byron-docs

Byron Dex

Dec 14, 2021

EXPLORE BYRON DEX

The main goal of this project is to design and implement a decentralized exchange app. DEX DApp allows users to exchange supported tokens with other users. Users can also invest their supported tokens and emit "liquidity orders" to provide a liquidity pool, and therefore obtain commission in the form of exchange fees for doing so.

CHAPTER

ONE

FEATURES

Feature	Description
Faucet	DEX is able to access faucet and draw funds from it
SWAPs	DEX is able to perform an exchange between at least 2 different coins on Testnet
Fees	DEX is provides fee for Liquidity providers and Performers
Liquidity	Users can add liquidity to the DEX and benefit from the fee *(DEX is able to initialize Liquidity "pools" by putting orders of "Liquidity Order" type)*
	Users can withdraw their liquidity from the DEX *(Users can withdraw their liquidity from the DEX by canceling "Liquidity Order" order)*
Rewards	Rewards are automatically added back to DEX, user can withdraw it
	Users can view the all the rewards they earned
Prices	DEX is able to calculate prices based on coins market prices on CEX
Indirect Swap	DEX is able to perform indirect Swaps via other liquidity pools if there's no matching pair between two coins
Matching algorithm	DEX is able to match orders from order book (basic algorithm)

CHAPTER

TWO

OVERVIEW

Our DEX model is based on **Order Book Pattern (OBP)** combined with a **2-phase commit pattern**. The model distinguishes the commit and execution phases. In the committing phase, actors submit swap orders with an expectation of a particular amount of an opposite coin. The execution phase matches orders and performs them. Successful execution locks money in the script for the beneficiary.

2.1 Orders

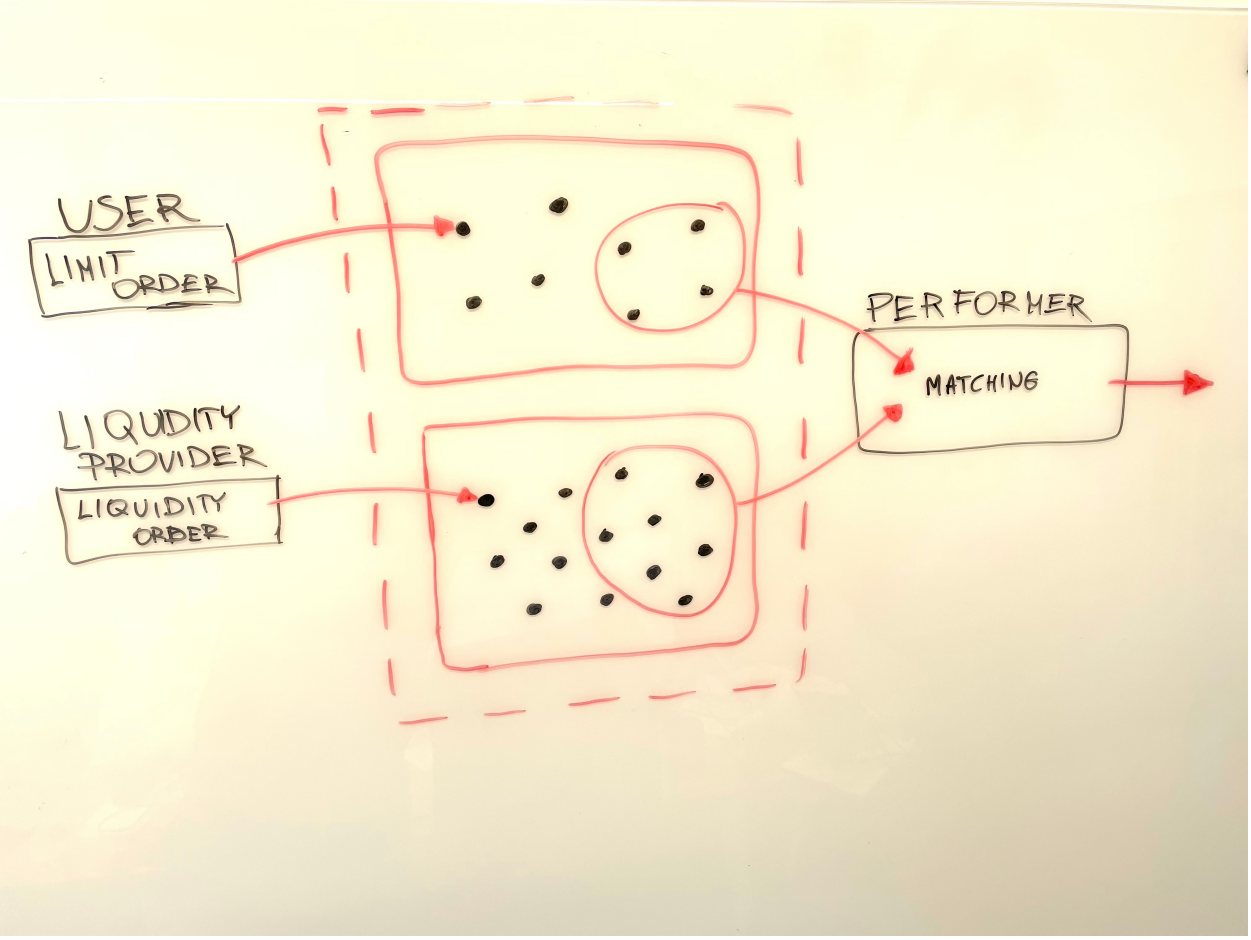
Orders are the core of OBP. Order is represented as UTxO, which contains all information needed to perform a swap on it. There may be different types of orders. For now, we implemented the following:

