

# Le YAM, comment ça marche...

En complément du guide utilisateur, pour les curieux qui souhaiteraient comprendre comment ça marche (« DYOR »), nous allons regarder ce qui se passe derrière l’affichage de l’application....  
(pas d’inquiétude : aucune compétence en développement n’est nécessaire ;-) ..)

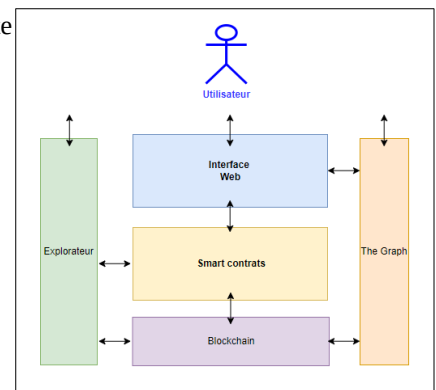
## 1 - Les composants d’une l’application WEB 3 (comme YAM ou RMM)

Les APPlications distribuées (dAPP) fonctionnent sur une blockchain. Elles sont constituées :

- d’une partie Interface Web (Front-end), que vous accédez via l’url du site et sur laquelle vous vous connectez avec votre wallet,
- de smart contrats, cœur de l’application, qui sont enregistrés et qui s’exécutent sur une blockchain (ici Gnosis),
- et dans le cas présent, d’un service d’accès et d’indexation des données sur la blockchain (The Graph).

L’utilisateur peut accéder à l’application :

- soit au travers de son interface Web,
- soit en accédant directement aux smart contrats de l’application (sans passer par l’interface), au moyen d’un explorateur de blockchain.



Pour la Gnosis Chain, deux explorateurs sont possibles : <https://gnosisscan.io/> et <https://gnosis.blockscout.com/> . (c’est bien utile d’en avoir deux, en cas d’indisponibilité d’un des deux..)

Le service d’indexation de données, The Graph, est accessible par l’utilisateur. C’est notamment utile pour accéder à des groupes ou des historiques de transactions. C’est d’un usage assez complexe, juste une brève introduction sera faite en fin de document.

L’accès à l’application, à partir de l’interface, est détaillé dans le guide utilisateur. Dans le présent document, nous allons accéder à l’application sans passer par l’interface et ainsi analyser ce qui se passe aux niveaux inférieurs...

## 2 – Actions sur le smart contrat

Vous pouvez accéder à l’application, sans passer par l’interface, directement auprès des smart contrats : soit pour contourner une défaillance (ponctuelle) de l’interface, soit pour visualiser des informations que l’interface n’afficherait pas, soit pour gagner en rapidité (cas des bots..).

Pour accéder à un smart contrat avec un explorateur, vous allez avoir besoin de son adresse. Chaque smart contrat a une adresse unique, sur la blockchain sur laquelle il est enregistré.

L’adresse du smart contrat YAM est : 0xC759AA7f9dd9720A1502c104DaE4F9852bb17C14. Vous pouvez y accéder avec l’explorateur GnosisScan via l’url suivante :

<https://gnosisscan.io/address/0xC759AA7f9dd9720A1502c104DaE4F9852bb17C14>

L’explorateur donne accès à de multiples informations sur le smart contrat : les différentes transactions exécutées (par type), ainsi que le contrat en lui-même.

Lorsqu'on sélectionne l'onglet « contrat », les informations suivantes apparaissent : L'accès en lecture ou écriture au contrat ou à son proxy...

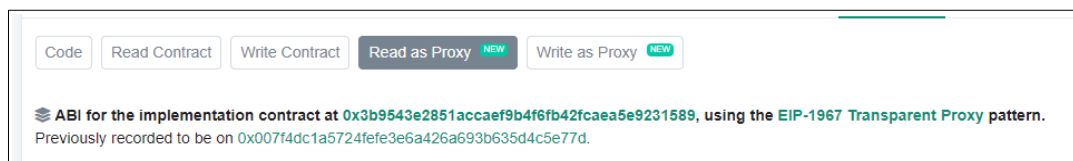


Mais qu'est ce donc qu'un Proxy ?

