華宮
凪沙
メガネスタイル!
失われた未来を求めて

Building Subgraphs

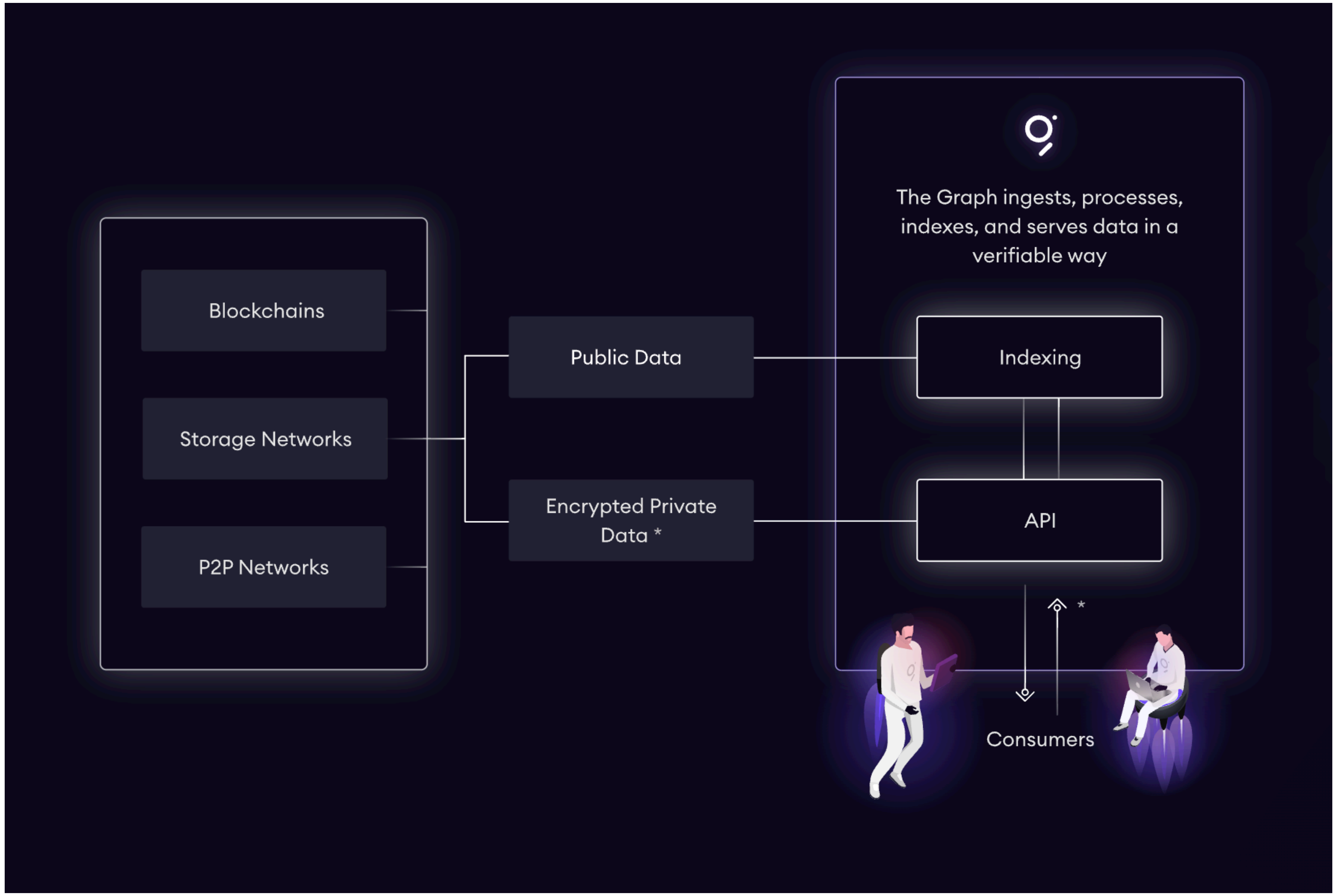
A case study with Yearn Finance

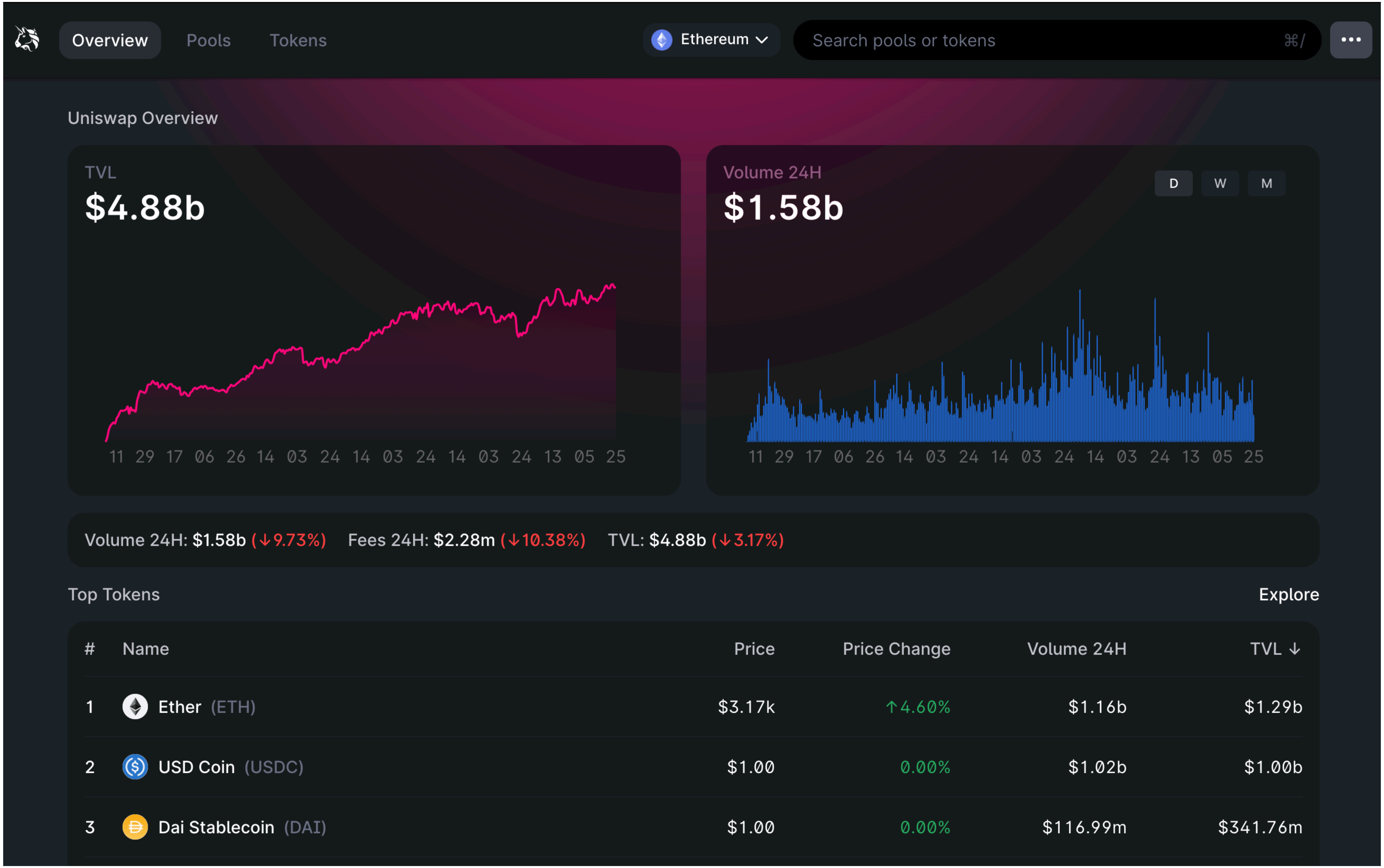
Vincent @ 03/30/2022

Agenda

- What is The Graph?
- What is a Subgraph?
- What is Yearn?
- Subgraph Schema
- Deconstructing Yearn
- Mapping Events
- Debugging/Validation

What is The Graph?


Blockchains
Storage Networks
P2P Networks
Public Data
Encrypted Private Data *
The Graph ingests, processes, indexes, and serves data in a verifiable way
Indexing
API
*
Consumers

