

Computer Project I

Project 1 Bitcoin Wallet

Muhammed YILMAZ – 150150149

Cihan GÜZEL – 150140302

16.11.2021

1. Introduction

Bitcoin wallets allow the crypto currency to be sent and retrieved by adhering to the Bitcoin protocol. These wallets are such programs that enable people with little to no technological expertise to connect without barriers with the Bitcoin blockchain to transact value globally.

In the same way as cash is kept in an actual pocket, Bitcoin is not contained in a wallet. On the blockchain, you don't necessarily own the individual bitcoin that you carry. Instead, you own the key combination that helps you to control and transfer the bitcoin through it. Once you store your keys securely, the Bitcoin you buy is safe. At least one linked private key and a single public key are kept by wallets. In a nutshell, the idea of a Bitcoin wallet is generated by the combination of the private key or keys and the public key.

There are different types of wallets but we implemented a web wallet in this project. Web wallets allow instant Bitcoin transactions. You deposit your coins into the online wallet of service providers if you use a web wallet. Also, they have simple-to-use, easy applications for sending, receiving and saving small quantities of bitcoin and are available with an Internet connection anywhere.

2. Software Tools and Technologies

- Python

We decided to use python for our programming language because it has an extensive library that contains public open source libraries for different uses, such as web browsers and databases that we benefit a lot. Also, when coding, we have the ability to think clearly, which also makes the code easier to manage.

- Flask Framework

Flask is a WSGI web application platform that is flexible. It is developed to ensure that fast and easy to get started, with the capacity to scale up to complicated applications. Thus, we benefit some common flask libraries to operate with database, doing database and login operations.

- Jinja

Jinja is a modern and designer-friendly templating language for Python and we used this to make better, flexible and more dynamic html pages.

- **Pony ORM**
Pony is an object-relational SQL mapper that we used to help us think our database and records more abstractly.
- **HTML and CSS**
We created our web pages using HTML and CSS because HTML is the default document markup language that is meant to be viewed in a web browser and the technology such as CSS can help sustain it and makes it our web pages look better.
- **Bit Library**
This library is an open source public Python library that enables users to use most common Bitcoin operations and create a connection between blockchain network.

➔ Details

Bit communicates with the blockchain using most common, trusted third-party APIs. These API's listed as above:

- i. BlockchairAPI
- ii. BlockstreamAPI
- iii. BitcoreAPI
- iv. SmartbitAPI
- v. BlockchainAPI
- vi. Bit library gathers these API's services under a NetworkAPI and uses to operate on blockchain network. Each API class has same field with different URLs to get information through the blockchain network. For example, BlockchainAPI class has fields like below:

```
1. class BlockchainAPI:
2.     ENDPOINT = 'https://blockchain.info/'
3.     ADDRESS_API = ENDPOINT + 'address/{}?format=json'
4.     UNSPENT_API = ENDPOINT + 'unspent'
5.     TX_PUSH_API = ENDPOINT + 'pushtx'
6.     TX_API = ENDPOINT + 'rawtx/'
7.     TX_PUSH_PARAM = 'tx'
```

Figure 1. BlockChainAPI class fields containing URLs

Each of API classes same fields with changing URLs to connect to the blockchain network. As an example, `ADDRESS_API` is an URL used to get balance of the given wallet address by the help of `get` method under `get_balance` class function.

```

1. @classmethod
2.     def get_balance(cls, address):
3.         r = requests.get(cls.ADDRESS_API.format(address) + '&limit=0', timeout=DEFAULT
4.             T_TIMEOUT)
5.         if r.status_code != 200: # pragma: no cover
6.             raise ConnectionError
7.         return r.json()['final_balance']

```

Figure 2. `get_balance` method under `BlockChainAPI` class fields containing URLs

This library mainly used for 3 functionalities that includes;

1. Creation of Private and Public Keys for the user.

Bit library contains a function to create a unique personal private key for each user. When a private key is created, a public key and address to that public key created automatically.