Les smart contrats, une fois enregistrés à une adresse sur une blockchain, sont non modifiables. C'est une des forces de la blockchain, mais ce peut être aussi une contrainte lorsqu'on développe et qu'on a besoin de faire des mises à jours. Pour ce faire, on utilise la technique du Proxy, qui permet de modifier la logique du contrat, sans changer son adresse et les valeurs qu'il stocke.

Vous avez alors deux contrats :

- Celui du Proxy : dont l'adresse est fixe et qui stocke les valeurs. Il s'agit ici de l'adresse mentionné ci-dessus. Qui devra être utilisée pour toute action sur le contrat.
- Celui de l'implémentation (le logique) du contrat, qui est « masqué » (et accédé) par le Proxy. Son adresse est visible lorsqu'on clique sur Read ou Write « as Proxy », par l'information « ABI for implementation contrat at... ». Cette adresse change au fur et à mesure des mises à jour.

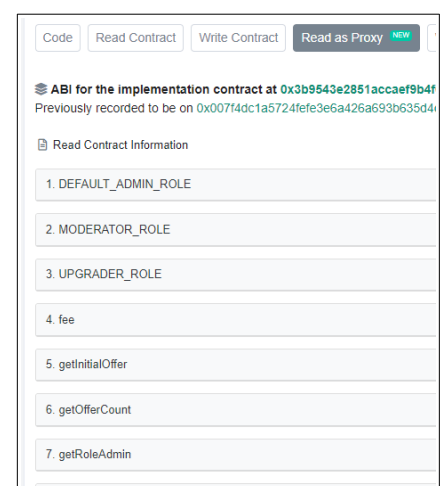


Pour lire la logique du contrat (écrite en Solidity sur Ethereum, ou des blockchains compatibles comme Gnosis) vous devrez donc aller sur cette adresse.

## 2.1 - Actions en lecture sur le contrat YAM

L'ensemble des actions possibles, apparaissent en sélectionnant « Read as Proxy ».

Prenons par exemple les actions 5 et 6 :



6. *getOfferCount* : indique le prochain numéro d'offre disponibles,  
 5. *getInitialOffer* : indique les données de l'offre (en prenant pas exemple la dernière publiée, avec un offer ID = *getOfferCount* - 1 .

Pour comprendre à quoi correspondent les différentes champs données par un ordre de lecture, il va hélas falloir aller dans le code !... Mais vous allez voir, ce code est très bien écrit et il y a plein de commentaires.

Comme mentionné précédemment, pour accéder au code il faut aller dans un autre smart contrat (au moment ou ce document est écrit, il s'agit du contrat ci-après):Il

5. getInitialOffer

offerId (uint256)

38408

Query

↳ address, address, address, address, uint256, uint256

[ getInitialOffer method Response ]

- » address : 0x5CC180BF9091a2284624567ee3c5a2A465656301
- » address : 0x0cA4f5554Dd9Da6217d62D8df2816c82bba4157b
- » address : 0x2DD84475c1165502c7053A906F82e440dF8797Ad
- » address : 0x00
- » uint256 : 5060000000000000000000000000000000
- » uint256 : 1000000000000000000000000000000000

