命令和参数进行解析，保证命令行输入的正确性和提示信息的清晰性。
- **功能调用：**
  - 每个 CLI 命令对应一个或多个函数调用。
  - 通过 CLI 命令，用户可以直接调用区块链模块、钱包模块及网络服务模块

的各项功能，实现链上数据的操作与节点间信息同步。

6. 部署与使用

6.1 开发环境

- **语言：** Rust
- **依赖管理：** 使用 Cargo 进行依赖管理与构建
- **数据库：** sled 嵌入式数据库用于存储区块、UTXO 集合及钱包数据

6.2 构建与运行

- **构建项目：**

```
D:\rust_camp_project\rust_camp_project_blockchain>cargo build --release
   Compiling proc-macro2 v1.0.94
   Compiling unicode-ident v1.0.18
   Compiling windows_x86_64_msvc v0.52.6
   Compiling cfg-if v1.0.0
```

- **运行 CLI：**

```
D:\rust_camp_project\rust_camp_project_blockchain>target\release\rust_camp_project_blockchain.exe createwallet
address: 3EuU7eaJkip8SxX6sJMYhB4GULTqvJD3Rq
```