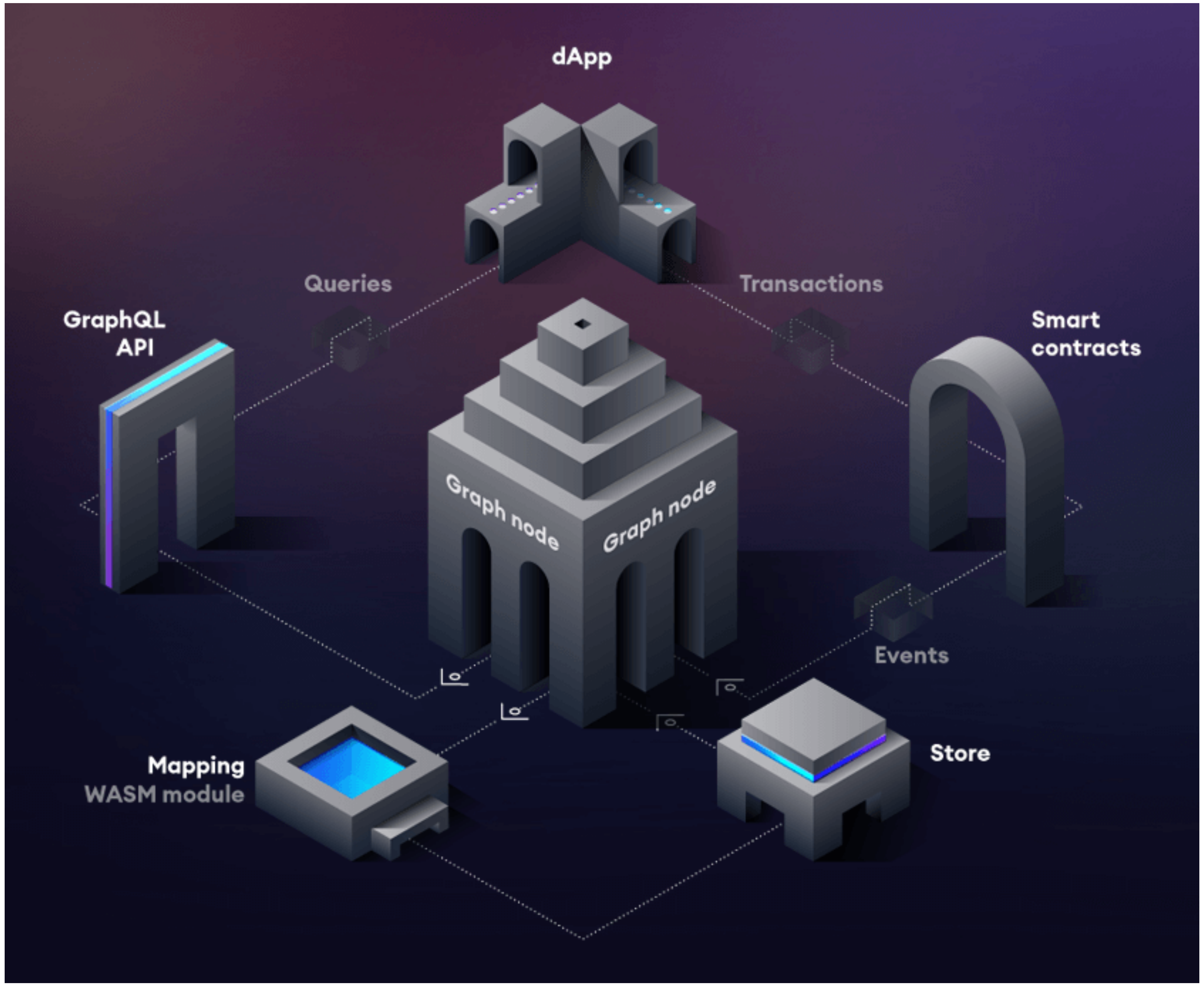


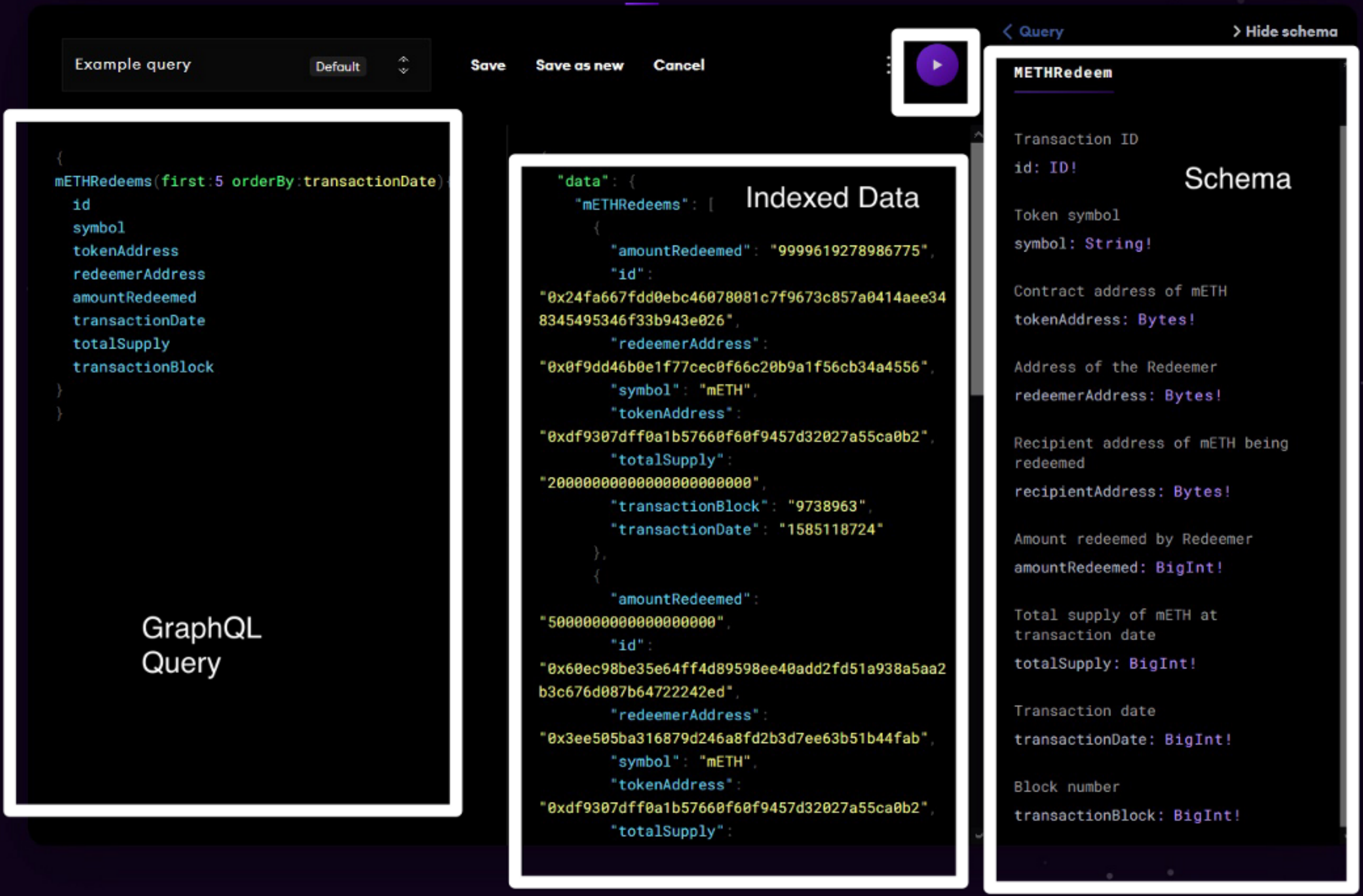
Uniswap Overview

Volume 24H: $1.58b (↓9.73%) Fees 24H: $2.28m (↓10.38%) TVL: $4.88b (↓3.17%)

Top Tokens

#	Name	Price	Price Change	Volume 24H	TVL ↓
1	Ether (ETH)	$3.17k	↑4.60%	$1.16b	$1.29b
2	USD Coin (USDC)	$1.00	0.00%	$1.02b	$1.00b
3	Dai Stablecoin (DAI)	$1.00	0.00%	$116.99m	$341.76m


dAapp
Queries
Transactions
GraphQL API
Smart contracts
Graph node
Graph node
Events
Mapping
WASM module
Store



```
{
mETHRedeems(first:5 orderBy:transactionDate){
  id
  symbol
  tokenAddress
  redeemerAddress
  amountRedeemed
  transactionDate
  totalSupply
  transactionBlock
 }
}
```

```
  "data": {
    "mETHRedeems": [
      {
        "amountRedeemed": "9999619278986775",
        "id":
"0x24fa667fdd0ebc46078081c7f9673c857a0414aee34
8345495346f33b943e026",
        "redeemerAddress":
"0x0f9dd46b0e1f77cec0f66c20b9a1f56cb34a4556",
        "symbol": "mETH",
        "tokenAddress":
"0xdf9307dff0a1b57660f60f9457d32027a55ca0b2",
        "totalSupply":
"2000000000000000000000",
        "transactionBlock": "9738963",
        "transactionDate": "1585118724"
      },
      {
        "amountRedeemed":
"5000000000000000000",
        "id":
"0x60ec98be35e64ff4d89598ee40add2fd51a938a5aa2
b3c676d087b64722242ed",
        "redeemerAddress":
"0x3ee505ba316879d246a8fd2b3d7ee63b51b44fab",
        "symbol": "mETH",
        "tokenAddress":
"0xdf9307dff0a1b57660f60f9457d32027a55ca0b2",
        "totalSupply":
```

```
METHRedeem

Transaction ID
id: ID!

Token symbol
symbol: String!

Contract address of mETH
tokenAddress: Bytes!

Address of the Redeemer
redeemerAddress: Bytes!

Recipient address of mETH being
redeemed
recipientAddress: Bytes!

Amount redeemed by Redeemer
amountRedeemed: BigInt!

Total supply of mETH at
transaction date
totalSupply: BigInt!

Transaction date
transactionDate: BigInt!

Block number
transactionBlock: BigInt!
```

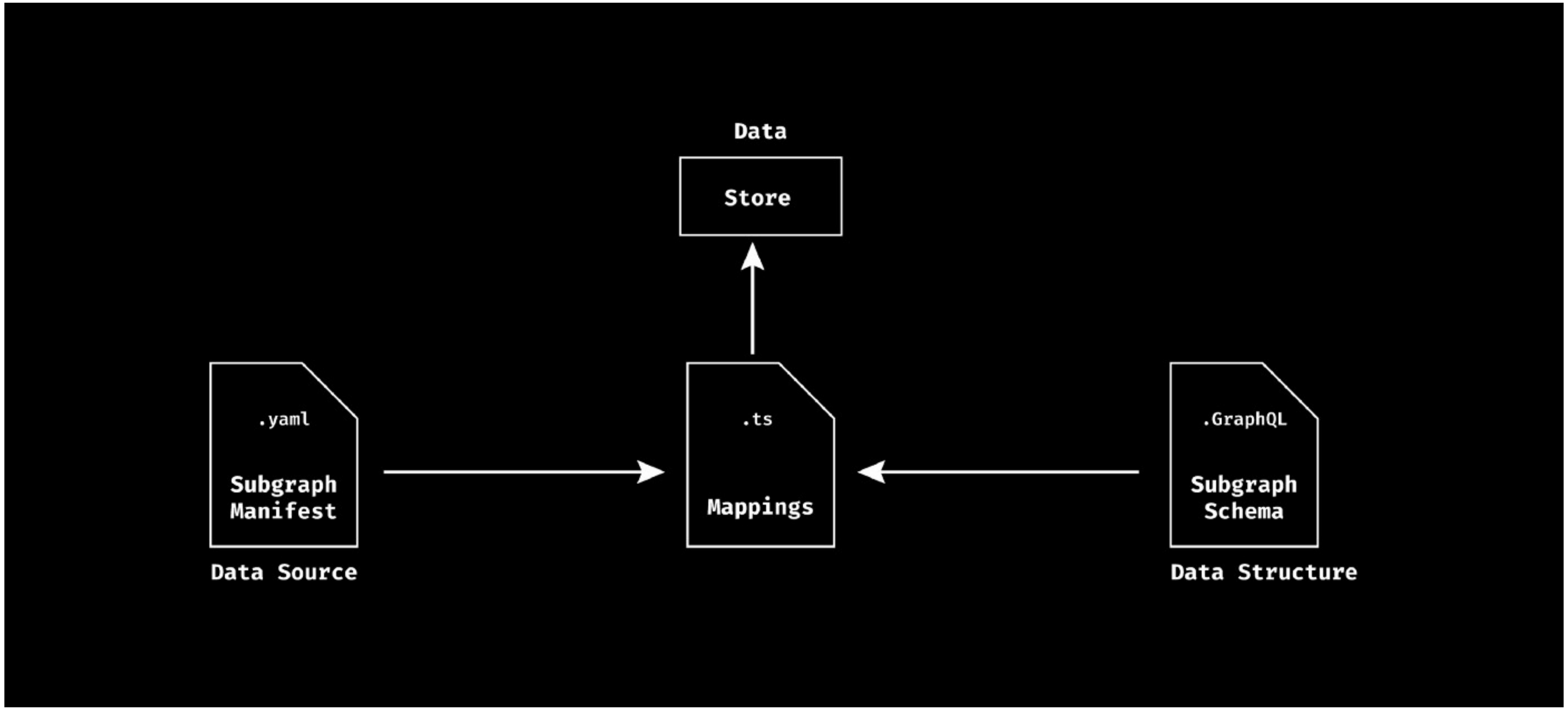
What is a Subgraph?

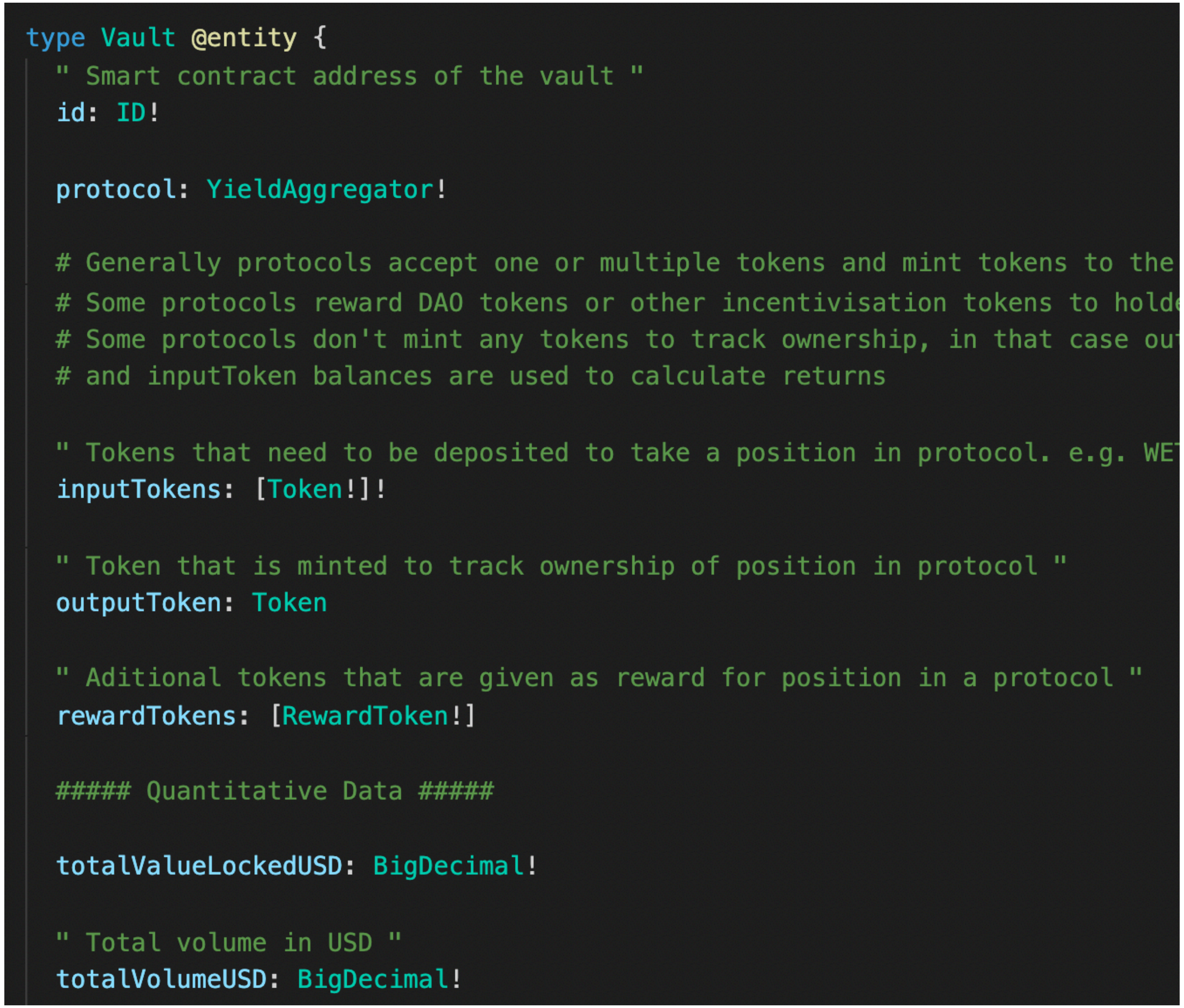
What is a Subgraph?

A subgraph defines which data The Graph will index from the blockchain, how to process these data, and how to store them.