6. getOfferCount

38409 uint256

<https://gnosisscan.io/address/0x3b9543e2851accaef9b4f6fb42fcaea5e9231589#code>

Le smart contrat est constitué de multiples fichiers. Celui dans lequel réside les ordres de lecture est le troisième « IRelaTokenbYAMUpgradeableV3.sol ». En cherchant, par exemple, l'ordre de lecture « GetInitialOffer » on trouve à partir de la ligne 249 le commentaire qui indique la signification de chacun des champs :

```
File 3 of 20 : IRelaTokenYamUpgradeableV3.sol
245     string memory
246     );
247
248     /**
249     * @notice Returns the offer information
250     * @param offerId The offer Id
251     * @return The offer token address
252     * @return The buyer token address
253     * @return The seller address
254     * @return The buyer address
255     * @return The price
256     * @return The amount of the offer token
257     */
258     function getInitialOffer(uint256 offerId)
259         external
260         view
```

```
[ getInitialOffer method Response ]
» address : 0x5CC180BF9091a2284624567ee3c5a2A465656301
» address : 0x0cA4f5554Dd9Da6217d62D8df2816c82bba4157b
» address : 0x2DD84475c1165502c7053A906F82e440dF8797Ad
» address : 0x0000000000000000000000000000000000000000
» uint256 : 5060000000000000000000000000000000
» uint256 : 1000000000000000000000000000000000
```

qui correspondent aux informations que vous pouvez, par exemple, retrouver dans le Telegram des OTC ( <https://t.me/RealTokenOTC> )

RealT OTC

NEW OFFER

PROPERTY #313 APR 2023

NEW LISTING

DETROIT - MICHIGAN

19003-19005 Moross Rd (DUPLX (1943))

QUANTITY 1 #38408

SELLING PRICE 50.60 -0.28%

TOKEN PRICE \$50.74 1,900 TOKENS

EXPECTED INCOME 9.68% \$4.91 / YEAR

NEW YIELD 9.70% APR 2024

19003-19005 Moross Rd Detroit

#38408 1 @ 50.6 aWXXDAI (-0.28%)

18 09:28

La seconde adresse est celle du stablecoin utilisé pour l'offre de vente. Si vous cliquez sur l'adresse, l'explorateur vous donnera le nom de ce token : ici *armmv3XDAI*

Contract 0x0cA4f5554Dd9Da6217d62D8df2816c82bba4157b

Contract Overview

Balance: 0 xDAI

xDAI Value: \$0.00

Token: \$506,127.07

More Info

My Name Tag: Not Available, login to update

Contract Creator: 0xde2e13f3228a8c767c... at bn 0x83b12e181412e

Token Tracker: RealT RMM V3 WXDAI (armmv3WXD...)

En cliquant sur le nom du Token, vous allez trouver une autre information importante : le nombre de décimal de ce token : ici 18

**Token RealIT RMM V3 WXDAI**

**Overview** ERC20

PRICE	FULLY DILUTED MARKET CAP
\$0.00 @ 9.000000 xDAI	\$0.00
Total Supply:	3,184,410.879626 <b>armmv3WXDAI</b>
Holders:	1,757 addresses

**Profile Summary**

Contract: 0x9D9a6217d62D8d782816c82bba4157b

Decimals: 18

Social Profiles: Not Available, Update ?

L'avant dernier champ, affiché dans `5. getInitialOffer`, est le montant en `armmv3XDAI`. Pour convertir le montant affiché (ici : `5060000000000000000`), il faut le diviser par 10 puissance 18 (nombre de décimal du token) ce qui fera 50,6.

Attention, tous les token n'ont pas les mêmes décimales :

- WXDAI : 0xe91D153E0b41518A2Ce8Dd3D7944Fa863463a97d – Decimal : 18
- USDC : 0xDDAfb505ad214D7b80b1f830fcCc89B60fb7A83 – Decimal : 6
- Armmv3XDAI : 0x0cA4f5554Dd9Da6217d62D8df2816c82bba4157b – Decimal : 18
- ArmmV3USDC : 0xeD56F76E9cBC6A64b821e9c016eAFbd3db5436D1 - - Decimal : 6
- Realtoken : .... Decimal : 18

Parmi tous les ordres de lecture, deux autres sont à voir :

- 13. *showOffer*, semblable à 5. *getInitialOffer* mais pour l'offre en cours,
- 15. *tokenInfo*, qui donne des infos sur l'adresse d'un RealToken

15. tokenInfo

tokenAddr (address)

0xc5CC180BF9091a2284624567ee3c5a2A465656301


Query

↳ `uint256, string, string`


[ tokenInfo method Response ]  
➤ `uint256`: 18  
➤ `string`: REALTOKEN-S-19003-19005-MOROSS-RD-DETROIT-MI  
➤ `string`: RealToken S 19003-19005 Moross Rd Detroit MI

