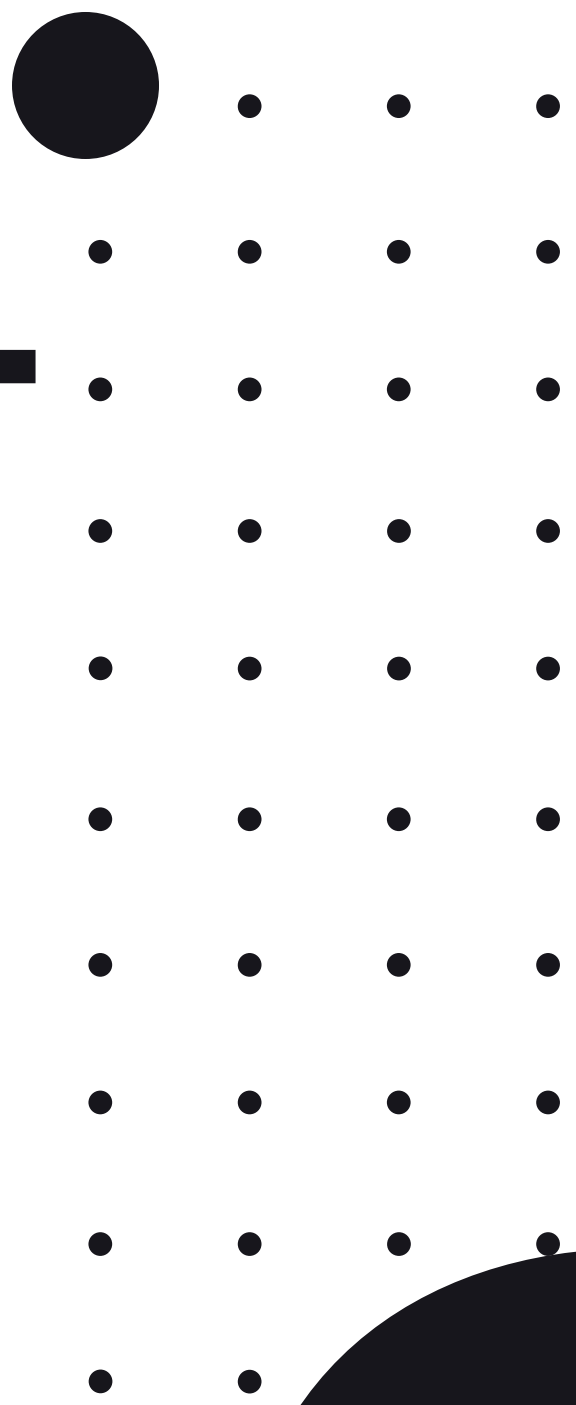


РЭР VALIDATOR

PHYTECH BUSINESS SOLUTIONS CASE CHAMPIONSHIP

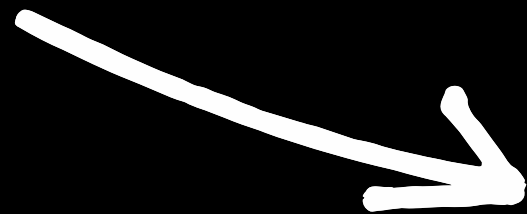
ВЫБОР ОДИН.
СОЗДАНИЕ АЛГОРИТМА КЛАСТЕРИЗАЦИИ СМАРТ-КОНТРАКТОВ



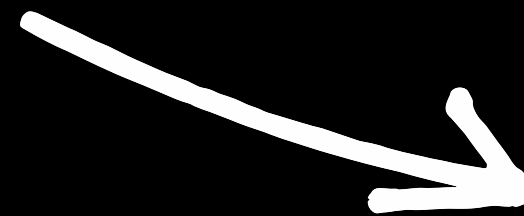
КОМАНДА
ФОРМУЛА
БУДУЩЕГО

ПРОБЛЕМА

**ОТСУТСТВИЕ ОБЪЕДИНЕНИЯ
СМАРТ-КОНТРАКТОВ В ПРОЕКТЫ**



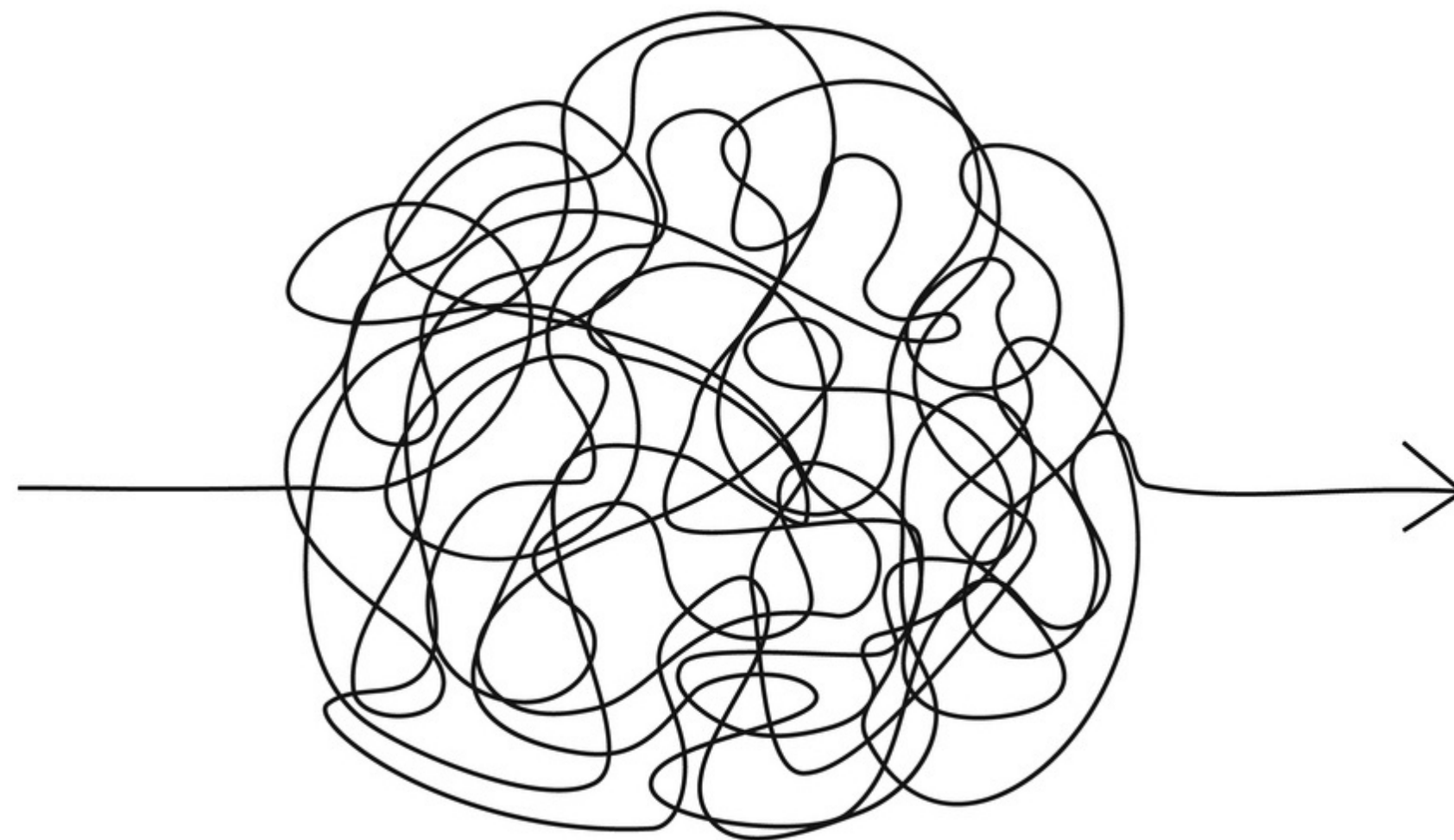
**СЛОЖНОСТЬ ПОНИМАНИЯ
МЕХАНИЗМОВ ТРАНЗАКЦИЙ**



**СТРАДАНИЯ
АНАЛИТИКОВ**

ВЫЗОВЫ
ТРАНЗАКЦИИ

АЛГОРИТМ КЛАСТЕРИЗАЦИИ
ФОРМУЛЫ БУДУЩЕГО



СГРУППИРОВАННЫЕ
СМАРТ-КОНТРАКТЫ

НАШ АЛГОРИТМ

- ОСНОВАН НА МЕТАДАННЫХ СМАРТ-КОНТРАКТОВ
- АНАЛИЗИРУЕТ ИХ ИСХОДНЫЙ КОД
- УЧИТЫВАЕТ МНОГООБРАЗИЕ РАЗРАБОТЧИКОВ БЛОКЧЕЙН-ИНДУСТРИИ
- ОТТАЛКИВАЕТСЯ ОТ СТАТИСТИЧЕСКИХ ЗАКОНОМЕРНОСТЕЙ

ДЛЯ КАЖДОГО СМАРТ-КОНТРАКТА МЫ ПРЕДЛАГАЕМ ПРОВЕРЯТЬ

Наличие у него Public Name Tag на Etherscan

Токены

Имена методов

Закомментированные строки кода

Почему сложно определить
принадлежность смарт-
контракта к проекту?

Потому что в сети разработчики
не подписывают свой код.

.....