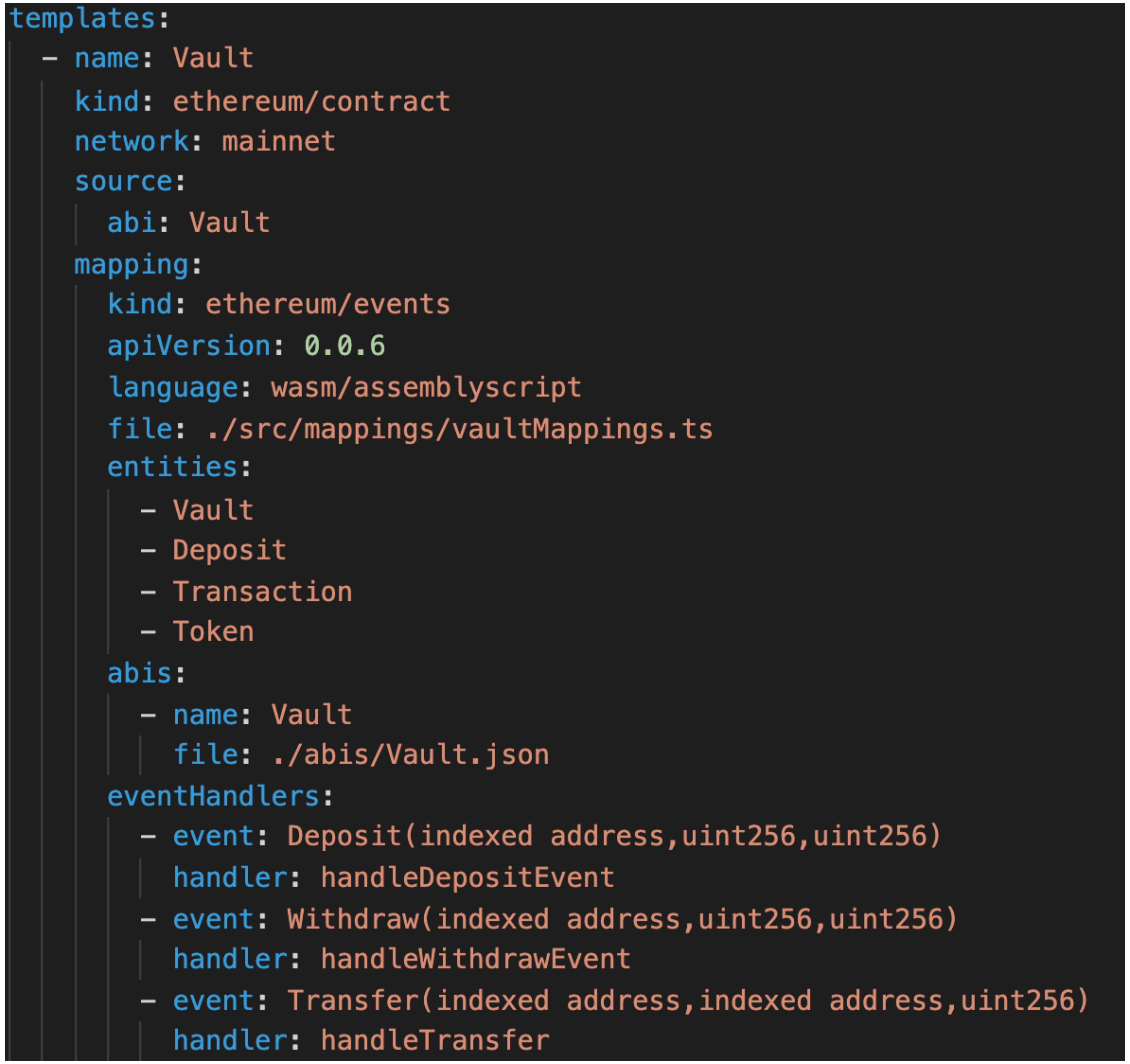
It consists of three main components:

- Subgraph Manifest
- Subgraph Schema
- AssemblyScript Mappings

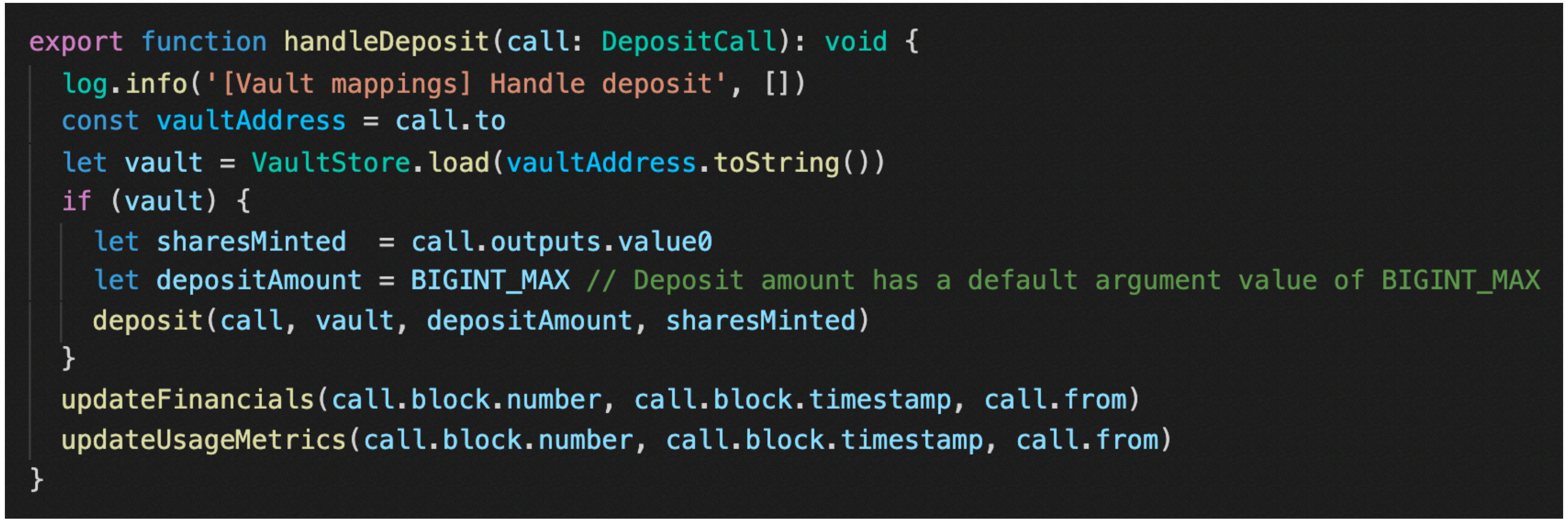

Data
Store
.yaml
Subgraph Manifest
Data Source
.ts
Mappings
.GraphQL
Subgraph Schema
Data Structure



```
type Vault @entity {
  " Smart contract address of the vault "
  id: ID!

  protocol: YieldAggregator!

  # Generally protocols accept one or multiple tokens and mint tokens to the
  # Some protocols reward DAO tokens or other incentivisation tokens to holde
  # Some protocols don't mint any tokens to track ownership, in that case out
  # and inputToken balances are used to calculate returns

  " Tokens that need to be deposited to take a position in protocol. e.g. WE
  inputTokens: [Token!]!

  " Token that is minted to track ownership of position in protocol "
  outputToken: Token

  " Aditional tokens that are given as reward for position in a protocol "
  rewardTokens: [RewardToken!]

  ##### Quantitative Data #####

  totalValueLockedUSD: BigDecimal!

  " Total volume in USD "
  totalVolumeUSD: BigDecimal!
```



```yaml
templates:
  - name: Vault
    kind: ethereum/contract
    network: mainnet
    source:
      abi: Vault
    mapping:
      kind: ethereum/events
      apiVersion: 0.0.6
      language: wasm/assemblyscript
      file: ./src/mappings/vaultMappings.ts
      entities:
        - Vault
        - Deposit
        - Transaction
        - Token
      abis:
        - name: Vault
          file: ./abis/Vault.json
      eventHandlers:
        - event: Deposit(indexed address,uint256,uint256)
          handler: handleDepositEvent
        - event: Withdraw(indexed address,uint256,uint256)
          handler: handleWithdrawEvent
        - event: Transfer(indexed address,indexed address,uint256)
          handler: handleTransfer
```



```
export function handleDeposit(call: DepositCall): void {
  log.info('[Vault mappings] Handle deposit', [])
  const vaultAddress = call.to
  let vault = VaultStore.load(vaultAddress.toString())
  if (vault) {
    let sharesMinted  = call.outputs.value0
    let depositAmount = BIGINT_MAX // Deposit amount has a default argument value of BIGINT_MAX
    deposit(call, vault, depositAmount, sharesMinted)
  }
  updateFinancials(call.block.number, call.block.timestamp, call.from)
  updateUsageMetrics(call.block.number, call.block.timestamp, call.from)
}
```

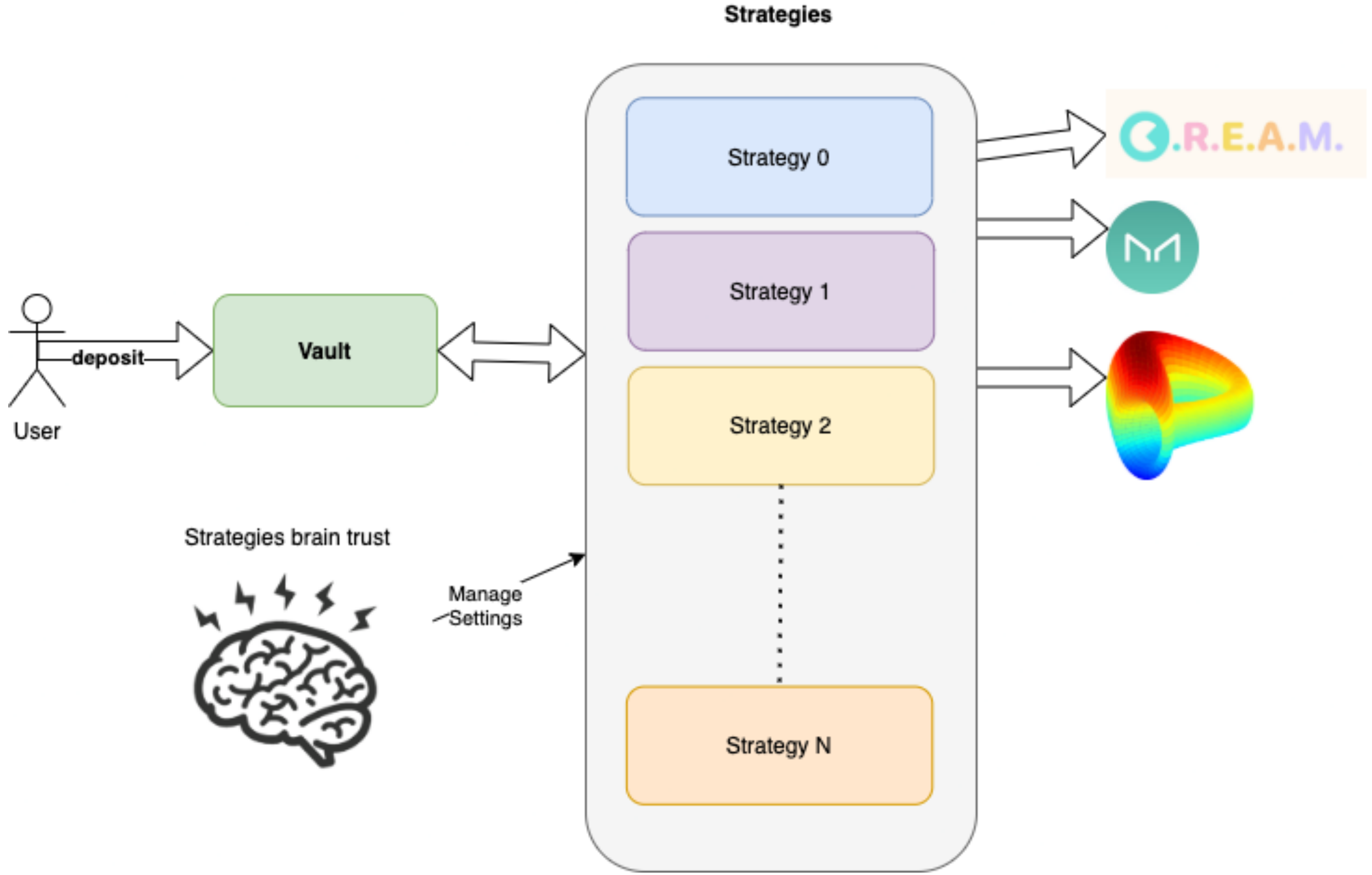
What is Yearn?

WANT SOME YFU?