## 2.1 - Actions en écriture sur le contrat YAM


Prenons comme exemple, l'achat de tout ou partie de l'offre 37809 :




**NEW OFFER**



PROPERTY  
**#455**  
JAN 2024



TOKEN PRICE  
**\$ 50.73**  
12,800 TOKENS




EXPECTED INCOME  
**8.03%**  
\$ 4.07 / YEAR


BOQUETE - CHIRIQUI

**Lucero Club 1R4**


SINGLE FAMILY (2009)




QUANTITY  
**1**  
#38237





SELLING PRICE  
**\$ 50.70**  
-0.06%



NEW YIELD  
**8.03%**  
APR 2024

 **Lucero Club 1R4 Boquete**

 **#38237 1 @ 50.7 USDC (-0.06%)**

 **#37809 1 @ 50.7 USDC (-1.44%)**

```
13. showOffer

offerId (uint256)

137809

Query

↳ address, address, address, address, uint256, uint256

[showOffer method Response]
>> address: 0x1Ba3C4502CA1E965a26F30fd1512A0800fb7A5B
>> address: 0x0ADa6b605ad214d708b61f830fcC89660fb7A5B
>> address: 0x6bf924388c74A1ea1aD017EaD8F4955A3f0eAd6
>> address: 0x0000000000000000000000000000000000000000000000000000
>> uint256: 50000000
>> uint256: 10000000000000000000000000000000000000000000000000000
```

Lors d'un achat, avec l'application front-end, le wallet demande deux autorisations : la première pour que le contrat YAM puisse accéder à vos stablecoin et la seconde pour exécuter l'ordre dans la quantité souhaitée.

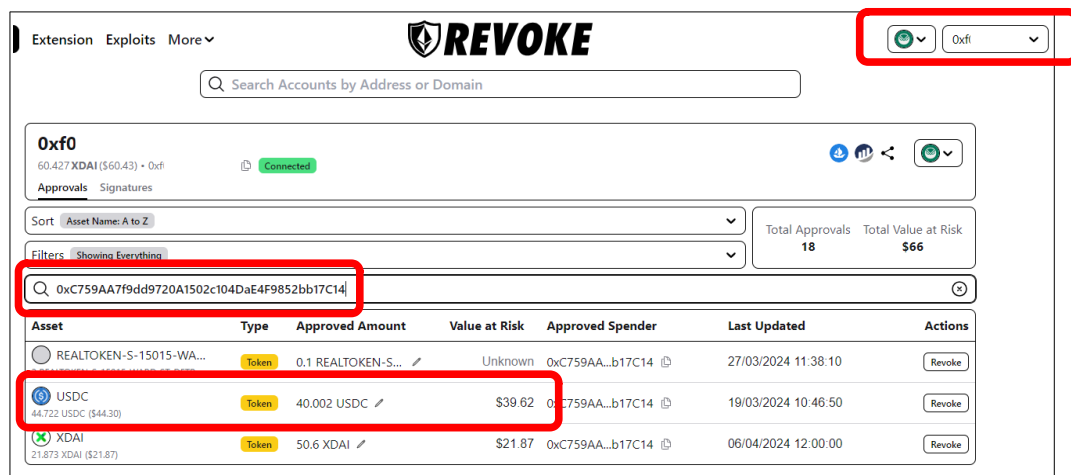
En agissant directement sur le smart contrat YAM, nous n'allons donner que la seconde autorisation (celle sur le contrat YAM, puisque c'est sur celui-ci que nous serons connecté).

Il faut donc qu'au préalable, nous ayons donné la première autorisation...

### Allowance :

Dans l'exemple, l'offre s'achète en USDC, il faut donc au préalable que le contrat YAM ait été autorisé à dépenser les USDC de votre wallet, pour un montant égal ou supérieur à votre achat.