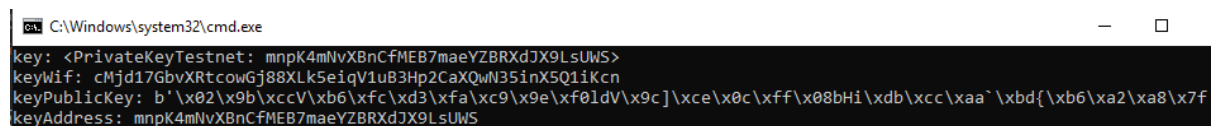
We can create a private key for the user, can be accessed to its public address and this private key can be converted to WIF format as shown below

```

1. key = PrivateKey()
2. address = key.address()
3. keyWif = key.to_wif()

```

Figure 3. Common methods in Bit library



```

C:\Windows\system32\cmd.exe
key: <PrivateKeyTestnet: mnpK4mNvXBnCfMEB7maeYZBRXdJX9LsUWS>
keyWif: cMjd17GbvXRtcowGj88XLk5eiqV1uB3Hp2CaXQwN35inX5Q1iKcn
keyPublicKey: b'\x02\x9b\xccV\xb6\xfc\xd3\xfa\xc9\x9e\xf0ldV\x9c]\xce\x0c\xff\x08bHi\xdb\xcc\xaa'\xbd{\xb6\xa2\xa8\x7f
keyAddress: mnpK4mNvXBnCfMEB7maeYZBRXdJX9LsUWS

```

Figure 4. Sample outputs of common methods in Bit library

2. Getting balance of the wallet address

Bit library has `get_balance` function to see balance related to that address by connecting to NetworkAPI as shown in Figure 1 and 2.

3. Sending bitcoin to and address

Bit library uses `send()` function to create a signed P2PKH transaction and attempts to broadcast it on the blockchain. This accepts the same arguments as `create_transaction()`. We can pass address, quantity and currency to this function and it creates a transaction. If we want we can pass message and determine fee value but optional is not to determine fee value because by default Bit will poll `<https://bitcoinfees.earn.com>` and use a fee that will allow our transaction to be confirmed as soon as possible. Here is the list of input parameters we can adjust if we want in the source code:

```

1. def create_transaction(
2.     self,
3.     outputs,

```

```

4.         fee=None,
5.         absolute_fee=False,
6.         leftover=None,
7.         combine=True,
8.         message=None,
9.         unspents=None,
10.        message_is_hex=False,
11.        replace_by_fee=False
12.    ):

```

After determining transaction parameters, *create_transaction()* function will try to fetch list of all available unspent transaction outputs as show below:

```

1.  try:
2.      unspents = unspents or self.get_unspents()
3.  except ConnectionError:
4.      raise ConnectionError('All APIs are unreachable. Please provide the unspents to spend from directly.')

```

To get unspent object list it calls the *get_unspent()* function. Unspent object consists of variables as shown below:

```

1. unspents, outputs = sanitize_tx_data(
2.     unspents,
3.     outputs,
4.     fee or get_fee_cached(),
5.     leftover or return_address,
6.     combine=combine,
7.     message=message,
8.     absolute_fee=absolute_fee,
9.     version=self.version,
10.    message_is_hex=message_is_hex,
11.    replace_by_fee=replace_by_fee
12. )

```

As we mentioned, *get_unspent()* function is a class function of NetworkAPI and uses *GET_UNSPENT_MAIN* URL of each API's with get method by passing the wallet address as shown below:

```

1. @classmethod
2.  def get_unspent(cls, address):
3.      """Gets all unspent transaction outputs belonging to an address.
4.      :param address: The address in question.
5.      :type address: ``str``
6.      :raises ConnectionError: If all API services fail.
7.      :rtype: ``list`` of :class:`~bit.network.meta.Unspent`
8.      """
9.
10.     for api_call in cls.GET_UNSPENT_MAIN:
11.         try:
12.             return api_call(address)
13.         except cls.IGNORED_ERRORS:
14.             pass
15.
16.     raise ConnectionError('All APIs are unreachable.')