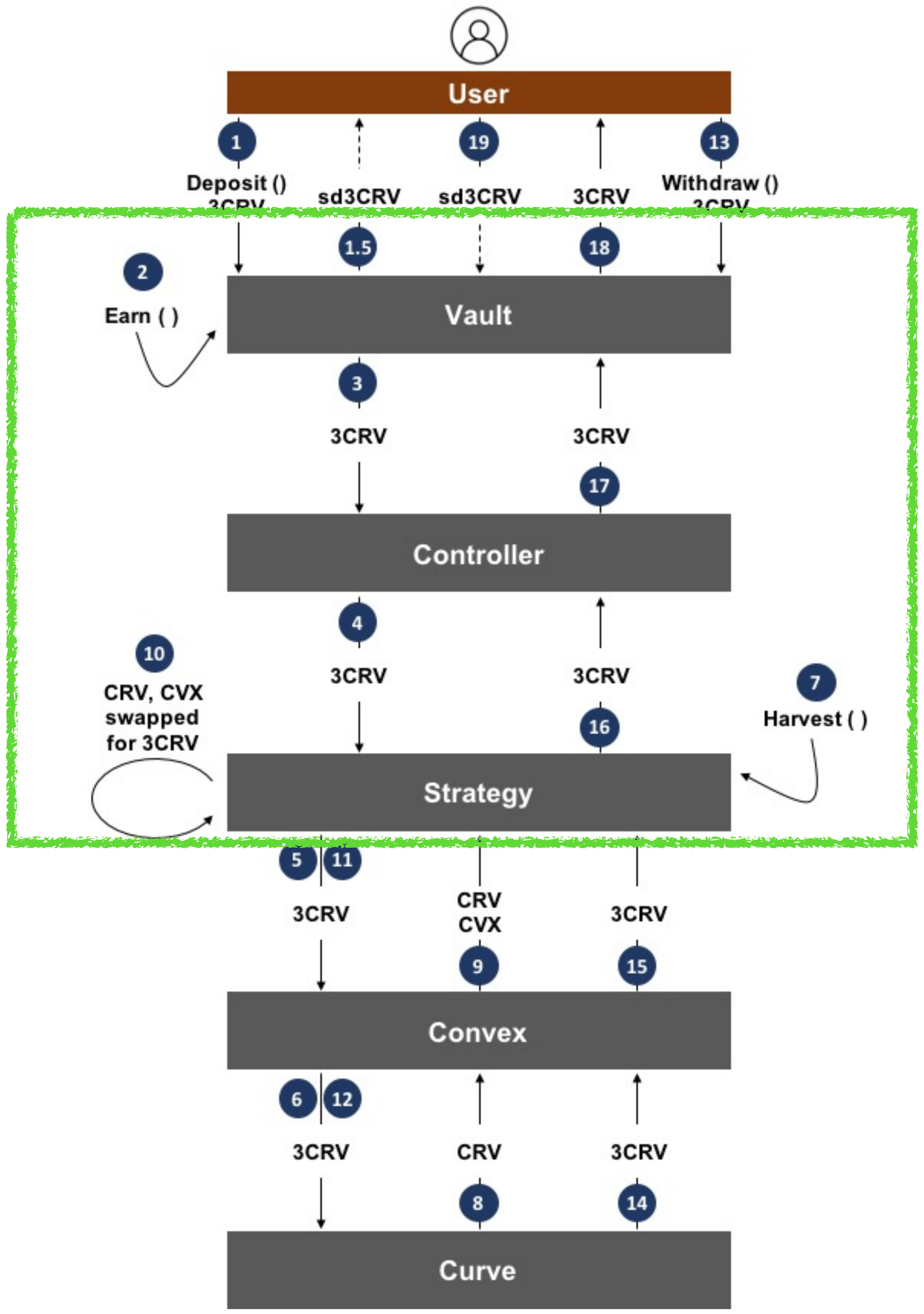
JUST DON´T TELL YOUR

$YFI

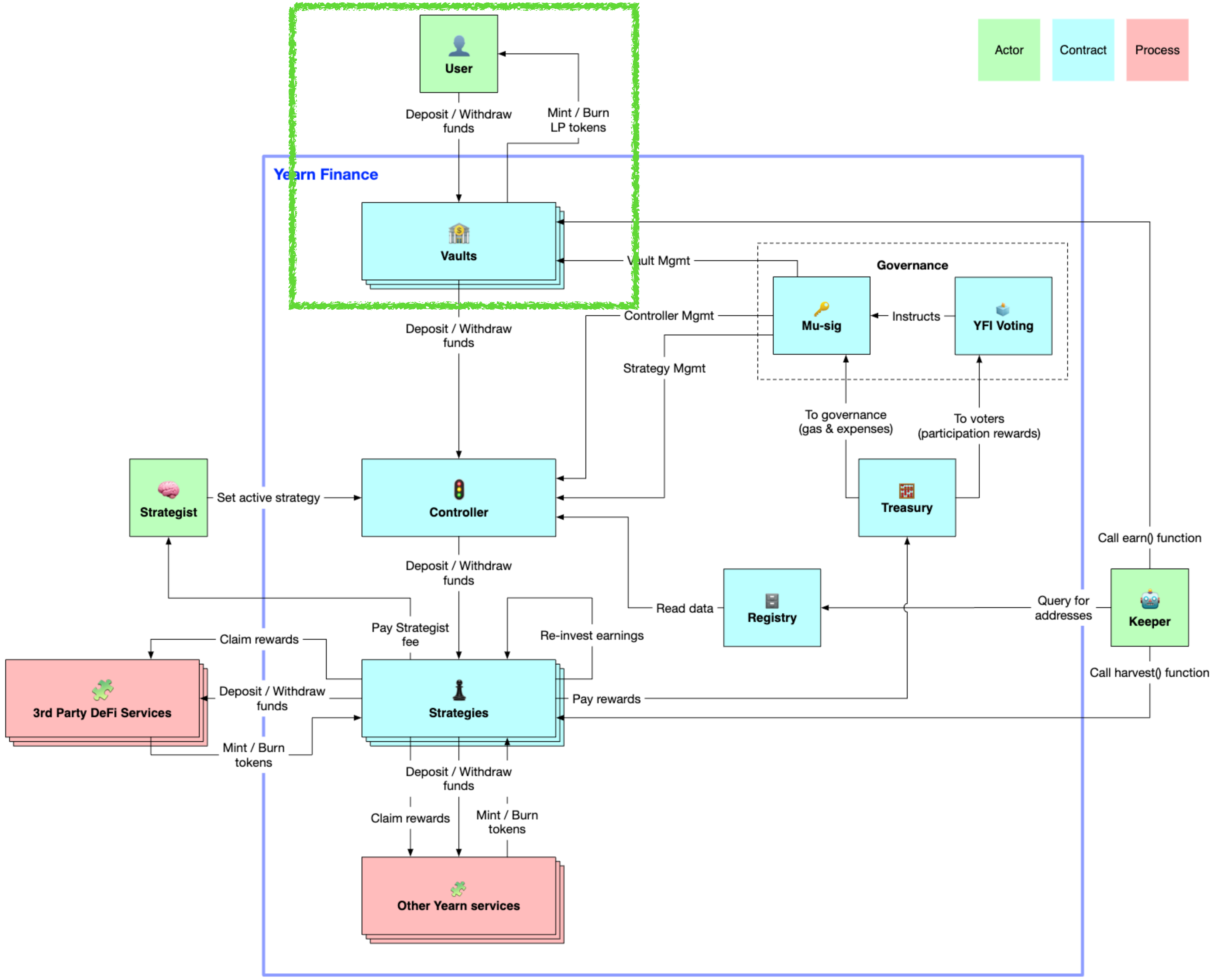
YFU.FINANCE

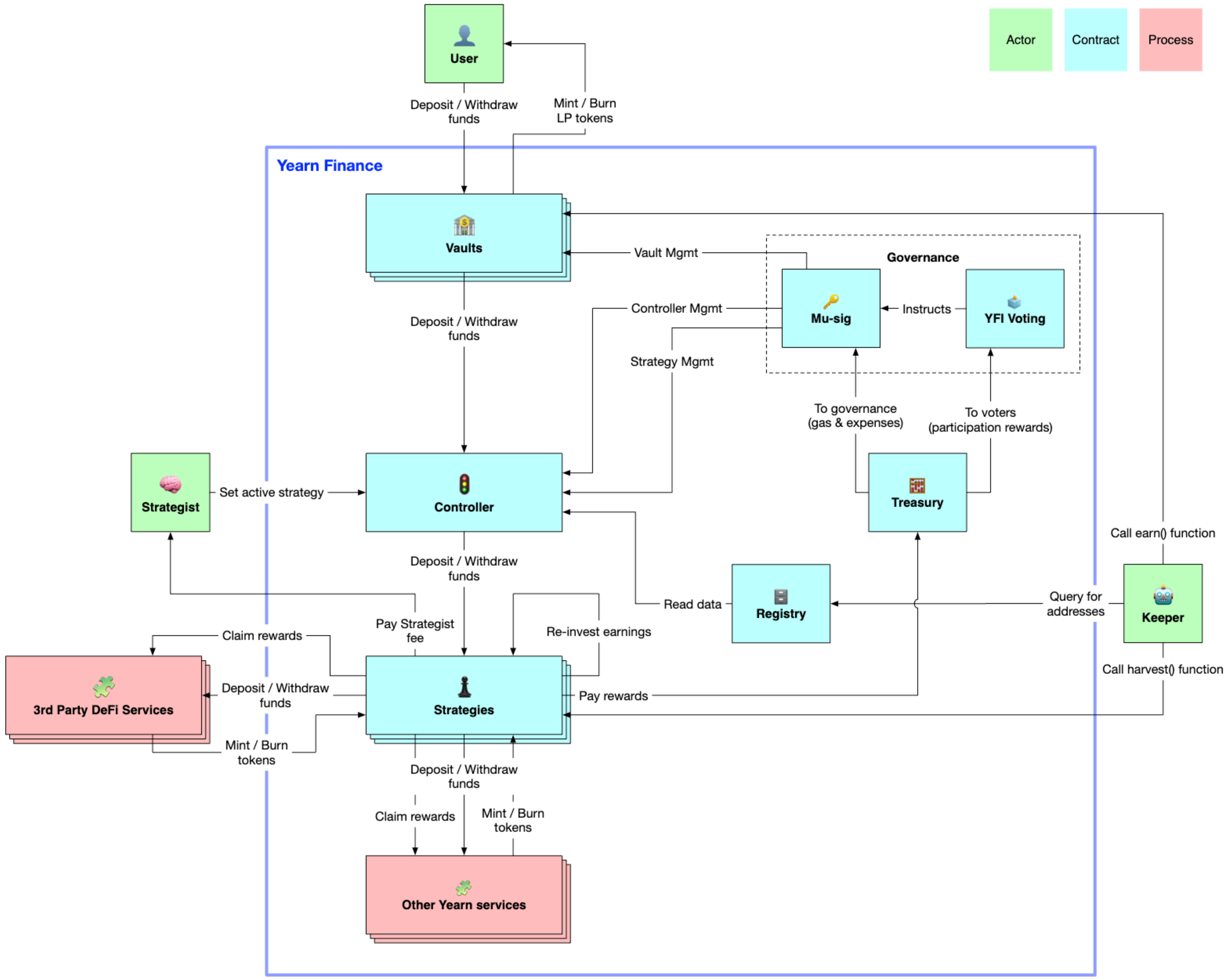

Strategies
User
deposit
Vault
Strategy 0
Strategy 1
Strategy 2
Strategy N
C.R.E.A.M.
Strategies brain trust
Manage Settings

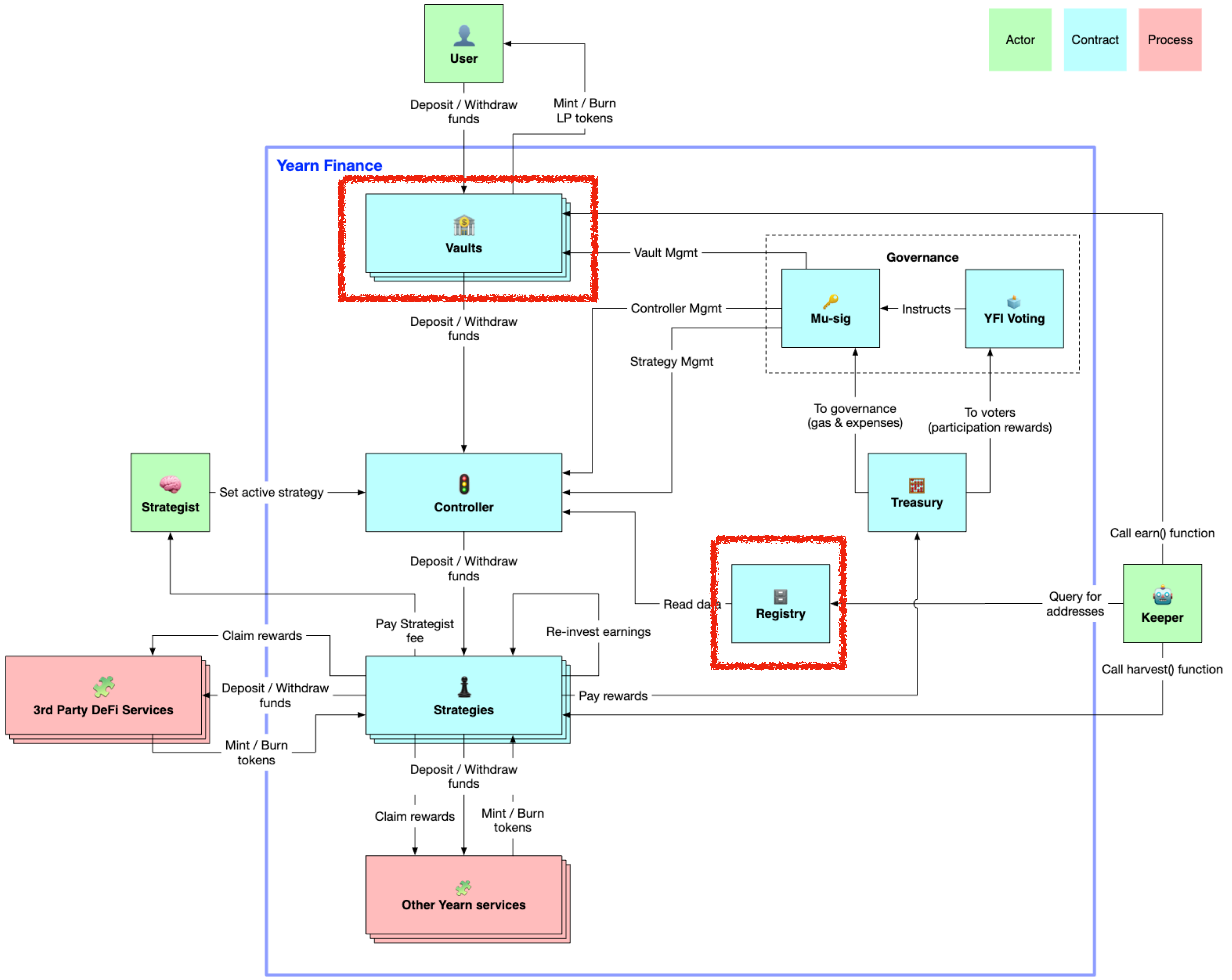
Demo


User
1
Deposit ()
3CRV
sd3CRV
19
sd3CRV
3CRV
13
Withdraw ()
3CRV
1.5
18
2
Earn ( )
Vault
3
3CRV
3CRV
17
Controller
4
3CRV
3CRV
16
10
CRV, CVX
swapped
for 3CRV
7
Harvest ( )
Strategy
5
11
3CRV
CRV
CVX
9
3CRV
15
Convex
6
12
3CRV
CRV
8
3CRV
14
Curve