Pour cela, vous pouvez aller vérifier dans l'application <https://revoke.cash/> si l'autorisation est présente :



Connecter votre wallet, rechercher les autorisations données au contrat YAM (avec son adresse) et vérifier si l'USDC est listé. Dans l'image ci-dessus, votre achat ne pourra dépasser 40 USDC.

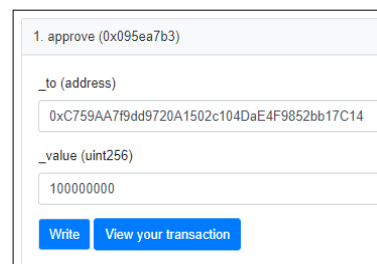
Si aucun USDC n'est listé ou que le montant n'est pas suffisant, il va falloir aller dans le contrat USDC pour faire un approve.

- <https://gnosisscan.io/address/0xddafbb505ad214d7b80b1f830fccc89b60fb7a83#writeProxycontrat>

- Connecter le wallet à autoriser :



- Utiliser la fonction *1. approve*, avec l'adresse du contrat YAM et la quantité de l'approbation (en décimal 6)



Dans revoke, l'approbation est maintenant passé de 40 à 100

0xC759AA7f9dd9720A1502c104DaE4F9852bb17C14					
Asset	Type	Approved Amount	Value at Risk	Approved Spender	
REALTOKEN-S-15015-WA...	Token	0.1 REALTOKEN-S...	Unknown	0xC759AA...b17C14	
USDC 1.222 USDC (\$1.21)	Token	100 USDC	\$1.21	0xC759AA...b17C14	

Nous pouvons maintenant aller acheter sur le contrat YAM :

- <https://gnosisscan.io/address/0xC759AA7f9dd9720A1502c104DaE4F9852bb17C14#writeProxycontrat>
- connecter le wallet, et compléter la fonction 1. buy :
  - avec le numéro de l'offre,
  - le prix (afin d'assurer qu'il n'y a pas eu de modification juste avant l'achat)
  - la quantité en décimal 18  
Attention a ne pas faire d'erreur sur le nombre de 0, car un de plus c'est 10 fois plus acheté !..  
(sur l'image l'achat est de 0,1 token)

1. buy (0x40993b26)

offerId (uint256)

37809

price (uint256)

50000000

amount (uint256)

1000000000000000000

Write View your transaction

### 2.1.2 - Création d'une offre

- fonction : 4. *CreateOffer*, avec
- l'adresse du token vendu
- l'adresse du stablecoin d'échange,
- l'adresse de l'acheteur (pour un achat en mode privé), sinon 0
- le prix  
Attention comme le prix est en USDC , il est exprimé en decimal 6 (ici 80 USDC)
- le nombre de token vendu  
Là c'est exprimé en decimal 18 (ici 0,1 token)

4. createOffer (0x2befd4ad)

offerToken (address)

0x4d6876Bd8d2f44d64b13A83a58CeF81860329ef4

buyerToken (address)

0xDDAfb505ad214D7b80b1f830fcCc89B60fb7A83

buyer (address)

0x00

price (uint256)

80000000

amount (uint256)

1000000000000000000

Write

A titre de comparaison, sur l'application Front-End, la création de la même offre se présenterait comme cela :  
C'est évidemment plus simple et plus sécurisé (grâce aux deux protections)

VENTE Créer votre offre 10%

Jeton en vente

15015 Ward

Jeton d'achat \*

USDC

Prix d'achat en \$ \*

80

Prix d'achat en USDC

80.000057

Le prix de vente officiel est de 50.63 \$USD. Mais vous êtes libre de le choisir.