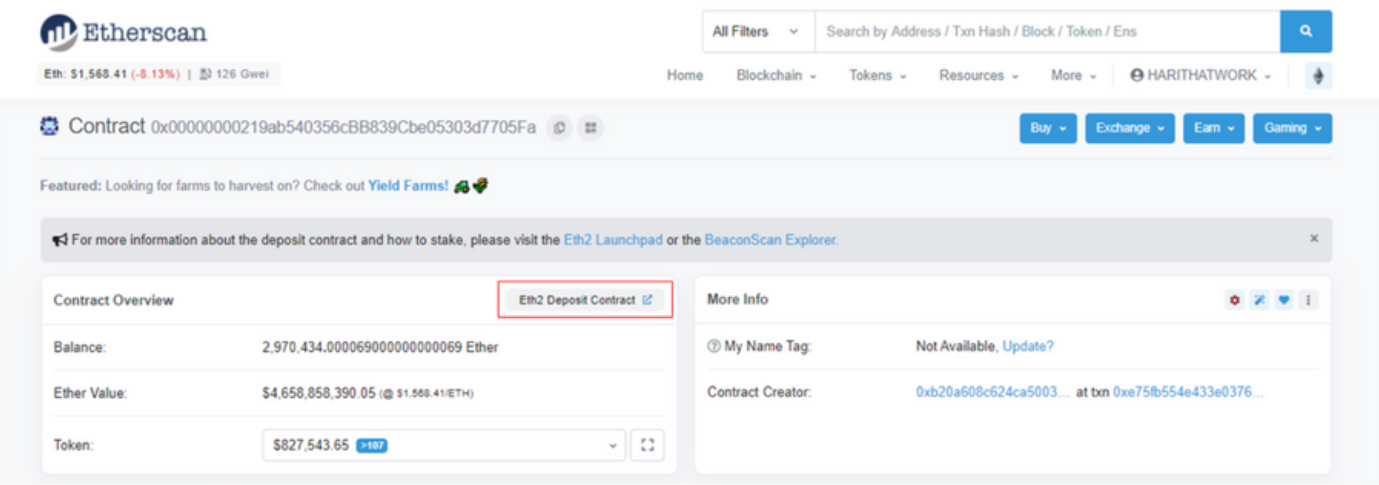
Или всё-таки подписывают?

I. Etherscan Public Name Tag

С помощью него Etherscan даёт
возможность проектам пометчать свои
контракты

Это гарантированно верное
отождествление контракта с компанией,
которая его создала

Public Name Tags indicates the owner of an address in the case of human-owned or Externally Owned
Addresses ([EOA](#)) and the owner or purpose of an address in the case of contract addresses.



Name Tags can refer to an organization or an individual. In most cases, it comes with a URL link that points to the source of the Name Tag. Sometimes, we'll also add a public note on top for announcements or warnings related to the address.

public_tags			
	url	tag	tag_link
7f12f8b7fc532	https://etherscan.io/address/0xb4e16d0168e52d35cacad2c6185b44281ec28c9dc	Uniswap V2: USDC	https://v2.info.uniswap.org/pair/0xb4e16d0168e52d35cacad2c6185b44281ec28c9dc
7f12f8b7fc532	https://etherscan.io/address/0xb1cd6e4153b2a390cf00a6556b0fc1458c4a5533	Bancor: ETHBNT Token	https://www.bancor.network/
7f12f8b7fc532	https://etherscan.io/address/0x0d4a11d5eeaac28ec3f61d100daf4d40471f1852	Uniswap V2: USDT	https://v2.info.uniswap.org/pair/0x0d4a11d5eeaac28ec3f61d100daf4d40471f1852
7f12f8b7fc532	https://etherscan.io/address/0x2af7ea6cb911035f3eb1ed895cb6692c39ecba97	InstaDApp: Event	https://instadapp.io/
7f12f8b7fc532	https://etherscan.io/address/0x11111125434b319222cdbf8c261674adb56f3ae	1inch Network v2	https://app.1inch.io/#/r/0xb82F564B5E59041DaF46909dB12eA4858D1530A5
B413ef929a5fbc	https://etherscan.io/address/0xaddc3e67a500f7037cd622b11df291a6351bfb64		
B413ef929a5fbc	https://etherscan.io/address/0xf7a232044c586d658f826e1932c37cad770b0687		
B413ef929a5fbc	https://etherscan.io/address/0x72012ada1d0a9e9cba22b3f13f3bc52b57ed6b66		
B413ef929a5fbc	https://etherscan.io/address/0xfd8a76dc204e461db5da4f38687adc9cc5ae4a86		
B413ef929a5fbc	https://etherscan.io/address/0x73d2f81fcea9832fc9ee90521abde1150f6b52a		
B413ef929a5fbc	https://etherscan.io/address/0x612447e8d0bdb922059ce048bb5a7cef9e017812		
B413ef929a5fbc	https://etherscan.io/address/0xe7f4c89032a2488d327323548ab0459676269331		
B413ef929a5fbc	https://etherscan.io/address/0xd9e1ce17f2641f24ae83637ab66a2cca9c378b9f	SushiSwap: Router	https://app.sushi.com/swap
B413ef929a5fbc	https://etherscan.io/address/0xc02aaa39b223fe8d0a0e5c4f2ead9083c756cc2	Wrapped Ether	https://weth.io/
B413ef929a5fbc	https://etherscan.io/address/0x2216d47494e516d8206b70fca8585820ed3c4946		

Мы спарсили страницу каждого контракта из рабочего датасета и
уверенно идентифицировали 30% адресов

II. Токены

Bitquery API позволяет различать смарт-контракты токенов. Их имена часто содержат полное или краткое название проекта.