- Limit order (sell order) - the simplest type of order. It is given by swappers. It contains the tokens that the swapper wants to replace and they can be used only when an appropriate payout is created for the owner of the order (UTxO, which can only be used by that person).
- Liquidity order - prepared with liquidity providers in mind. It contains information about which two tokens (in specific their quantities) will be exchanged between each other, and the amount of the fee. For example: we have a liquidity order between 100A and 200B, with a commission of 1%. In the beginning, we create it as UTxO, and add 100A to it. If someone wants to receive 100A, they will have to create a new liquidity order, this time between 202B (101% of 200) and 100A, and will have to create 202B back. If someone wants to get this 202B, they will have to create a liquidity order between 101A and 202B and put 101A in it. This works until the original owner cancels the order - then he receives all the funds stored in this UTxO. The more swaps were made on this liquidity order, the more profit the person submitting it will receive.

2.2 Actors

We have 3 types of actors:

- **swappers** - people wishing to exchange one token for another. They want the exchange to take place as soon as possible and with the best exchange rate.
- **liquidity providers** - people with funds willing to provide liquidity. They want to earn as much as possible on their investment.
- **performers** - people (or bots) executing orders. They look for orders that match and choose the ones from which they can earn. For example, Alice wants to trade 10A for 9B and Bob wants to trade 10B for 9A. If we take these two orders and execute them in one transaction, Alice gets 9B, Bob gets 9A, and the performer gets 1A and 1B. Performers want to be able to easily analyze existing DEX orders and be able to combine as many orders as possible into one transaction.



2.3 Liquidity Swamps

Liquidity Swamps are a sets of discrete Liquidity Orders, where each order has a slightly different rate. The distribution of subsequent orders can be determined on the basis of the $x * y = k$ curve, but there is nothing to prevent the use of other curves as well. Every Liquidity Order is an independent exchange offer, but all of them together simulate the operation of the classic Liquidity Pool. The Liquidity Provider can add their funds in swamps of liquidity instead of through one or more Liquidity Orders with identical rates. Thus, his funds earn better in an environment of ever-changing cryptocurrency prices.

Imagine a traditional Liquidity Pool. It is some kind of system containing large amounts of two different tokens (A and B) and allowing for pulling them out as long as certain rules are adhered to. For example, we can make it so the product of the quantities of both coins can never decrease, and each swap must additionally draw 0.3% less than it would like (a commission). If we analyze such a pool, it turns out that each consecutive swap of coin A for coin B is less and less profitable. If we were to establish that each swap could move only 100 coin A (and as much B as the math of this pool requires), we could **divide the entire pool into a set of independent exchange offers.**