```

After all of that, `send()` function uses broadcasts the transaction in the network by using the classmethod of `NetworkAPI` by sending a raw transaction as shown below:

```
1. def broadcast_tx(cls, tx_hex): # pragma: no cover
2.     """Broadcasts a transaction to the blockchain.
3.     :param tx_hex: A signed transaction in hex form.
4.     :type tx_hex: ``str``
5.     :raises ConnectionError: If all API services fail.
6.     """
7.     success = None
8.
9.     for api_call in cls.BROADCAST_TX_MAIN:
10.         try:
11.             success = api_call(tx_hex)
12.             if not success:
13.                 continue
14.             return
15.         except cls.IGNORED_ERRORS:
16.             pass
17.
18.     if success is False:
19.         raise ConnectionError('Transaction broadcast failed, or Unspents were al
20. ready used.')
21.     raise ConnectionError('All APIs are unreachable.')
```

To sum up, here is an example showing us to how we use this `send()` function to send a bitcoin:

```
1. >>> from bit import Key
2. >>>
3. >>> my_key = Key(...)
4. >>> my_key.get_balance('usd')
5. '12.51'
6. >>>
7. >>> # Let's donate!
8. >>> outputs = [
9. >>>     # Wikileaks
10. >>>     ('1HB5XMLmzFVj8ALj6mfBsbfRoD4miY36v', 0.0035, 'btc'),
11. >>>     # Internet Archive
12. >>>     ('1Archive1n2C579dMsAu3iC6tWzuQJz8dN', 190, 'jpy'),
13. >>>     # The Pirate Bay
14. >>>     ('129TQVAroeehD9fZpzK51NdZGQT4TqifbG', 3, 'eur'),
15. >>>     # xkcd
16. >>>     ('14Tr4HaKkKuC1Lmpr2YMAuYVZRWqAdRTcr', 2.5, 'cad')
17. >>> ]
18. >>>
19. >>> my_key.send(outputs)
20. '9f59f5c6757ec46fdc7440acbeb3920e614c8d1d247ac174eb6781b832710c1c'
```

3. Functionalities

We added some extra features of our application. Test user's info we implemented during development are below:

username: user5
password: user
username: user3
password: use

- **Creating Account**

We built a feature for users to build an account on this website by showing them a Page Register. We take the fields of username, email and authentication of login and password. If there is another user with the same username in the database, we want the user to fill out a form with different details. Using the PrivateKey process, the user's private key is created automatically after successful registration. If the user is a new user, we store this information in the database's site user table.

Register

Username:*

Email:*

Password:*

Password verification:*

© WeeWallet, 2020

- **Login Page**

For users already in our database, we built a login page to access their wallets. Using their username and password that they have already created before, users may login. If all the information entered is right and it is in the database, the user may otherwise access their wallet page linked to that username, before valid input is entered, we ask for this information again.

Login Page

Username:*

Password:*

© WeeWallet, 2020

- **My Wallet Page**

Users will see their wallet address and their balance after successfully signing in or registering with new arrivals. The wallet balance displays the cumulative amount of bitcoin in the current wallet of the customer. We used a temporary internet site that transfers bitcoin over the test net to test our user interface and its features. We show summary of transfers received on this account and also table of recent transfers with some information about those transfers.