Avec USDC = 0.99999928\$

Utiliser le ratio de conversion du jeton USDC

Solde du portefeuille

15015 Ward

2.0000000000

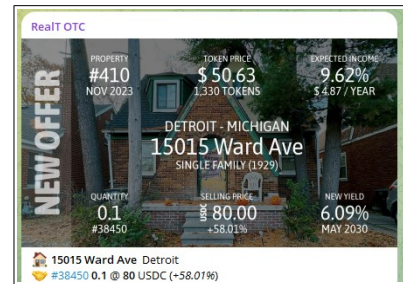
Quantité \*

0.1

Je veux créer une offre privée

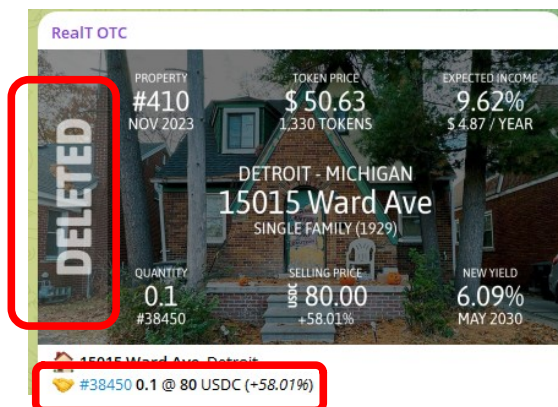
Vous souhaitez vendre jusqu'à 0.1 "15015 Ward" avec un prix unitaire de 80 "USDC" et pour un total de 8.000000 "USDC".

Une fois approuvée, l'offre est visible sur le Telegram OTC



### 2.1.3 Effacer une offre

Là rien de plus simple, puisqu'il y a juste à indiquer le numéro de son offre



7. deleteOffer (0x74268ff2)

offerId (uint256)

38450

Write

Effacer un ensemble d'offres : avec la fonction *8.deleteOfferBatch* et en faisant se succéder les numéros d'offres, sans oublier les crochets :

8. deleteOfferBatch (0xa55f1df0)

offerIds (uint256[])

[10001,10002,10003]

Write

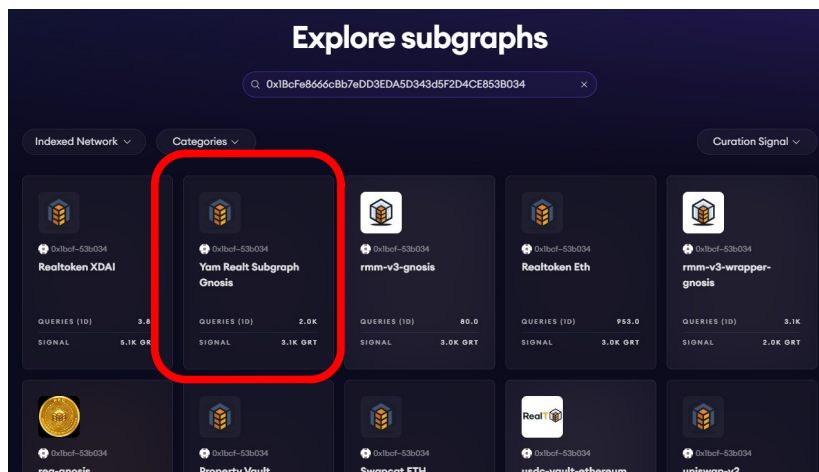
Les autres types d'ordres sont sur le même principe et ne seront pas détaillés.

### 3 - Services The Graph

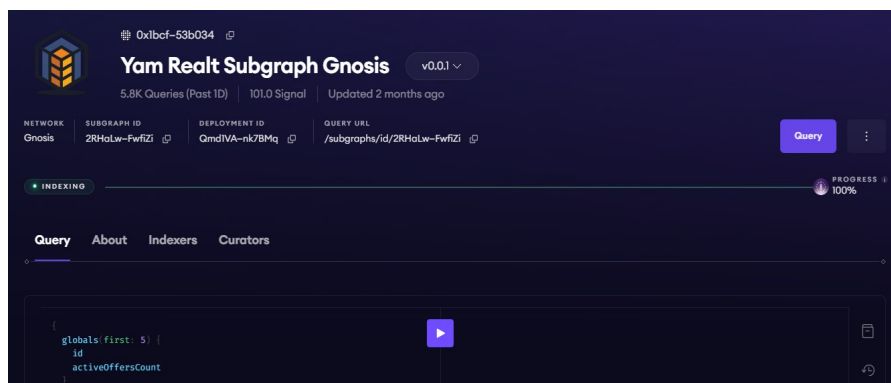
The Graph est un protocol (/service) pour « simplifier » l'accès aux données des blockchains. La simplification est surtout au niveau des applications, car : en une seule requête, on peut accéder à un ensemble d'informations (par ex liste des offres sur YAM) qui autrement aurait réclamé de multiples accès au smart contrat (option plus lente et couteuse).