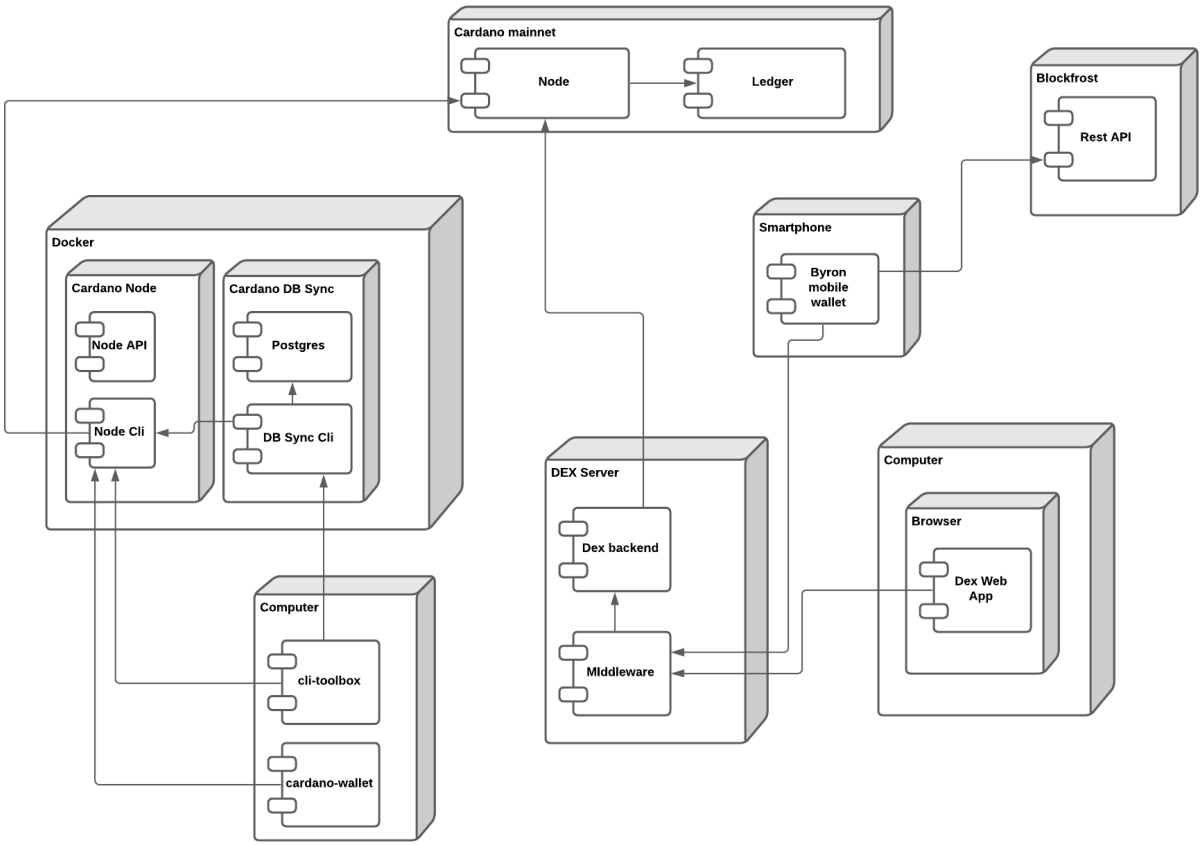
Not every offer is equally profitable - only those on the top will be close to the market rate. However, when the global market price of coins fluctuates, the A to B swap offers will disappear faster than B to A (or vice versa). The situation will stabilize when the most profitable offer remains close to the market price.

CHAPTER

THREE

DEPLOYMENT

3.1 High level deployment



CHAPTER

FOUR

API

4.1 Wallet Funds

Method	**GET**
URL	`http://localhost:8080/cd6f61d0-c503-4e4c-835d-d38a78613066/funds`
Response Code	200

Response body:


```
[
    {
        "amount": 1000000,
        "coin": {
        "tokenName": "A",
        "currencySymbol": "25ae866bd6b617664c054dc97f4d5b3a8cff3fb6ab0354622ada4a11"
        }
    },
    {
        "amount": 1000000,
        "coin": {
        "tokenName": "B",
        "currencySymbol": "25ae866bd6b617664c054dc97f4d5b3a8cff3fb6ab0354622ada4a11"
        }
]
```


4.2 Collect Funds

Method	**POST**
URL	`http://localhost:8080/cd6f61d0-c503-4e4c-835d-d38a78613066/create-sell-order`
Response Code	202

4.3 Create Sell Order (Limit Order)

Method	**POST**
URL	`http://localhost:8080/cd6f61d0-c503-4e4c-835d-d38a78613066/create-sell-order`
Response Code	201