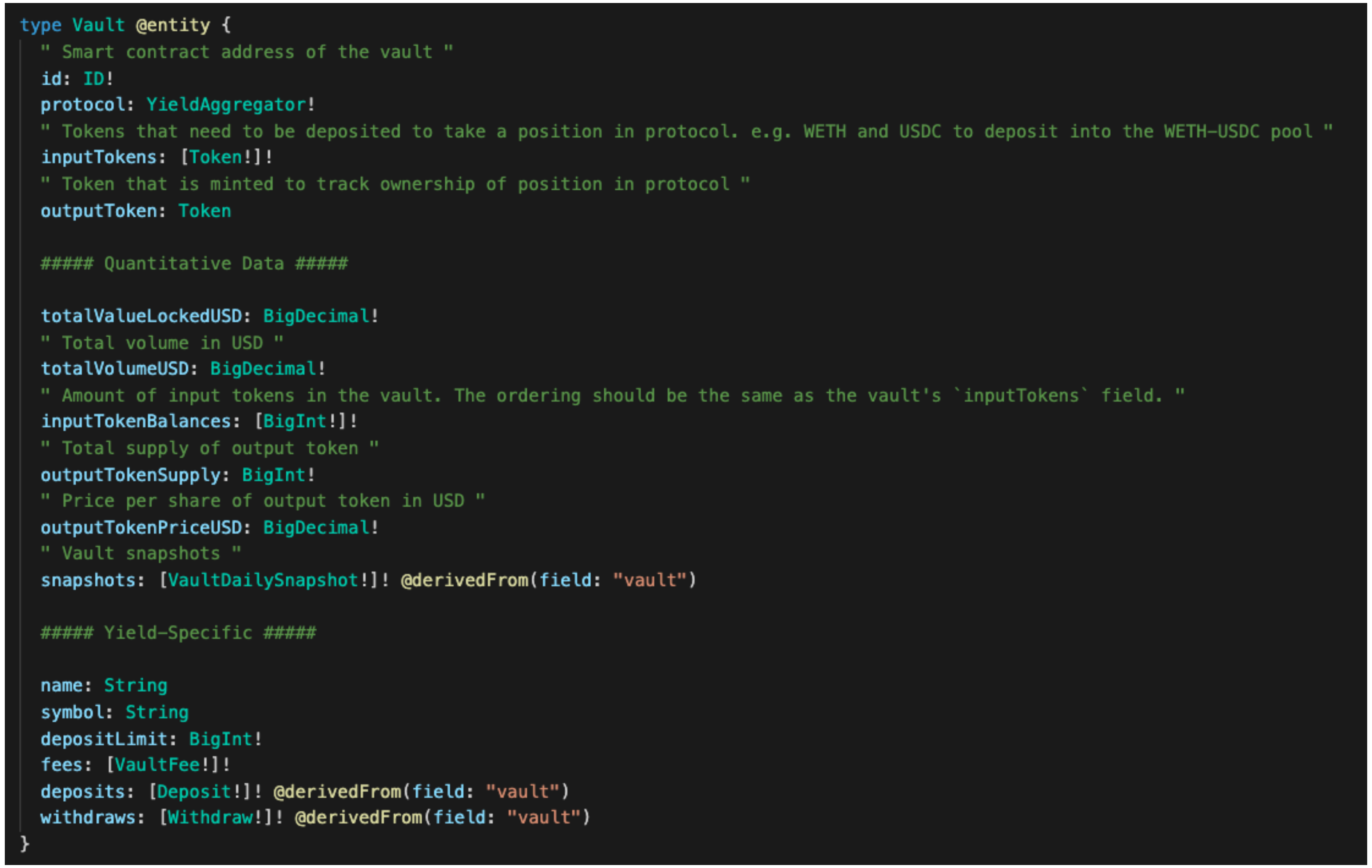
Deconstructing Yearn


Actor
Contract
Process
User
Deposit / Withdraw funds
Mint / Burn LP tokens
Yearn Finance
Vaults
Vault Mgmt
Governance
Mu-sig
Instructs
YFI Voting
Controller Mgmt
Deposit / Withdraw funds
Strategy Mgmt
To governance (gas & expenses)
To voters (participation rewards)
Strategist
Set active strategy
Controller
Treasury
Call earn() function
Deposit / Withdraw funds
Read data
Registry
Query for addresses
Keeper
Pay Strategist fee
Re-invest earnings
Claim rewards
3rd Party DeFi Services
Deposit / Withdraw funds
Strategies
Pay rewards
Call harvest() function
Mint / Burn tokens
Deposit / Withdraw funds
Claim rewards
Mint / Burn tokens
Other Yearn services

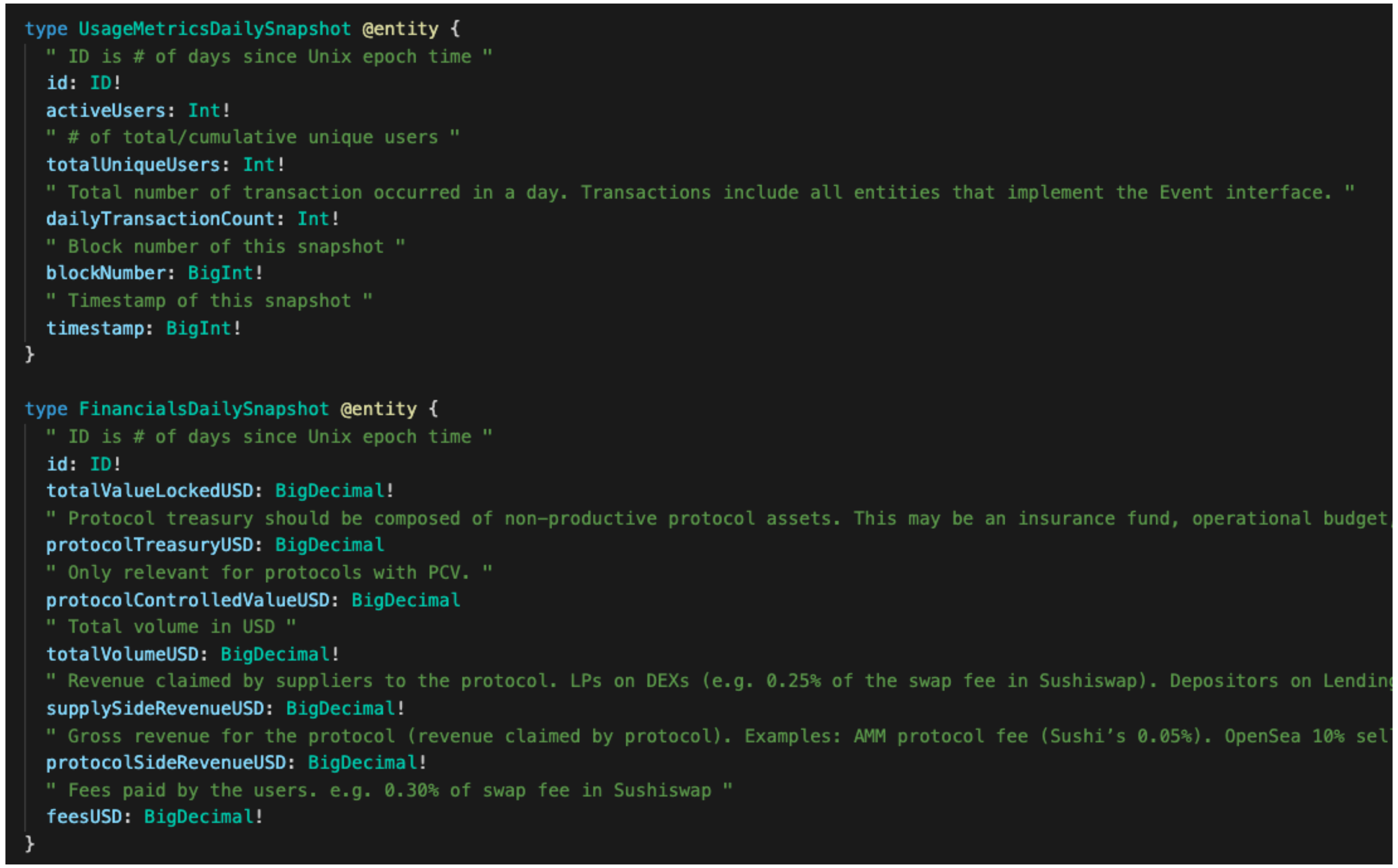

Actor
Contract
Process
User
Deposit / Withdraw funds
Mint / Burn LP tokens
Yearn Finance
Vaults
Vault Mgmt
Governance
Mu-sig
Instructs
YFI Voting
Controller Mgmt
Strategy Mgmt
Deposit / Withdraw funds
To governance (gas & expenses)
To voters (participation rewards)
Strategist
Set active strategy
Controller
Treasury
Call earn() function
Deposit / Withdraw funds
Read data
Registry
Query for addresses
Keeper
Pay Strategist fee
Re-invest earnings
Claim rewards
Call harvest() function
3rd Party DeFi Services
Deposit / Withdraw funds
Strategies
Pay rewards
Mint / Burn tokens
Deposit / Withdraw funds
Claim rewards
Mint / Burn tokens
Other Yearn services


Actor
Contract
Process
User
Deposit / Withdraw funds
Mint / Burn LP tokens
Yearn Finance
Vaults
Vault Mgmt
Governance
Controller Mgmt
Mu-sig
Instructs
YFI Voting
Deposit / Withdraw funds
Strategy Mgmt
To governance (gas & expenses)
To voters (participation rewards)
Strategist
Set active strategy
Controller
Treasury
Call earn() function
Deposit / Withdraw funds
Read data
Registry
Query for addresses
Keeper
Pay Strategist fee
Re-invest earnings
Claim rewards
Call harvest() function
3rd Party DeFi Services
Deposit / Withdraw funds
Strategies
Pay rewards
Mint / Burn tokens
Deposit / Withdraw funds
Claim rewards
Mint / Burn tokens
Other Yearn services