Однако мы должны отличать простые токены, участвующие во многих других проектах:

- 1) Они не вызываются через delegate call.
- 2) Мы знаем их имена :)

address	annotations	currencies
https://etherscan.io/address/0x64eda51d3ad40d56b9dfc5554e06f94e1dd786fd		Curve.fi tBTC/sbtcCrv
https://etherscan.io/address/0x6def55d2e18486b9ddfaa075bc4e4ee0b28c1545		Badger Sett Curve.fi renBTC/wBTC
https://etherscan.io/address/0x63cf44b2548e4493fd099222a1ec79f3344d9682		-
https://etherscan.io/address/0x2a8facc9d49fbc3ecff569847833c380a13418a8		-
https://etherscan.io/address/0x93054188d876f558f4a66b2ef1d97d16edf0895b		-
https://etherscan.io/address/0x444b860128b7bf8c0e864bdc3b7a36a940db7d88		-
https://etherscan.io/address/0x49849c98ae39ff122806c06791fa73784fb3675	Curve.fi renBTC/wBTC (crvRenWBTC), Curve.fi	Curve.fi renBTC/wBTC
https://etherscan.io/address/0x72012ada1d0a9e9cba22b3f13f3bc52b57ed6b66		-
https://etherscan.io/address/0x77655099f72484fa7d8cd701112c129b35b6ca6a		xWAIFUWETH
https://etherscan.io/address/0xb91bca4c6a607448a093803b3b2a9a4ed3e9f71e		xWAIFU
https://etherscan.io/address/0xbd4765210d4167ce2a5b87280d9e8ee316d5ec7c		-
https://etherscan.io/address/0x4e977830ba4bd783c0bb7f15d3e243f73ff57121		Aave stable debt bearing WETH
https://etherscan.io/address/0xd9e1ce17f2641f24ae83637ab66a2cca9c378b9f	Router, SushiSwap	-
https://etherscan.io/address/0xc02aaa39b223fe8d0a0e5c4f27ead9083c756cc2	https://weth.io/ , 0x Ecosystem, WrappedEther, Stableco	Wrapped Ether
https://etherscan.io/address/0x2216d47494e516d8206b70fca8585820ed3c4946		Waifus



III. Имена методов

Благодаря Bitquery API и GraphQL выгружаем список методов контракта

	-	Generic	withdrawToken, Contract Creation, depositEther
0345dbdf3a34d4266cf5ebe9	Uniswap V2	DEX	approve, transferFrom, mint, balanceOf, transfer, burn, getReserves, token1, token0, swap, Cont
f23dc50a7bf38a8d6b233f6		Token	approve, transferFrom, symbol, balanceOf, transfer, burn, allowance, decimals, Contract Creation
	-	Generic	exitTokens, lockTokens, grantRole, Contract Creation, initialize, renounceRole

В приведённом примере контракт без метки имеет схожие с контрактом Uniswap названия функций. Действительно, ручной анализ показал, что они относятся к одному проекту (и конечно, кластеру в датасете P2P).

Таким образом, мы можем сопоставлять контракты по синтаксису, который использовался разработчиком. Делать это можно, например, с помощью библиотек SpaCy и NLTK.

IV. Закомментированные строки кода

Недостающая информация о контракте может быть заполнена благодаря тому, что иногда разработчик упоминает имя проекта в комментариях. Как правило оно находится в одной строке со словами "author", "copyright", "developed" и тп

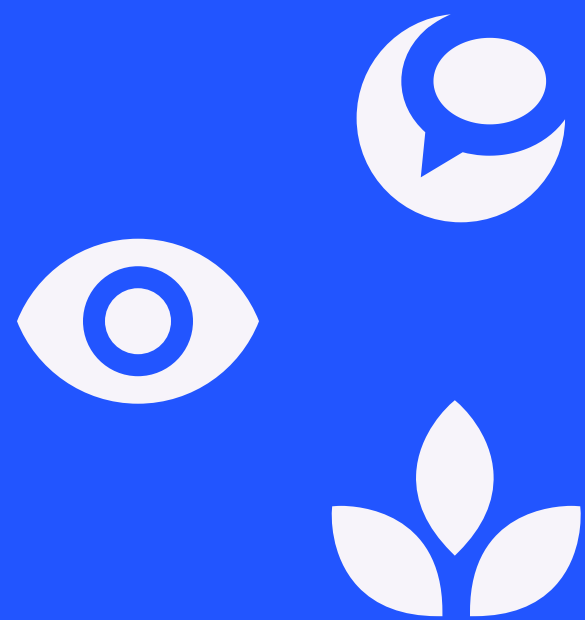
 **Contract Source Code** (Solidity)