RealT utilise TheGraph et plusieurs Subgraph sont disponibles en relation avec les applications qu'ils ont développées. Les Subgraph précédemment disponibles sur le service « TheGraph Hosted » ont été migrés sur la solution décentralisée de TheGraph, depuis le 12 juin 2024.

La liste des subgraph RealT : <https://thegraph.com/explorer?page=2&orderBy=Curation+Signal&orderDirection=desc&search=0x1BcFe8666cBb7eDD3EDA5D343d5F2D4CE853B034>

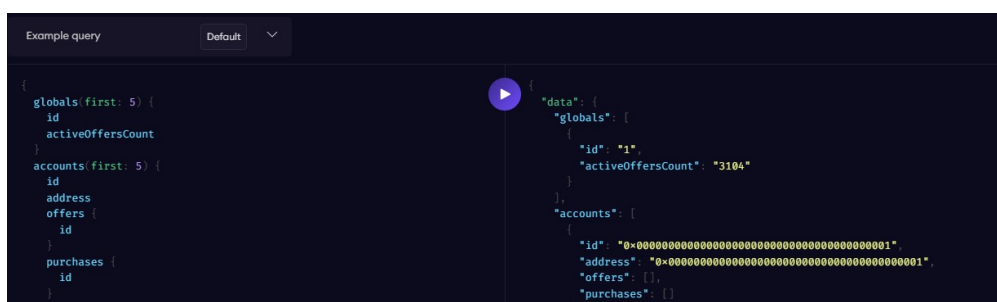


Le second étant celui pour YAM :



L'interrogation des données se fait au travers de requêtes en GraphQL :

Dans l'explorer (onglet Query) : elles sont écrites sur la gauche, en cliquant sur la flèche central (violette) le résultat apparaît sur la droite.





Les résultats sont présentés au format JSON : plutôt adaptés à des programmes, qu'à des humains, quand les ils sont nombreux..

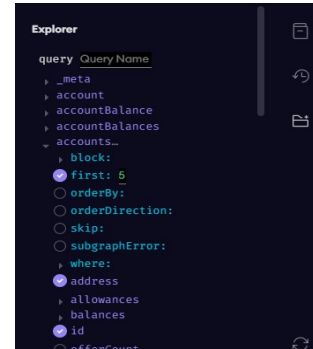
Pour explorer les données et fabriquer sa requête, vous disposez d'un outils sur la droite :



Vous voyez alors apparaître l'ensemble des données interrogeables (celles cochées correspondent à la requête existante).

La requête se constitue donc au fur et à mesure de ce que vous cochez ou décochez des données souhaitées.

Toute la difficulté étant de savoir à quoi correspondent chacune des données pour le YAM (il n'existe hélas pas de dictionnaire ....).



Les services The Graph peuvent être accédés : « manuellement » comme décrit ci-avant, mais le plus souvent ce sera via une application.

Cette application peut être un simple tableur, au moyen de script ou de connecteurs :

- L'accès se fait via une url, indiquée dans la page du Subgraph.
- L'accès est maintenant sécurisé par une clé API qui s'obtient avec l'application Studio (<https://thegraph.com/studio/apikeys/>) et qu'il faudra inclure dans l'url.
- le script convertissant les résultats, du format Json en format tableur.

Exemple pour les liquidations du RMM :

<https://docs.google.com/spreadsheets/d/1K7UY-IYJ-Cs8-YF9oGgCUyBq84Cj7XEYc7mljOPXWK4>