Request Body:

```
{
    "lockedCoin": {
        "tokenName": "A",
        "currencySymbol": "25ae866bd6b617664c054dc97f4d5b3a8cff3fb6ab0354622ada4a11"
    },
    "lockedAmount": 50,
    "expectedAmount": 50,
    "expectedCoin": {
        "tokenName": "B",
        "currencySymbol": "25ae866bd6b617664c054dc97f4d5b3a8cff3fb6ab0354622ada4a11"
    }
}
```

4.4 Create Liquidity Order

Method	**POST**
URL	`http://localhost:8080/cd6f61d0-c503-4e4c-835d-d38a78613066/create-liquidity-order`
Response Code	201

Request Body:

```
{
    "lockedCoin": {
        "currencySymbol": "25ae866bd6b617664c054dc97f4d5b3a8cff3fb6ab0354622ada4a11",
        "tokenName": "A"
    },
    "expectedCoin": {
        "currencySymbol": "25ae866bd6b617664c054dc97f4d5b3a8cff3fb6ab0354622ada4a11",
        "tokenName": "B"
    },
    "lockedAmount": 100,
    "expectedAmount": 100,
    "swapFee": 0.05
}
```

4.5 Create Liquidity Pool

Method	**POST**
URL	`http://localhost:8080/cd6f61d0-c503-4e4c-835d-d38a78613066/create-liquidity-pool`
Response Code	201

Request Body:

```
{
    "coinA": {
        "currencySymbol": "25ae866bd6b617664c054dc97f4d5b3a8cff3fb6ab0354622ada4a11",
        "tokenName": "A"
    },
    "coinB": {
    "currencySymbol": "25ae866bd6b617664c054dc97f4d5b3a8cff3fb6ab0354622ada4a11",
    "tokenName": "B"
    },
    "amountA": 1000,
    "poolPartsParams": {
        "coinAPriceChange": 0.6,
        "coinBPriceChange": 0.5,
        "numberOfParts": 3
    },
    "swapFee": 5.5,
    "exchangeRate": 5.5
}
```

4.6 List All Orders

Method	**GET**
URL	`http://localhost:8080/cd6f61d0-c503-4e4c-835d-d38a78613066/all-orders`
Response Code	200

Response Body:

```
[
    {
        "lockedCoin": {
        "amount": 1000,
        "coin": {
            "currencySymbol": "25ae866bd6b617664c054dc97f4d5b3a8cff3fb6ab0354622ada4a11",
            "tokenName": "A"
        }
        },
        "orderType": "Liquidity",
        "orderHash": {
        "txOutRefIdx": 1,
```

(continues on next page)

(continued from previous page)


```
        "txOutRefId": {
            "getTxId": "9dec4e3d5cd0b7c1265bdd4e8642d95ec3465e309427cb7288c946d25d09fc7d"
        }
        },
        "expectedCoin": {
        "amount": 100,
        "coin": {
            "currencySymbol": "25ae866bd6b617664c054dc97f4d5b3a8cff3fb6ab0354622ada4a11",
            "tokenName": "B"
        }
        }
    },
    {
        "lockedCoin": {
        "amount": 100,
        "coin": {
            "currencySymbol": "25ae866bd6b617664c054dc97f4d5b3a8cff3fb6ab0354622ada4a11",
            "tokenName": "A"
        }
        },
        "orderType": "Sell",
        "orderHash": {
        "txOutRefIdx": 1,
        "txOutRefId": {
            "getTxId": "e719479a74061667b66f8131740e11287770602ed26adf44c5b6b05509741c76"
        }
        },
        "expectedCoin": {
        "amount": 100,
        "coin": {
            "currencySymbol": "25ae866bd6b617664c054dc97f4d5b3a8cff3fb6ab0354622ada4a11",
            "tokenName": "B"
        }
        }
    }
]
```


4.7 List My Orders

Method	**GET**
URL	`http://localhost:8080/cd6f61d0-c503-4e4c-835d-d38a78613066/orders`
Response Code	200

Response Body:


```
[
    {
        "lockedCoin": {
        "amount": 100,
```


(continues on next page)