Outline ▾

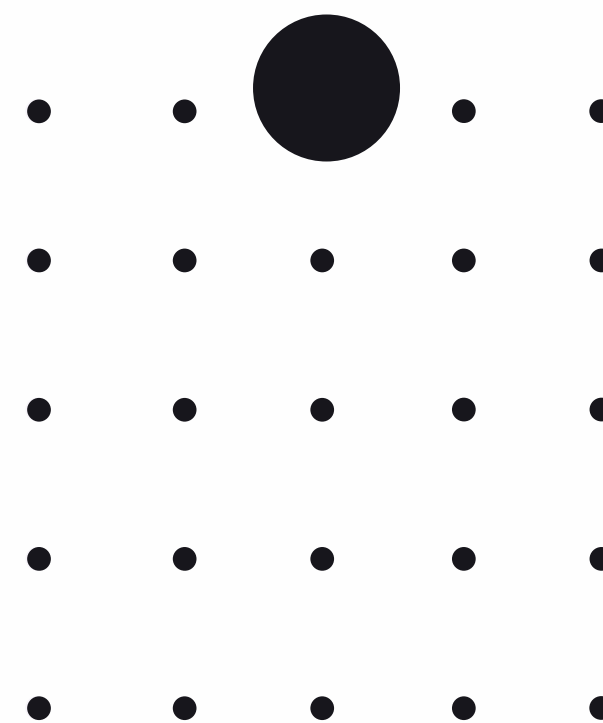
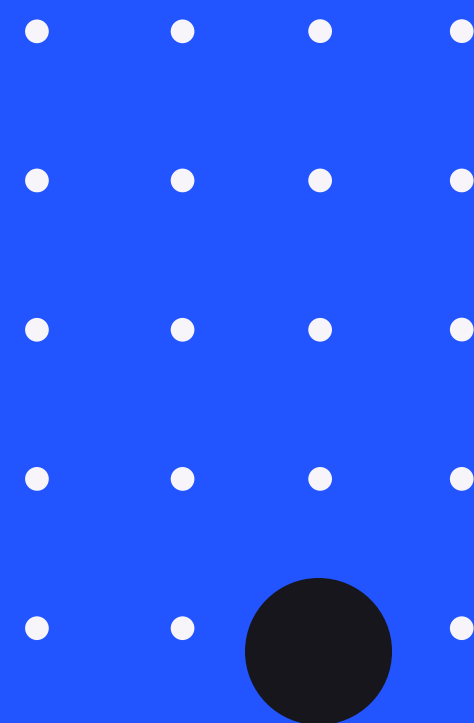
More Options ▾



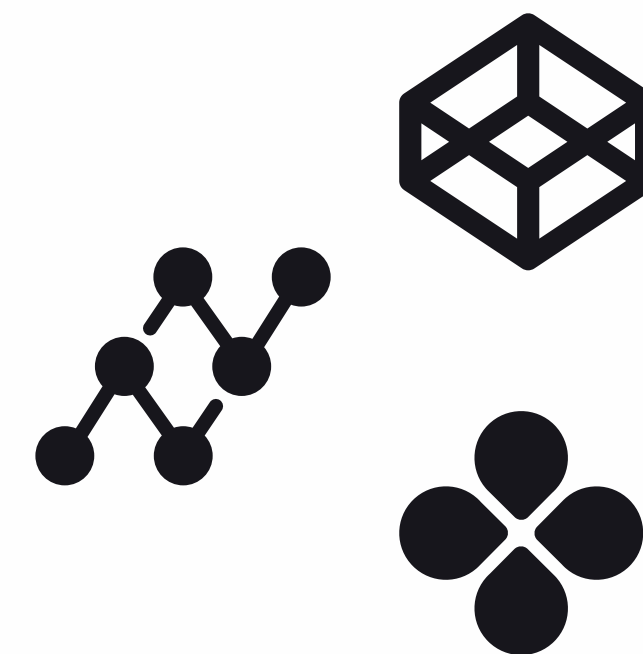
```
190     * Reverts when dividing by zero.
191     */
192     function mod(uint256 a, uint256 b) internal pure returns (uint256) {
193         require(b != 0);
194         return a % b;
195     }
196 }
197
198 // File: contracts/protocol/lib/Require.sol
199
200 /**
201  * @title Require
202  * @author dYdX
203  *
204  * Stringifies parameters to pretty-print revert messages. Costs more gas than regular require()
205  */
206 library Require {
207
208     // ===== Constants =====
209
210     uint256 constant ASCII_ZERO = 48; // '0'
211     uint256 constant ASCII_RELATIVE_ZERO = 87; // 'a' - 10
212     uint256 constant ASCII_LOWER_EX = 120; // 'x'
213     bytes2 constant COLON = 0x3a20; // ': '
214     bytes2 constant COMMA = 0x2c20; // ', '
215     bytes2 constant L_PAREN = 0x2820; // '('
```



Данные собрали.
А что с ними
делать?



Сопоставлять
их со списком
блокчейн-
проектов!

