ANDROID MOBILE APPLICATION FOR MANAGING ACCOUNTS with erc20 cryptocurrency

**Teodor Todorov – Student**

Department of Computer Systems and Technologies,

University of Ruse “Angel Kanchev”

Tel.: +359 895377807

E-mail: banskaliqtabg@gmail.com

**Abstract:** The paper reviews how ERC20 wallet works using Web3j library on Android platform. The application needs to be connected to connected to a blockchain network where the node is either remote or on the local machine. The paper consist Web3j documentation and how the components are connected in the application, and how transfers are made.

***Keywords:*** *ERC20, Wallet, Web3j, Android, Blockchain network, Node*

ВЪВЕДЕНИЕ

С увеличаване интереса към криптовалутите и токените се увеличава и броя на хората, които ги ползват. Това налага промени в начина на свързване на клиентското приложени към мрежатам, а именно чрез отдалечен достъп до възел. Сега хора които нямат познания в областта ги ползват благодарение на електронните портфейли. С ERC20 стандарта тези портфейли вече могат да обслужват безброй токени.

Електронният портфейл е приложение служещо за управляване на блокчейн сметки.

ИЗЛОЖЕНИЕ

Функционални изисквания на приложението

Приложението трябва да прави следните неща:

* Създаване и възстановя на портфейл
* Добавяне на токени към портфейла
* Опериране с договори

Създаване и възстановяване на портфейл

Начален прозорец на приложението при първо пускане се състои от два бутона, един засъздаване и един за въстановяне. Създаване се състои от извикване на:

|  |  |  |
| --- | --- | --- |
|  | WalletUtils.generateBip39Wallet(password, destinationDirectory) | (1) |

Паролата се използва за кодиране на портфейла и на BIP-39 семената. Дестинацията е директорията съдържаща портфейла. Метода създава файл, който се отключва с подадената парола и връща обект съдържащ името на файла и мнемоничния код.

Възстановяването се състои във възстановяване на личния ключ чрез мнемоничния код и паролата:

|  |  |  |
| --- | --- | --- |
|  | SHA-256(BIP\_39\_SEED(mnemonic, password)) | (2) |

Когато портфела вечесъществува за да се зареди се изпълнява:

|  |  |  |
| --- | --- | --- |
|  | WalletUtils.loadBip39Credentials(password, mnemonic); | (3) |

**Добавяне на токени към портфейла**

Добавяне и следнето на токен става като се добави адреса на токена в базата данни и се зареди контракта чрез въвеждането му или декодиране от баркод.

|  |  |  |
| --- | --- | --- |
|  | YourSmartContract contract = YourSmartContract.load(  "0x<address>|<ensName>", web3j, credentials, GAS\_PRICE, GAS\_LIMIT); | (4) |
|  |  |  |
|  | Web3j web3 = Web3j.build(new HttpService()); // defaults to http://localhost:8545/ | (5) |

Параметрите в метода за зареждане на договора по метод (4) са: адрес на договора, инстанция на възела от метод (5), идентификация получени от метод (3), цена на газта и лимит на газта. Договара е инстанция на клас разширяващ org.web3j.tx.Contract класа.

**Опериране с договори**

Оперирането с договори става с методите имплементирани в класа. Те са обвивка на JSON-RPC комуникация между клиента и възела и изключват нуждата от кодиране и декодиране на обръщанията към функциите с ABI(Abstract Binary Interface).

ERC20 стандарта има следните методи и събития:

* totalSupply – общ брой предлагани токени,
* balanceOf(address \_owner) const returns (uint256 balance) – взема баланса на подадения адрес,
* transfer(address \_to, uint256 \_value) returns (bool success) – изпраща определения брой токени към избрания адрес,
* transferFrom(address \_from, address \_to, uint256 \_value) returns (bool success) – изпраща определения брой токени от единия адрес към другия,
* approve(address \_spender, uint256 \_value) returns (bool success) – разрешаване на някой да ползва средства от сметка до определена стойност,
* allowance(address \*\_owner\*, address \*\_spender\*) constant returns (uint256 remaining) – оставащата стойност на която е разрешена да се тегли
* Transfer(address indexed \_from, address indexed \_to, uint256 \_value) – събитие което се изпълнява, когато токените са изпратени
* Approval(address indexed \_owner, address indexed \_spender, uint256 \_value) – събитие което се изпълнява, когато се извика approve.

Извършването на транзакция напрмер се свежда до създаване на инстанция на org.web3j.abi.datatypes.Function класа. Където първи параметър е името на функцията, следван от параметрите на виканата функция, и като последен параметър е връщания тип.

|  |  |  |
| --- | --- | --- |
|  | Function function = new Function("balanceOf",  Arrays.<Type>asList(\_owner),  Arrays.<TypeReference<?>>asList(new TypeReference<Uint256>() {})); | (6) |

ИЗВОДИ

Изграждането на портфейла с помощта на Web3j е сравнително лесно. Трудната част е да се гарантира сигурността на частния ключ, затова се използва порфейл на който личния ключ се получава от кодиране на семената получени от кодирането на мнемоничната фраз с Sha256, а семената са получени от кодиране на мнемоничната фраза с паролата. С мнемоничната фраза портфейла може да се зареди без да се пази файла му на устройството.

REFERENCES

Web3j documentation: https://docs.web3j.io/abi.html (Accessed on 26.04.2018).