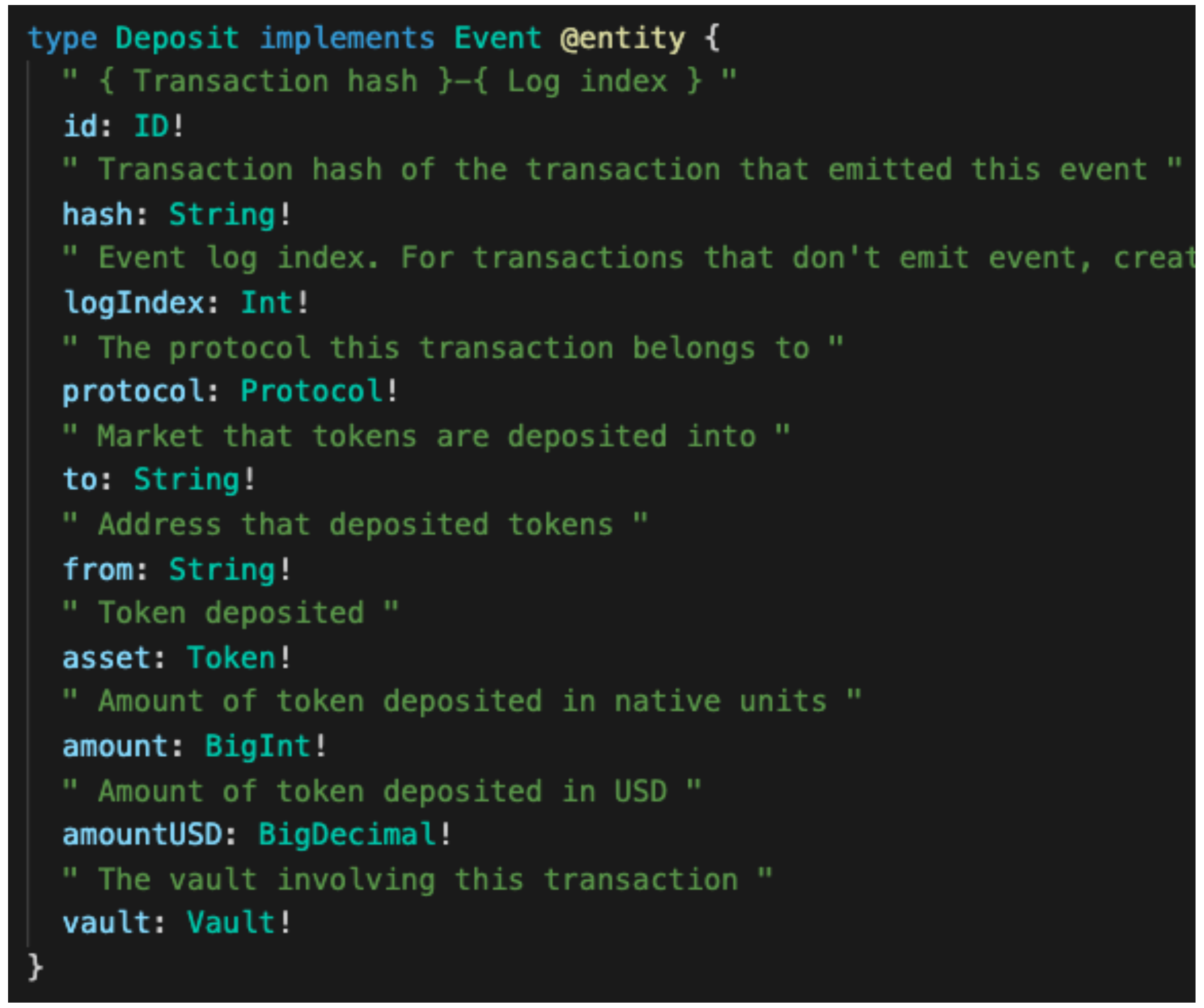
Subgraph Schema



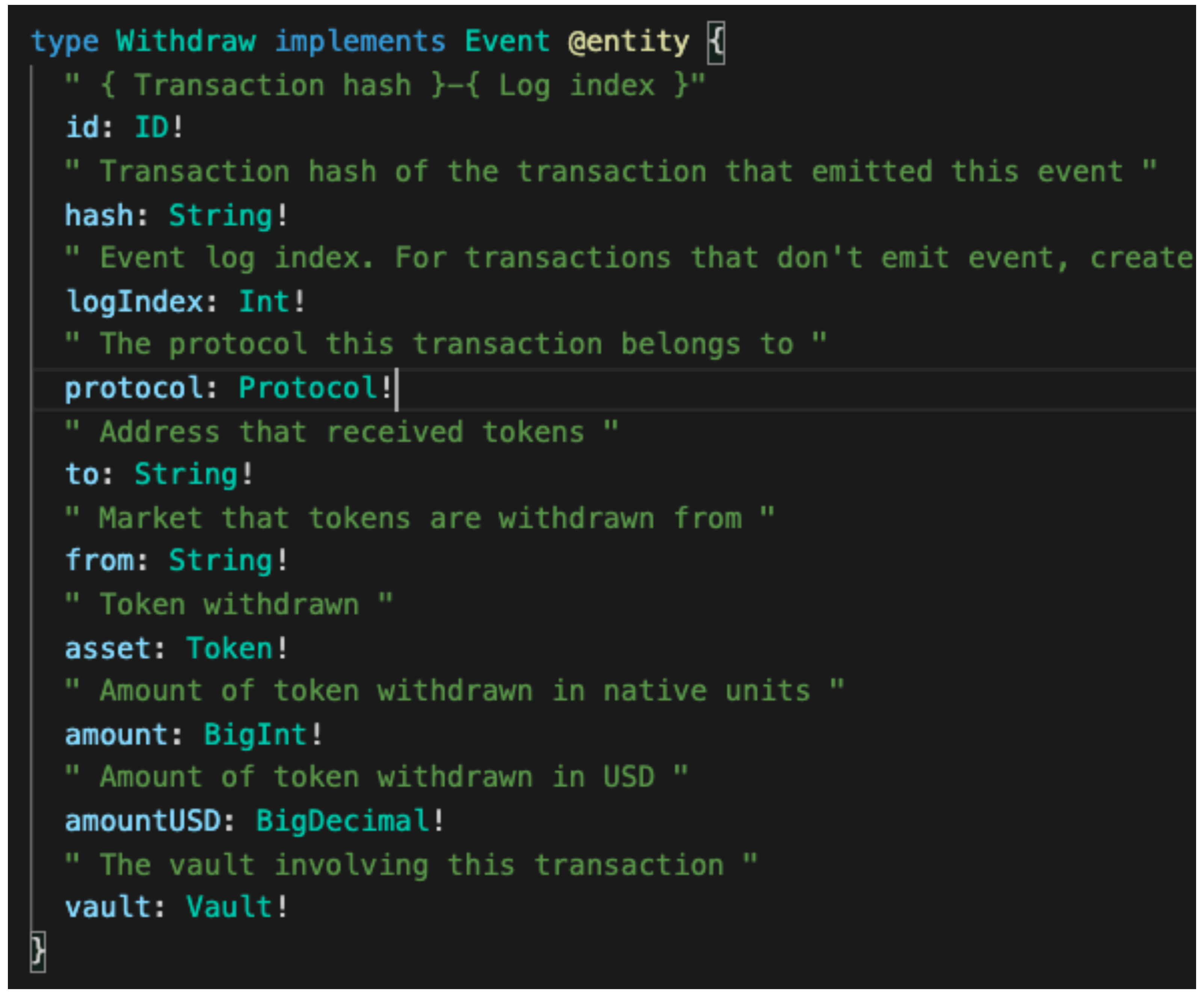
```graphql
type Vault @entity {
  " Smart contract address of the vault "
  id: ID!
  protocol: YieldAggregator!
  " Tokens that need to be deposited to take a position in protocol. e.g. WETH and USDC to deposit into the WETH-USDC pool "
  inputTokens: [Token!]!
  " Token that is minted to track ownership of position in protocol "
  outputToken: Token

  ##### Quantitative Data #####

  totalValueLockedUSD: BigDecimal!
  " Total volume in USD "
  totalVolumeUSD: BigDecimal!
  " Amount of input tokens in the vault. The ordering should be the same as the vault's `inputTokens` field. "
  inputTokenBalances: [BigInt!]!
  " Total supply of output token "
  outputTokenSupply: BigInt!
  " Price per share of output token in USD "
  outputTokenPriceUSD: BigDecimal!
  " Vault snapshots "
  snapshots: [VaultDailySnapshot!]! @derivedFrom(field: "vault")

  ##### Yield-Specific #####

  name: String
  symbol: String
  depositLimit: BigInt!
  fees: [VaultFee!]!
  deposits: [Deposit!]! @derivedFrom(field: "vault")
  withdraws: [Withdraw!]! @derivedFrom(field: "vault")
}
```



```
type UsageMetricsDailySnapshot @entity {
  " ID is # of days since Unix epoch time "
  id: ID!
  activeUsers: Int!
  " # of total/cumulative unique users "
  totalUniqueUsers: Int!
  " Total number of transaction occurred in a day. Transactions include all entities that implement the Event interface. "
  dailyTransactionCount: Int!
  " Block number of this snapshot "
  blockNumber: BigInt!
  " Timestamp of this snapshot "
  timestamp: BigInt!
}

type FinancialsDailySnapshot @entity {
  " ID is # of days since Unix epoch time "
  id: ID!
  totalValueLockedUSD: BigDecimal!
  " Protocol treasury should be composed of non-productive protocol assets. This may be an insurance fund, operational budget
  protocolTreasuryUSD: BigDecimal
  " Only relevant for protocols with PCV. "
  protocolControlledValueUSD: BigDecimal
  " Total volume in USD "
  totalVolumeUSD: BigDecimal!
  " Revenue claimed by suppliers to the protocol. LPs on DEXs (e.g. 0.25% of the swap fee in Sushiswap). Depositors on Lendin
  supplySideRevenueUSD: BigDecimal!
  " Gross revenue for the protocol (revenue claimed by protocol). Examples: AMM protocol fee (Sushi’s 0.05%). OpenSea 10% sel
  protocolSideRevenueUSD: BigDecimal!
  " Fees paid by the users. e.g. 0.30% of swap fee in Sushiswap "
  feesUSD: BigDecimal!
}
```



```
type Deposit implements Event @entity {
  " { Transaction hash }-{ Log index } "
  id: ID!
  " Transaction hash of the transaction that emitted this event "
  hash: String!
  " Event log index. For transactions that don't emit event, creat
  logIndex: Int!
  " The protocol this transaction belongs to "
  protocol: Protocol!
  " Market that tokens are deposited into "
  to: String!
  " Address that deposited tokens "
  from: String!
  " Token deposited "
  asset: Token!
  " Amount of token deposited in native units "
  amount: BigInt!
  " Amount of token deposited in USD "
  amountUSD: BigDecimal!
  " The vault involving this transaction "
  vault: Vault!
}
```



```
type Withdraw implements Event @entity {
  " { Transaction hash }-{ Log index }"
  id: ID!
  " Transaction hash of the transaction that emitted this event "
  hash: String!
  " Event log index. For transactions that don't emit event, create
  logIndex: Int!
  " The protocol this transaction belongs to "
  protocol: Protocol!
  " Address that received tokens "
  to: String!
  " Market that tokens are withdrawn from "
  from: String!
  " Token withdrawn "
  asset: Token!
  " Amount of token withdrawn in native units "
  amount: BigInt!
  " Amount of token withdrawn in USD "
  amountUSD: BigDecimal!
  " The vault involving this transaction "
  vault: Vault!
}
```