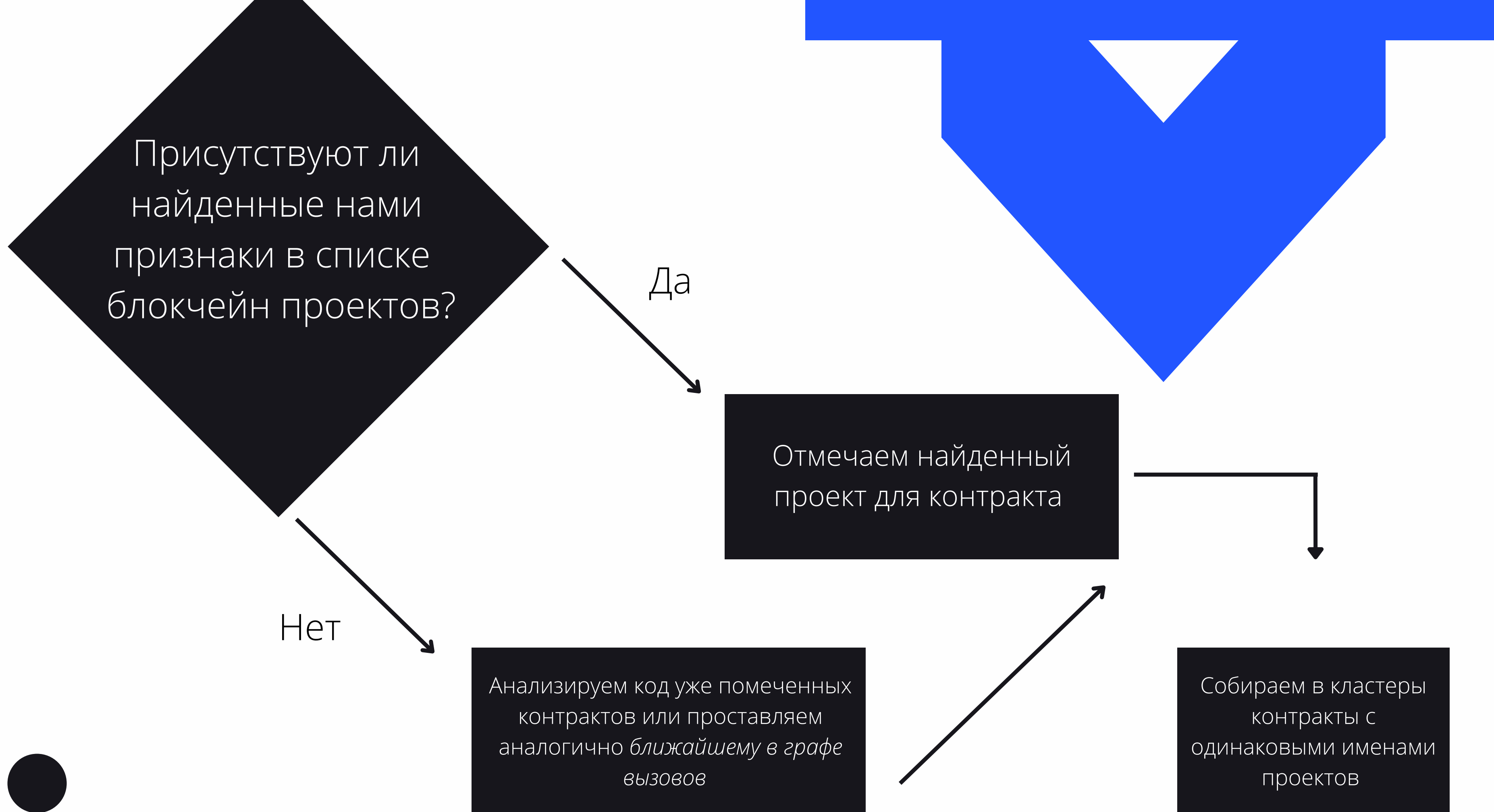


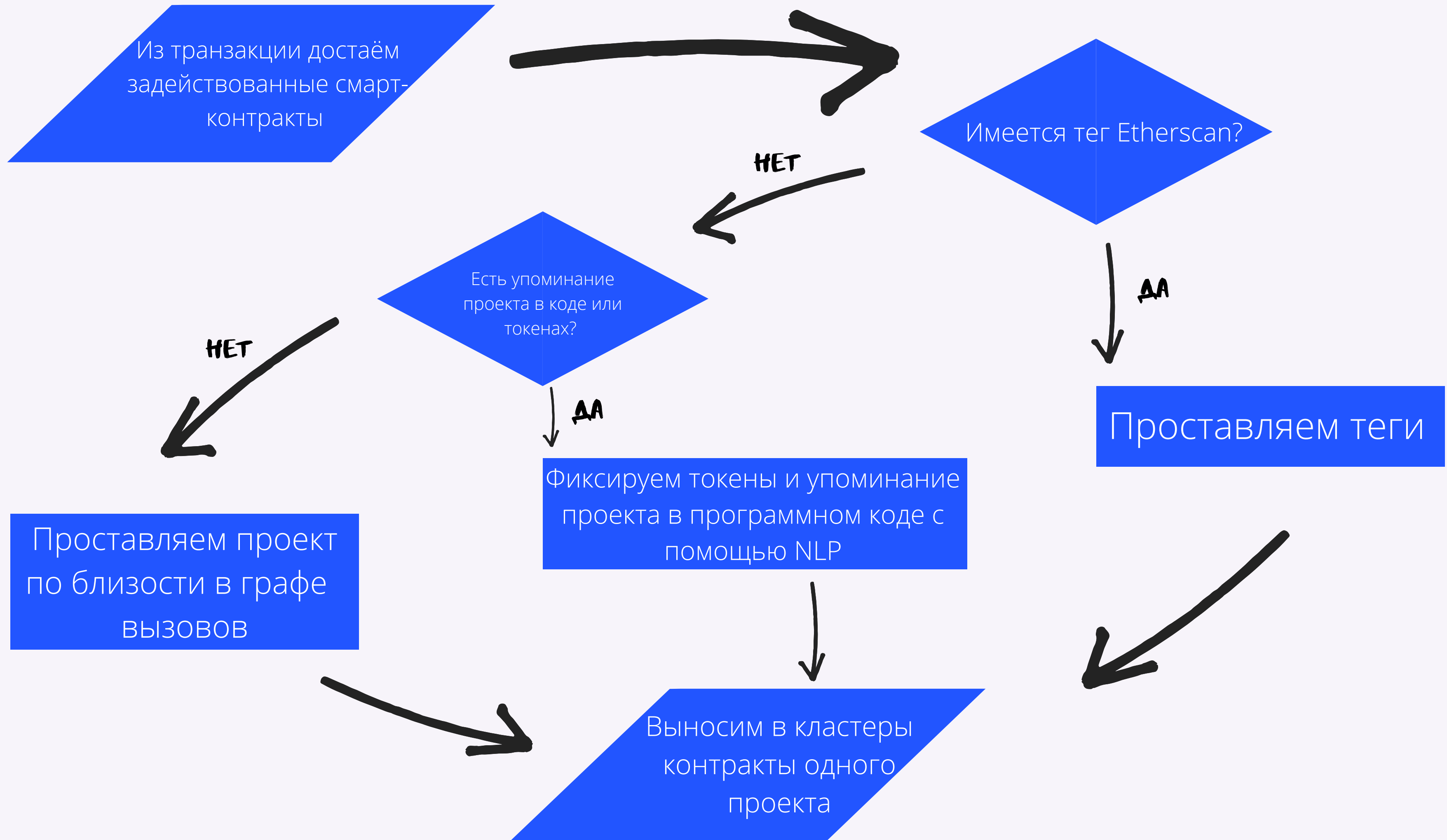
Мы сравниваем полученные нами данные на предмет их вхождения в список блокчейн-проектов.