(continued from previous page)


```
        "coin": {
            "currencySymbol": "25ae866bd6b617664c054dc97f4d5b3a8cff3fb6ab0354622ada4a11",
            "tokenName": "A"
        }
        },
        "orderType": "Sell",
        "orderHash": {
        "txOutRefIdx": 1,
        "txOutRefId": {
            "getTxId": "e719479a74061667b66f8131740e11287770602ed26adf44c5b6b05509741c76"
        }
        },
        "expectedCoin": {
        "amount": 100,
        "coin": {
            "currencySymbol": "25ae866bd6b617664c054dc97f4d5b3a8cff3fb6ab0354622ada4a11",
            "tokenName": "B"
        }
        }
    }
]
```


4.8 List Orders By Coin Set

Method	**POST**
URL	`http://localhost:8080/cd6f61d0-c503-4e4c-835d-d38a78613066/orders-by-set`
Response Code	200

Request Body:


```
{
    "lockedCoin": {
        "currencySymbol": "25ae866bd6b617664c054dc97f4d5b3a8cff3fb6ab0354622ada4a11",
        "tokenName": "A"
    },
    "expectedCoin": {
    "currencySymbol": "25ae866bd6b617664c054dc97f4d5b3a8cff3fb6ab0354622ada4a11",
    "tokenName": "B"
    }
}
```


Response Body:


```
[
    {
        "lockedCoin": {
        "amount": 100,
        "coin": {
            "currencySymbol": "25ae866bd6b617664c054dc97f4d5b3a8cff3fb6ab0354622ada4a11",
```


(continues on next page)

(continued from previous page)


```
            "tokenName": "A"
        }
        },
        "orderType": "Sell",
        "orderHash": {
        "txOutRefIdx": 1,
        "txOutRefId": {
            "getTxId": "e719479a74061667b66f8131740e11287770602ed26adf44c5b6b05509741c76"
        }
        },
        "expectedCoin": {
        "amount": 100,
        "coin": {
            "currencySymbol": "25ae866bd6b617664c054dc97f4d5b3a8cff3fb6ab0354622ada4a11",
            "tokenName": "B"
        }
        }
    }
]
```


4.9 List All Coin Sets

Method	**GET**
URL	`http://localhost:8080/cd6f61d0-c503-4e4c-835d-d38a78613066/sets`
Response Code	200

Response Body:


```
[
    {
        "lockedCoin": {
        "tokenName": "B",
        "currencySymbol": "25ae866bd6b617664c054dc97f4d5b3a8cff3fb6ab0354622ada4a11"
        },
        "expectedCoin": {
        "tokenName": "A",
        "currencySymbol": "25ae866bd6b617664c054dc97f4d5b3a8cff3fb6ab0354622ada4a11"
        }
    },
    {
        "lockedCoin": {
        "tokenName": "A",
        "currencySymbol": "25ae866bd6b617664c054dc97f4d5b3a8cff3fb6ab0354622ada4a11"
        },
        "expectedCoin": {
        "tokenName": "B",
        "currencySymbol": "25ae866bd6b617664c054dc97f4d5b3a8cff3fb6ab0354622ada4a11"
        }
```


(continues on next page)

(continued from previous page)

```
    }
]
```

4.10 Cancel Submitted Order

Method	**POST**
URL	`http://localhost:8080/cd6f61d0-c503-4e4c-835d-d38a78613066/cancel`
Response Code	202

Request Body:

```
{
    "unTxOutRef": "0149cf347cdf4c2e"
}
```

4.11 Perform

Method	**POST**
URL	`http://localhost:8080/cd6f61d0-c503-4e4c-835d-d38a78613066/perform`
Response Code	202

4.12 Perform N Random

Method	**POST**
URL	`http://localhost:8080/cd6f61d0-c503-4e4c-835d-d38a78613066/perform-random`
Response Code	202

4.13 Stop

Method	**POST**
URL	`http://localhost:8080/cd6f61d0-c503-4e4c-835d-d38a78613066/stop`
Response Code	202

4.14 My Payouts

Method	**GET**
URL	`http://localhost:8080/cd6f61d0-c503-4e4c-835d-d38a78613066/payouts`
Response Code	200

Response Body:


```
[
    {
        "amount": 1000,
        "coin": {
        "currencySymbol": "25ae866bd6b617664c054dc97f4d5b3a8cff3fb6ab0354622ada4a11",
        "tokenName": "A"
        },
    },
    {
        "amount": 100,
        "coin": {
        "currencySymbol": "25ae866bd6b617664c054dc97f4d5b3a8cff3fb6ab0354622ada4a11",
        "tokenName": "B"
        }
    }
]
```