Solana Wallet Backend

A comprehensive backend for a Solana Web3 wallet. This backend provides wallet management, token operations, staking, transaction history, address book, and network status APIs.

Table of Contents

Features

- Wallet Management: Create/import wallets, encrypt secrets, retrieve balances.
- Token Operations: Send SOL/custom tokens, get wallet tokens with metadata/prices.
- Staking: Stake, unstake, withdraw SOL, view validators and stake accounts.
- Transaction History: Fetch Solana/EVM transaction history, simulate and estimate fees.
- Address Book: Add, fetch, and delete contacts for Solana/EVM chains.
- Network Status: Get real-time network status for Solana and EVM chains.

Architecture

- Node.js/Express backend.
- Raydium API integration for token metadata and prices.

API Endpoints (Summary)

Wallet

- `POST /api/wallet/create` — Create new wallet
- `POST /api/wallet/import/mnemonic` — Import wallet from mnemonic
- `POST /api/wallet/import/privatekey` — Import wallet from private key
- `GET /api/wallet/balance/:address` — Get Solana balance

Token

- `POST /api/token/send-sol` — Send SOL
- `POST /api/token/send-custom` — Send SPL token
- `GET /api/token/wallet-tokens/:address` — Get wallet tokens with metadata/prices

Staking

- `GET /api/staking/validators` — Get Solana validators
- `POST /api/staking/stake` — Stake SOL
- `POST /api/staking/unstake` — Unstake SOL
- `POST /api/staking/withdraw` — Withdraw unstaked SOL
- `GET /api/staking/accounts/:walletAddress` — Get stake accounts

Transactions

- `GET /api/transactions/history/:address` — Solana/EVM transaction history
- `POST /api/transactions/simulate` — Simulate transaction
- `POST /api/transactions/estimate-fee` — Estimate gas/fee

Address Book

- `POST /api/address-book/add` — Add contact
- `GET /api/address-book/:walletAddress` — Get contacts
- `DELETE /api/address-book/:walletAddress/:contactId` — Delete contact

Network Status

- `GET /api/network-status/:chain` — Get network status (Solana/EVM)

Environment Setup

1. Clone the repository:

```
git clone <repo-url>
cd solana_wallet_backend
```

2. Install dependencies:

```
npm install
```

3. Configure environment variables:
   Copy `.env.example` to `.env` and fill in required values (Solana RPC URL, etc).

Running Locally (Server)

1. Start the backend server:

```
npm start
```

   The server runs on `http://localhost:8080` by default.
2. API base path: All endpoints are prefixed with `/api` .

Deploying to Vercel

1. Create a Vercel project:
   Link your GitHub repo to Vercel.
2. Set environment variables:
   In the Vercel dashboard, add all required env vars (see `.env.example`).
3. Configure Vercel settings:
   Set the build command to `npm install` (if not automatic).
   Set the output directory to the project root or as needed.
   Set the start command to `npm start` (for Node.js serverless functions, see Vercel docs).
4. Deploy:
   Push to your main branch; Vercel will auto-deploy.

Project Structure

```
solana_wallet_backend/
├── controllers/ # Express route controllers
├── domain/ # Core business/domain logic (wallet, staking, encryption) ├──
infrastructure/ # Blockchain integrations
├── routes/ # Express route definitions
├── use-cases/ # Service logic for wallet, token, staking, etc ├──
config/ # Configuration and constants
├── index.js # Entry point
├── package.json # Dependencies and scripts
└── ...
```

Firebase

Create a Firebase Project and Download the ServiceAccount Json and replace it with existing One.

Network & Customization
Changing Solana Network

To change the Solana network (mainnet, devnet, etc.), edit the `infrastructure/solana.js` file:

```
 const { Connection } = require("@solana/web3.js");
const { SOLANA_NETWORK_MAINNET, SOLANA_NETWORK_DEVNET } = require("../config/constants");

// Example: switch between devnet and mainnet
const connection = new Connection(SOLANA_NETWORK_DEVNET, connectionConfig);
// Change to:
// const connection = new Connection(SOLANA_NETWORK_MAINNET, connectionConfig);
```

You can also customize:

- RPC URLs: Set your own endpoint in `config/constants.js` .
- Connection options: Adjust `connectionConfig` in `solana.js` for commitment, timeouts, etc.
- Batching: Use or extend the `batchRequestMiddleware` in `solana.js` for advanced request handling.

Contributing

1. Fork the repo and create your branch from `main` .
2. Follow code style and naming conventions.
3. Add tests for new features.
4. Submit a pull request.

License

This project is licensed under the MIT License.