Diagram

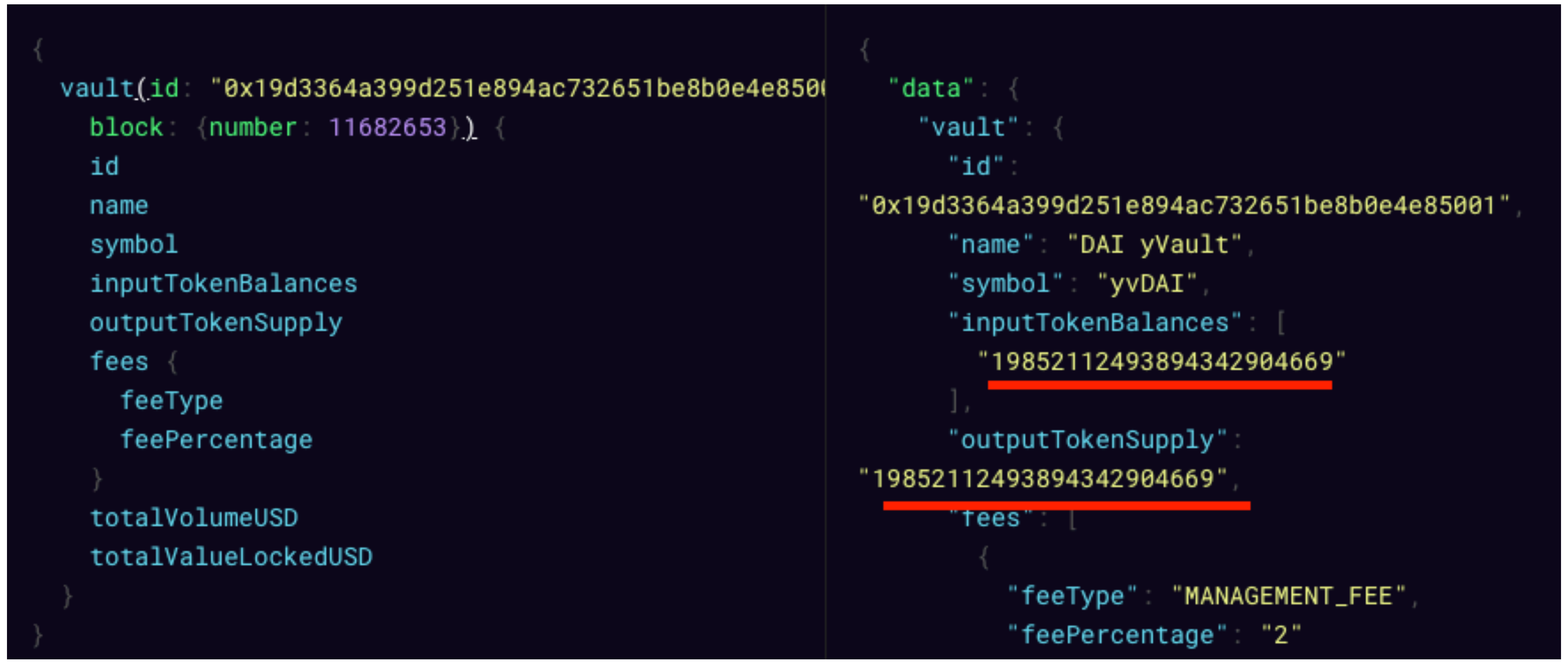
Mapping Events

Vaults

Mapping Events

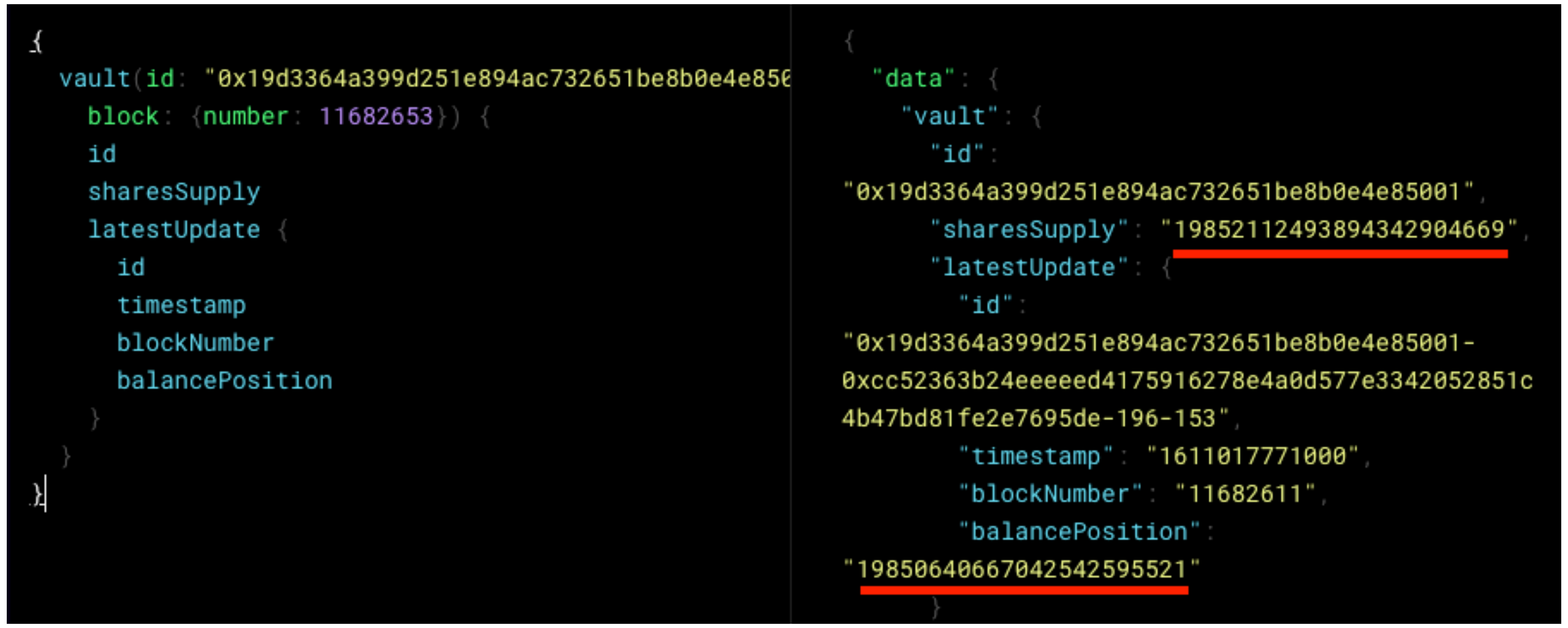
Deposits / Withdraws

Debugging



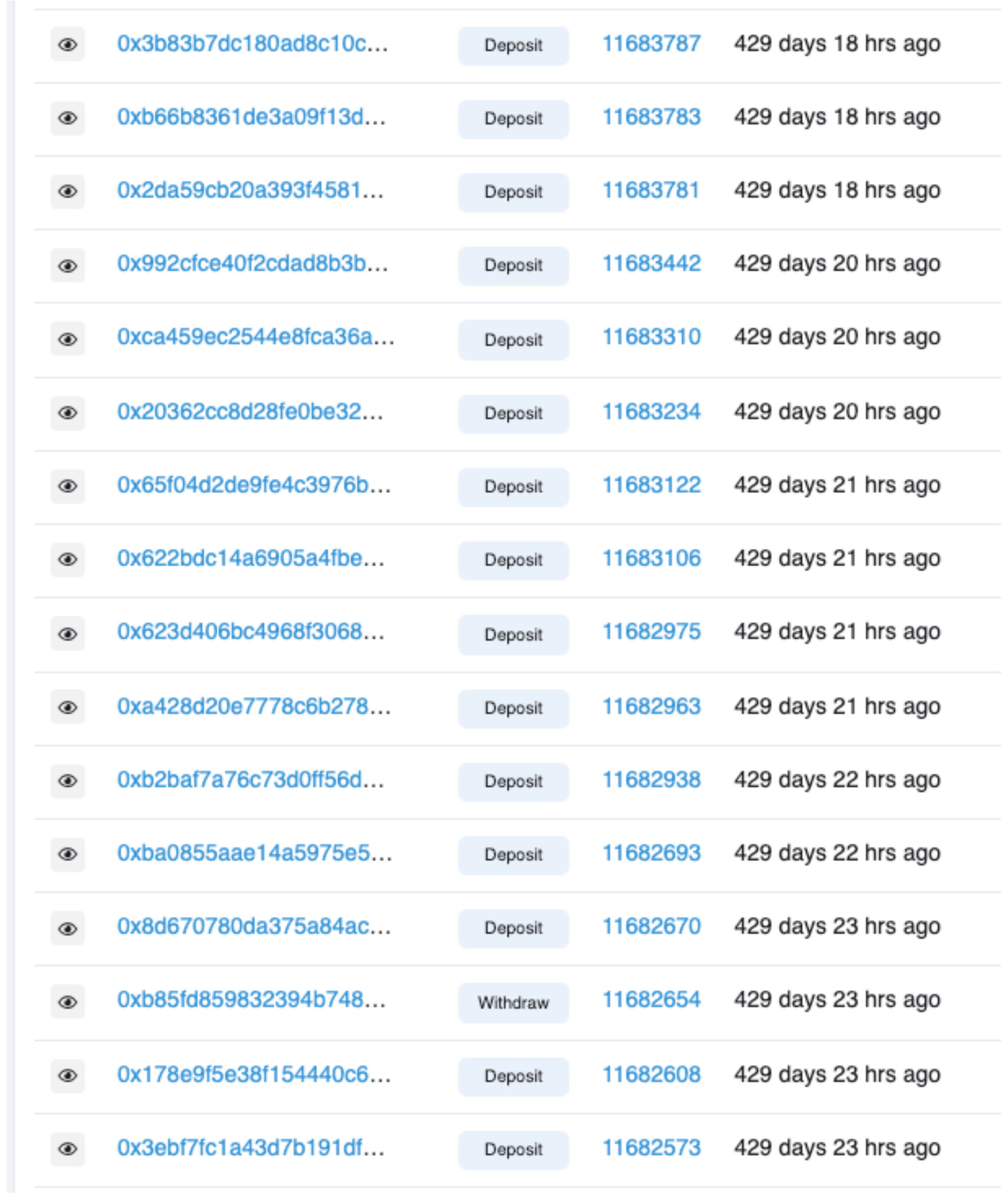
```
{
  vault(id: "0x19d3364a399d251e894ac732651be8b0e4e850
    block: {number: 11682653}) {
    id
    name
    symbol
    inputTokenBalances
    outputTokenSupply
    fees {
      feeType
      feePercentage
    }
    totalVolumeUSD
    totalValueLockedUSD
  }
}
```

```
{
  "data": {
    "vault": {
      "id":
"0x19d3364a399d251e894ac732651be8b0e4e85001",
      "name": "DAI yVault",
      "symbol": "yvDAI",
      "inputTokenBalances": [
        "19852112493894342904669"
      ],
      "outputTokenSupply":
"19852112493894342904669",
      "fees": [
        {
          "feeType": "MANAGEMENT_FEE",
          "feePercentage": "2"
```



```
{
  vault(id: "0x19d3364a399d251e894ac732651be8b0e4e85
    block: {number: 11682653}) {
    id
    sharesSupply
    latestUpdate {
      id
      timestamp
      blockNumber
      balancePosition
    }
  }
}
```

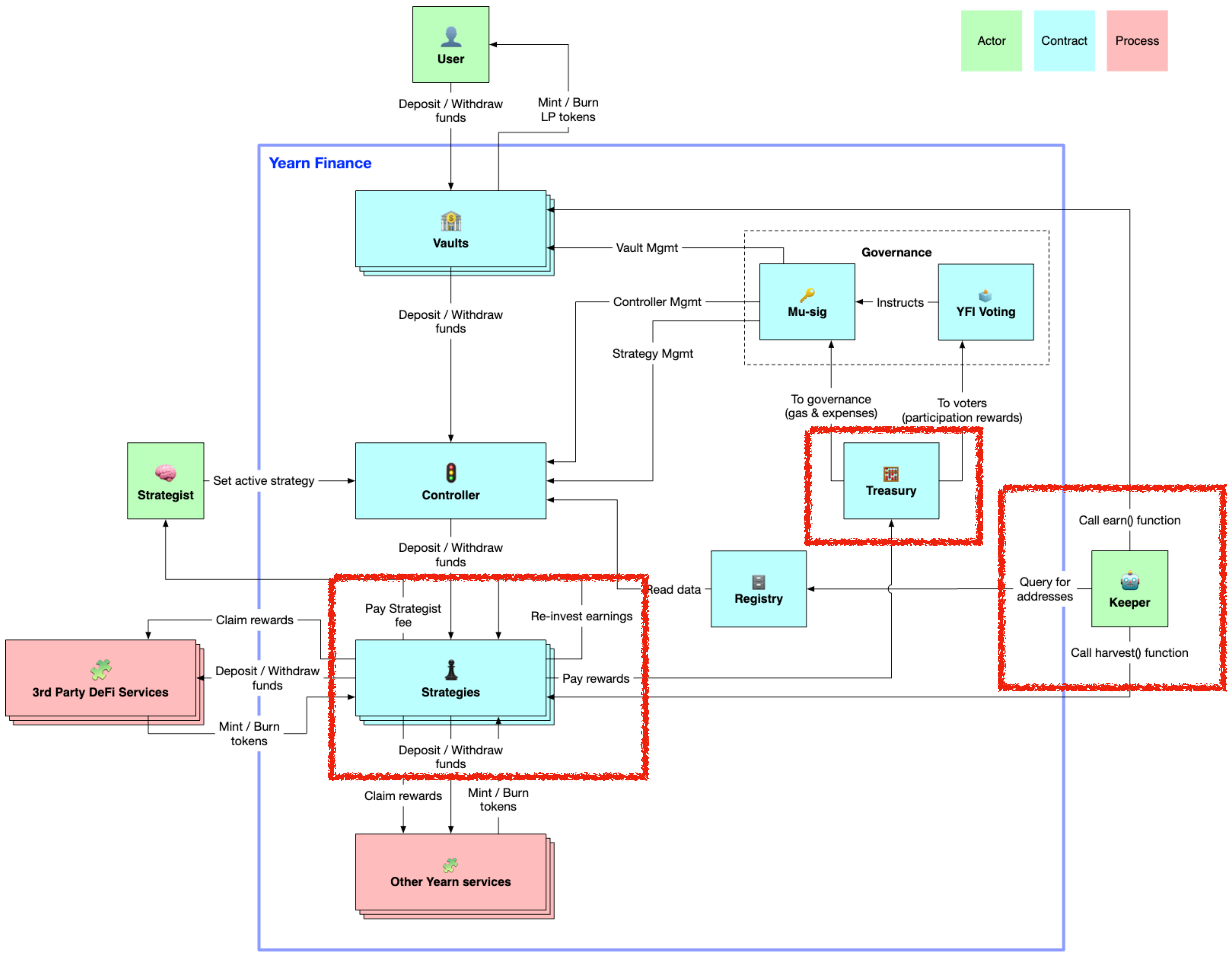
```
{
  "data": {
    "vault": {
      "id":
"0x19d3364a399d251e894ac732651be8b0e4e85001",
      "sharesSupply": "19852112493894342904669",
      "latestUpdate": {
        "id":
"0x19d3364a399d251e894ac732651be8b0e4e85001-
0xcc52363b24eeeeed4175916278e4a0d577e3342052851c
4b47bd81fe2e7695de-196-153",
        "timestamp": "1611017771000",
        "blockNumber": "11682611",
        "balancePosition":
"19850640667042542595521"
      }
```



0x3b83b7dc180ad8c10c...	Deposit	11683787	429 days 18 hrs ago
0xb66b8361de3a09f13d...	Deposit	11683783	429 days 18 hrs ago
0x2da59cb20a393f4581...	Deposit	11683781	429 days 18 hrs ago
0x992cfce40f2cdad8b3b...	Deposit	11683442	429 days 20 hrs ago
0xca459ec2544e8fca36a...	Deposit	11683310	429 days 20 hrs ago
0x20362cc8d28fe0be32...	Deposit	11683234	429 days 20 hrs ago
0x65f04d2de9fe4c3976b...	Deposit	11683122	429 days 21 hrs ago
0x622bdc14a6905a4fbe...	Deposit	11683106	429 days 21 hrs ago
0x623d406bc4968f3068...	Deposit	11682975	429 days 21 hrs ago
0xa428d20e7778c6b278...	Deposit	11682963	429 days 21 hrs ago
0xb2baf7a76c73d0ff56d...	Deposit	11682938	429 days 22 hrs ago
0xba0855aae14a5975e5...	Deposit	11682693	429 days 22 hrs ago
0x8d670780da375a84ac...	Deposit	11682670	429 days 23 hrs ago
0xb85fd859832394b748...	Withdraw	11682654	429 days 23 hrs ago
0x178e9f5e38f154440c6...	Deposit	11682608	429 days 23 hrs ago
0x3ebf7fc1a43d7b191df...	Deposit	11682573	429 days 23 hrs ago