В рамках решения кейса и в целях упрощения прототипирования мы взяли в качестве списков рейтинг ERC-20 и список разработчиков крипто-инструментов.

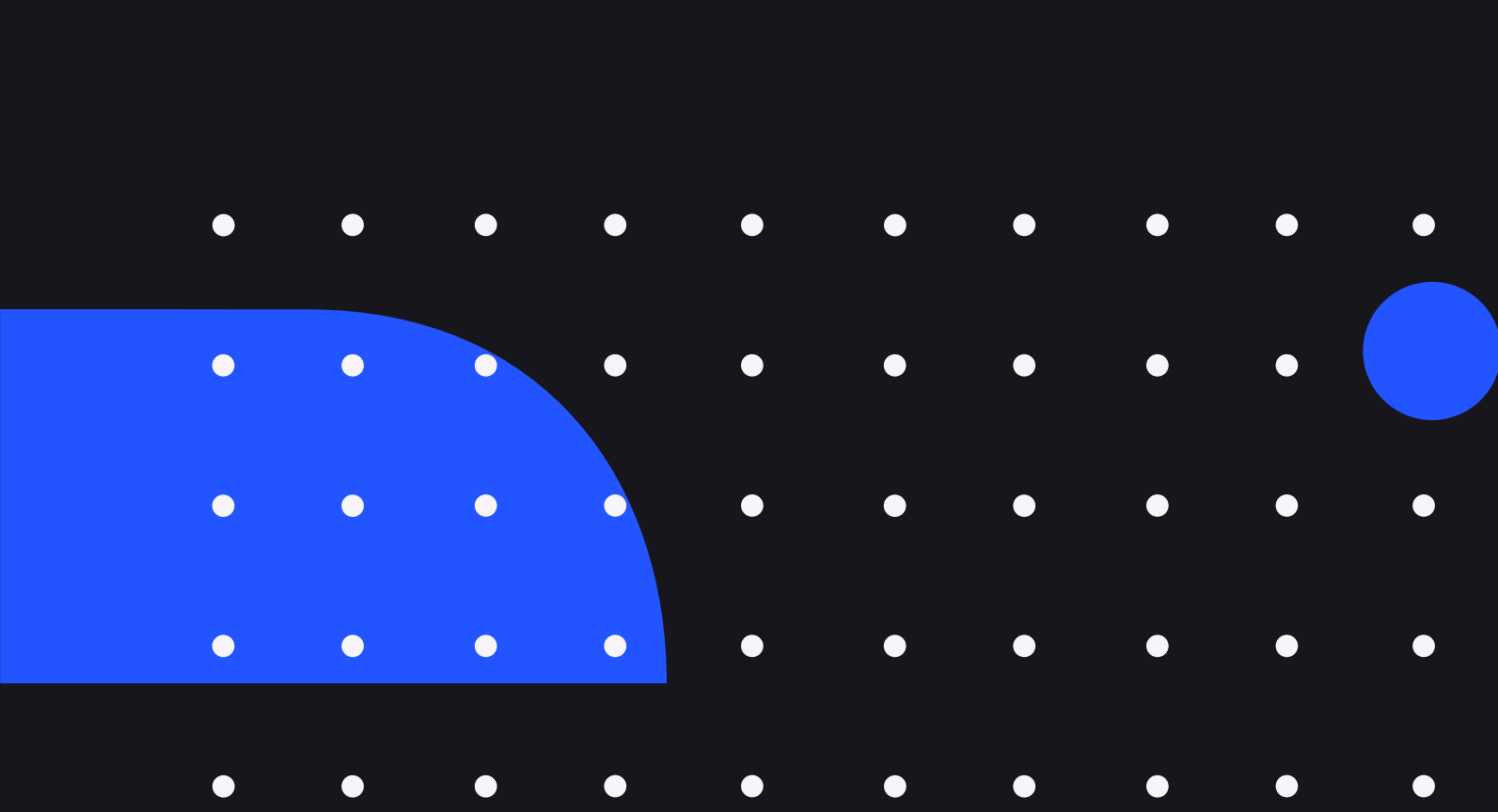
Так мы избавляемся от версионных символов и прочих побочных деталей и объединяем контракты в большой кластер.







