Bitmaido Project

* Data stream structure

Web Application

localhost:9082

localhost:9081

Web Socket Server

API Server

Redis Pub/Sub

MySQL

Redis

-Front page

Markets Component

Tradingview Chart

Component

Depth Component

OrderBook

Component

Order Component

Orders Component

Last Trades

Component

* Crypto\_margin server

Crypto\_margin server

modules

Socket communication url: <http://localhost:9082>

Rest Api communication url: http://localhost:9081

1. Frontend side

Last Trades

Component

Orders Component

Order Component

OrderBook

Component

Depth Component

Tradingview Chart

Component

Markets Component

**Import Data to show in front**

* Markets Component (Store/markets.module.js)

Getters/markets ()

* Tradingview Chart Component (Store/markets.module.js)

Getters/quote ()

* OrderBook Component (Store/markets.module.js)

Getters/orderBook()

* Orders Component (Store/trade.module.js)

Getters/orders()

* Last Trades Component (Store/markets.module.js)

Getters/lastTrades()

**Socket connect**

* App.vue/mounted()

Dispatch(APP\_INIT)

* Store/io.module.js

1. Backend side
2. Server/app.js

Socket connection

Server/server/socket.js

* Server/module/crypto\_margin.js

Create models related Crypto\_margin

Define Rest init router

1. Server/engines/crypto\_margin/app.js

App.js

./modules/

Marketmaker.js

Orderbook.js

Platform.js (module related with mysql model)

Storage.js (module manipulating data about database)

**Initial run module**

* Module/marketmaker.js

Generate token data form via setTimeout()

{

User\_id,

Account\_id,

Symbol,

Quantity,

Type,

Side,

}

In this function, get quote about each coin using “websocket (‘wss//stream.binance.com/stream’)”

This data send orderbook module

redisPub.publish(‘crypto\_margin:request’, data)

* Module/orderbook.js

Add properties in token data form

ProcessOpenRequest()

{

User\_id,

Account\_id,

Symbol,

Quantity,

Type,

Side,

Price,

Leverage,

Margin,

Id,

Quantity\_filled,

Time\_open,

Status,

}

Publish(‘crypto\_margin:order, order)

* Subscribe (‘crypto\_margin:order, order)
* Marketmaker,
* Storage,
* Crypto\_margin, (server/module/)

1. Server/engines/crypto/app.js
2. Buy/Sell stream in order

* first step (insert db)
* front

click buy or sell in order component

go to store/trade.module.js

call action/[ORDER\_REQUEST]

RestService.post(‘crypto\_margin/id/orders, order)

* backend

server/module/crypto\_margin.js

call openorder()

reidsPub.publish(‘crypto\_margin:request’, order)

if you clicked BUY button, the result is

{

side: 1,

symbol: 'BTCUSDTm',

type: 'LIMIT',

quantity: 4000000,

price: 59000,

priceStop: 0,

leverage: 100,

account\_id: 1,

user\_id: 1,

price\_stop: NaN,

margin: 39010.86303765358,

id: '1118b4a9-04aa-481f-8c67-256e643b0162',

quantity\_filled: 0,

time\_open: 1634951372536,

status: 'NEW',

position\_id: '3e67a032-7b37-4603-8bfc-bd056a734062'

}

if you clicked SELL button, the result is

{

side: 0,

symbol: 'BTCUSDTm',

type: 'LIMIT',

quantity: 4000000,

price: 59000,

priceStop: 0,

leverage: 100,

account\_id: 1,

user\_id: 1,

price\_stop: NaN,

margin: 39010.86303765358,

id: '1118b4a9-04aa-481f-8c67-256e643b0162',

quantity\_filled: 0,

time\_open: 1634951372536,

status: 'NEW',

position\_id: '3e67a032-7b37-4603-8bfc-bd056a734062'

}

* server/engine/crypto\_margin/module/orderbook.js

redisSub.subscribe(‘crypto\_margin:request’)

case ‘open’

call processopenRequest()

go to ./storage.js

insert order to crypto\_margin\_order table in mySQL database

* second step (show trade)
* front

trade.module/mutations

call [ACCOUNT]

get(‘crypto\_margin/id/orders’)

get(‘crypto\_margin/id/positions’)

…

* backend

Server/moudle/crypto\_margin.js

Call Getorder()

This results are records that status is ‘ACTIVE’ or ‘PARTIALLY\_FILLED’