Mapping Events

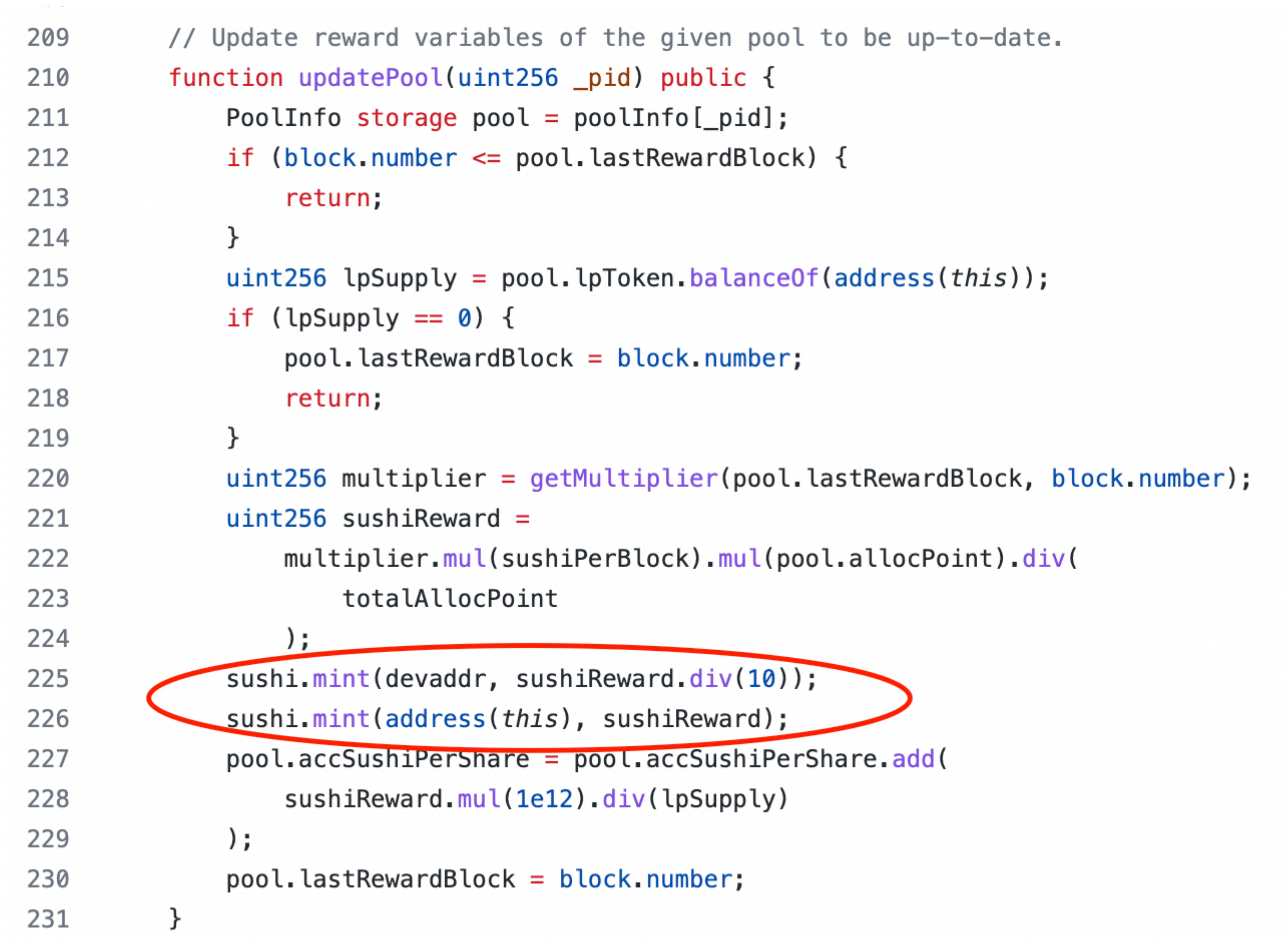
Revenue


Actor
Contract
Process
User
Deposit / Withdraw funds
Mint / Burn LP tokens
Yearn Finance
Vaults
Vault Mgmt
Governance
Controller Mgmt
Mu-sig
Instructs
YFI Voting
Strategy Mgmt
Deposit / Withdraw funds
To governance (gas & expenses)
To voters (participation rewards)
Strategist
Set active strategy
Controller
Treasury
Call earn() function
Deposit / Withdraw funds
Read data
Registry
Query for addresses
Keeper
Pay Strategist fee
Re-invest earnings
Claim rewards
Call harvest() function
3rd Party DeFi Services
Deposit / Withdraw funds
Strategies
Pay rewards
Mint / Burn tokens
Deposit / Withdraw funds
Claim rewards
Mint / Burn tokens
Other Yearn services

Validation

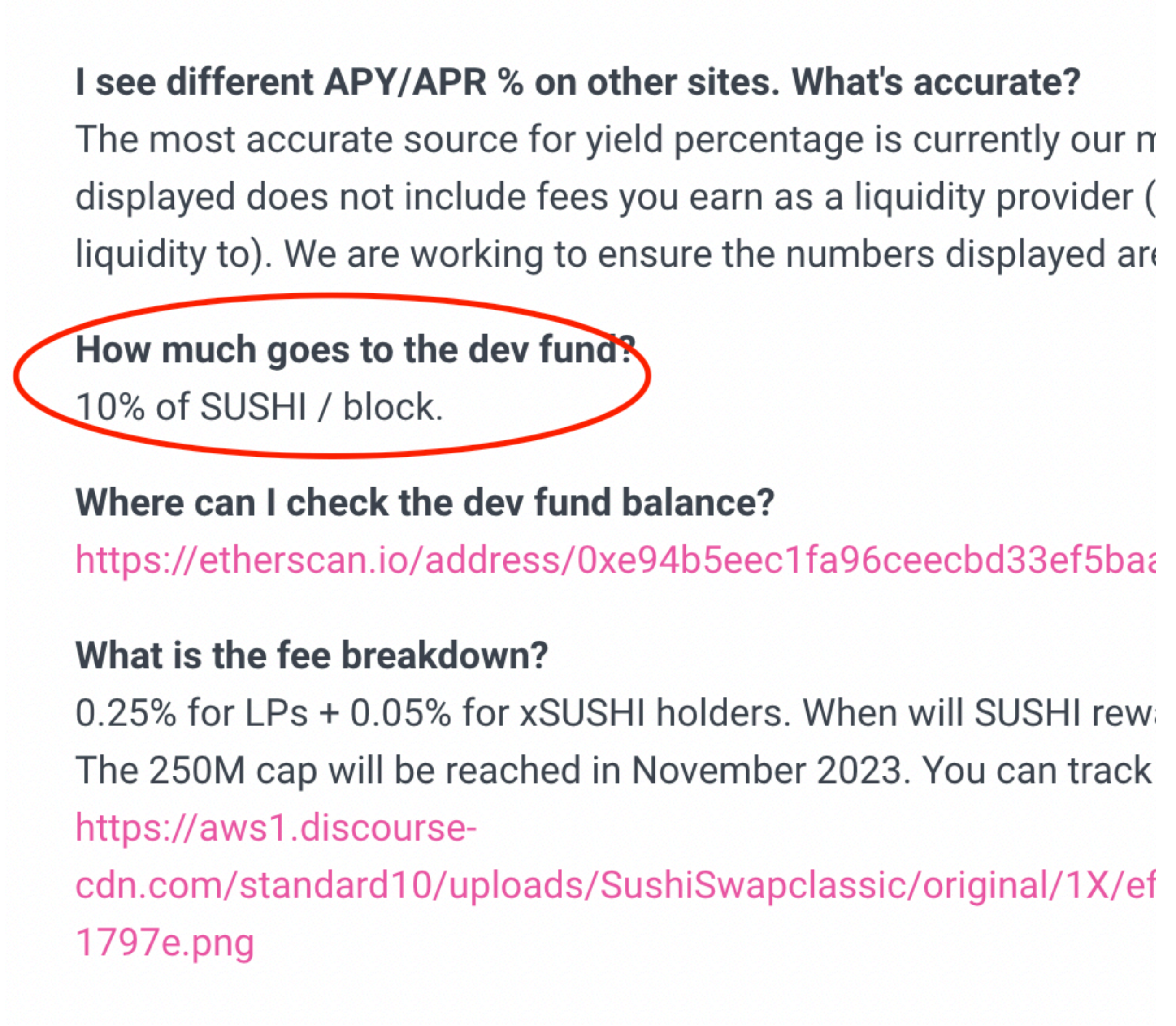
Sources of Errors

- Two main categories:
  - Incorrect data
  - Missing data
- EVM/Solidity quirks
  - ERC20 decimals
  - Failed transactions
- Lack of activity (snapshots)



```solidity
    // Update reward variables of the given pool to be up-to-date.
    function updatePool(uint256 _pid) public {
        PoolInfo storage pool = poolInfo[_pid];
        if (block.number <= pool.lastRewardBlock) {
            return;
        }
        uint256 lpSupply = pool.lpToken.balanceOf(address(this));
        if (lpSupply == 0) {
            pool.lastRewardBlock = block.number;
            return;
        }
        uint256 multiplier = getMultiplier(pool.lastRewardBlock, block.number);
        uint256 sushiReward =
            multiplier.mul(sushiPerBlock).mul(pool.allocPoint).div(
                totalAllocPoint
            );
        sushi.mint(devaddr, sushiReward.div(10));
        sushi.mint(address(this), sushiReward);
        pool.accSushiPerShare = pool.accSushiPerShare.add(
            sushiReward.mul(1e12).div(lpSupply)
        );
        pool.lastRewardBlock = block.number;
    }
```

https://github.com/sushiswap/sushiswap/blob/master/contracts/MasterChef.sol



I see different APY/APR % on other sites. What's accurate?
The most accurate source for yield percentage is currently our n
displayed does not include fees you earn as a liquidity provider (
liquidity to). We are working to ensure the numbers displayed are

How much goes to the dev fund?
10% of SUSHI / block.

Where can I check the dev fund balance?
https://etherscan.io/address/0xe94b5eec1fa96ceecbd33ef5baa

What is the fee breakdown?
0.25% for LPs + 0.05% for xSUSHI holders. When will SUSHI rew
The 250M cap will be reached in November 2023. You can track
https://aws1.discourse-
cdn.com/standard10/uploads/SushiSwapclassic/original/1X/ef
1797e.png

https://docs.sushi.com/faq-1/sushi-nomics-faq

Challenges

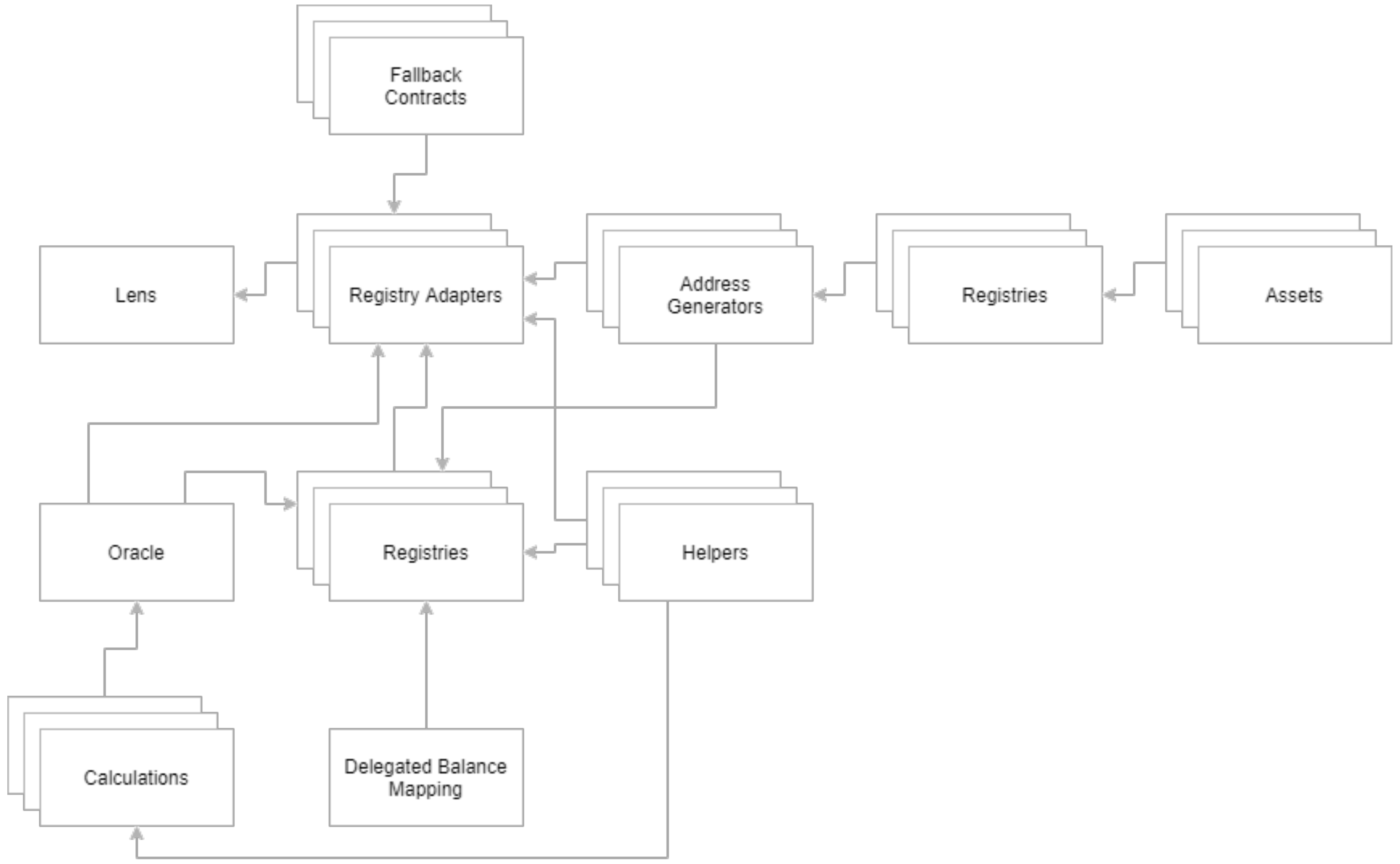
Challenges

- Indexing time; lack of debugging tools
- Protocol history (e.g. proxy upgrades, new releases)
- Bad smart contract design (e.g. missing events, no call return values)
- Price oracle
- Lack of documentation
- Validation

華宮 凪沙 c
メガネスタイル!
失われた未来を求めて

Prices

Yearn Lens


Fallback Contracts
Lens
Registry Adapters
Address Generators
Registries
Assets
Oracle
Registries
Helpers
Calculations
Delegated Balance Mapping