WeeWallet
My Wallet Logout

My Wallet

Wallet Address: msmwGRzJPGRUnPHhbpEh9bNIRtJuvNB4DW

Balance: 0.01904666

[Send bitcoin](#)
[Ask bitcoin](#)
[My requests](#)

My transfers

#	Transfer time	Amount	Transfer type	Receiver/Sender wallet address
1	14-11-2020 23:19	0.01	Sent	mkz53mGY6jKYpqpBeUxcRZrBPS9rwPchwD
2	15-11-2020 00:05	0.001	Receipt	msmwGRzJPGRUnPHhbpEh9bNIRtJuvNB4DW
3	15-11-2020 00:06	0.001	Receipt	msmwGRzJPGRUnPHhbpEh9bNIRtJuvNB4DW
4	15-11-2020 00:07	0.001	Receipt	msmwGRzJPGRUnPHhbpEh9bNIRtJuvNB4DW
5	15-11-2020 00:07	0.001	Receipt	msmwGRzJPGRUnPHhbpEh9bNIRtJuvNB4DW
6	15-11-2020 00:16	0.001	Receipt	msmwGRzJPGRUnPHhbpEh9bNIRtJuvNB4DW

- Send Bitcoin

Any consumer with an account can transfer bitcoin by entering the public address of the receiver side and the amount of bitcoin that they want to send.

Send Bitcoin

Receiver address:*

Receiver address

Amount:*

Amount

Send

© WeeWallet, 2020

- Ask/Request Bitcoin

Bitcoins are collected for a specific purpose through bitcoin requests. People who will send bitcoins can send bitcoin directly to this bitcoin request via the address of this bitcoin request. Bitcoin request can be created with ask/request bitcoin page. When creating a bitcoin request, limits can be set, such as the total number of bitcoins to be accepted, the number of transfers to be accepted, and the minimum amount of bitcoin to be sent in one

transfer. In addition, fixed bitcoin-consistent requests can be created for each transfer to be sent.

Request Bitcoin

Amount:*

Minimum transfer amount to be received:*

Maximum amount to be received in total:*

Maximum number of transfers to be received:

*

© WeeWallet, 2020

- **My Bitcoin Requests Page**

The My Bitcoin requests page lists all the bitcoin requests you have made so far.

Bitcoin request details can be viewed with the URL next to each request.

The same address can be shared with the people who will send bitcoin to the bitcoin request. When the person who creates the request enters this address, you can see the details, when others enter, the bitcoin sending page is opened.

[New bitcoin request](#)

#	Creating date	Amount	Min amount	Max amount in total	Max count	Count of received	Total received	Request URL
1	2020-11-14	0.0	0.0	0.0	0	1	0.001	http://127.0.0.1:5000/bitcoin_request/1
3	2020-11-14	0.001	0.0	0.0	0	5	0.005	http://127.0.0.1:5000/bitcoin_request/3

© WeeWallet, 2020

- **Bitcoin request detail page**

On this page, all the details such as the limits of the bitcoin request can be seen.

Details such as the number of transfers made so far, the total amount of bitcoin received are also shown. In addition, all transfers sent to the bitcoin request can be listed.

WeeWallet		My Wallet Logout	
Bitcoin Request		Summary of transfers received	
Id:	1	Count of received transaction:	1
Amount per transfer:	0.0	Total amount of received transactions:	0.001
Minimum amount:	0.0		
Maximum amount in total:	0.0		
Maximum transaction count:	0		
Received transfers			
#	Date:	Amount:	Sender wallet key:
3	2020-11-15 00:06:45.558015	0.001	mkz53mGY6jKYpqpBeUxcRZrBPS9nwPchwD

4. Task Distribution

Muhammed YILMAZ created database design, implemented the Bit library functionalities on the web application and backend software development. Cihan GÜZEL worked on the front-end development and documentation of libraries that we used.

5. Conclusion

In this project, we implemented a bitcoin wallet as a web application with some common functionalities such as login-registration, database operations and sending transaction over a public bitcoin Python library. Before we implemented our application, we did quiet amount of research about the how bitcoin and bitcoin network works and decided which library we will used. We could not download whole bitcoin client due to its huge size so we decided to get the help of public libraries.