Ещё одна вещь...



Смарт-контракты не обязательно кластеризовать по проектам

Статья [Assessing the Similarity of Smart Contracts by Clustering their Interfaces](#) натолкнула нас на множество идей

1

Статистический анализ

Оценка выполнения контрактов в каждой транзакции

2

Функционал смарт-контракта

Кластеризация по принципу действия, выделение стандартных интерфейсов

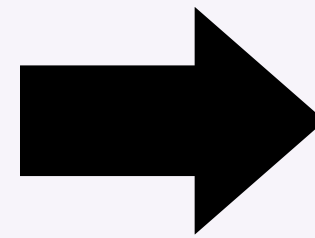
3

Граф вызовов между контрактами

Можно выделить адреса, играющие ключевую роль и строить кластеры вокруг них



Наш граф

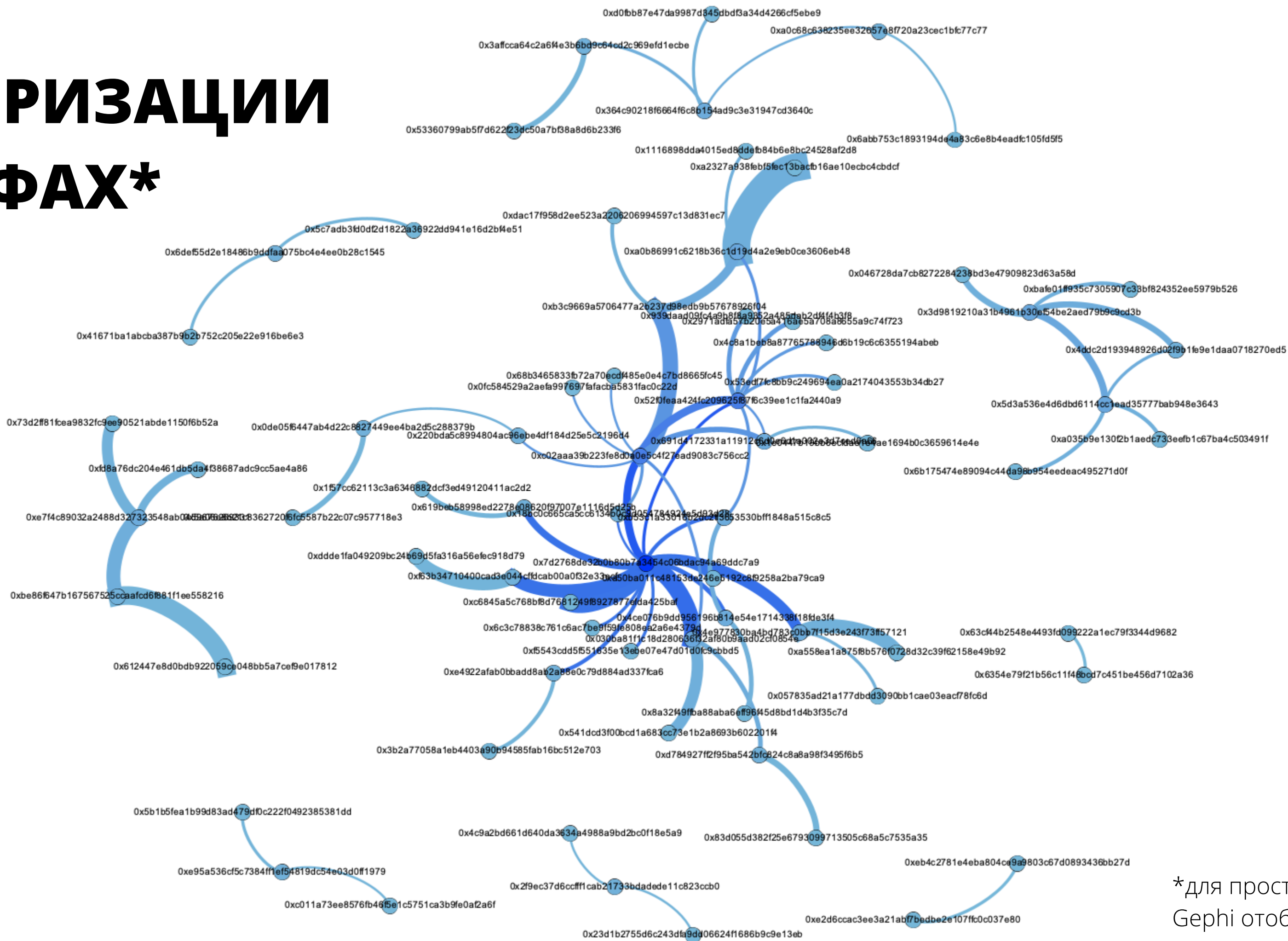


Попарно
считаем
количество
вызовов между
контрактами

Благодаря
networkx строим
взвешенный граф,
где вес рёбер -
число обращений
для каждой пары
контрактов

Контракты
взаимодействуют
больше других -
считаем их
причастными к
одному кластеру

МЕТОД КЛАСТЕРИЗАЦИИ НА ГРАФАХ*

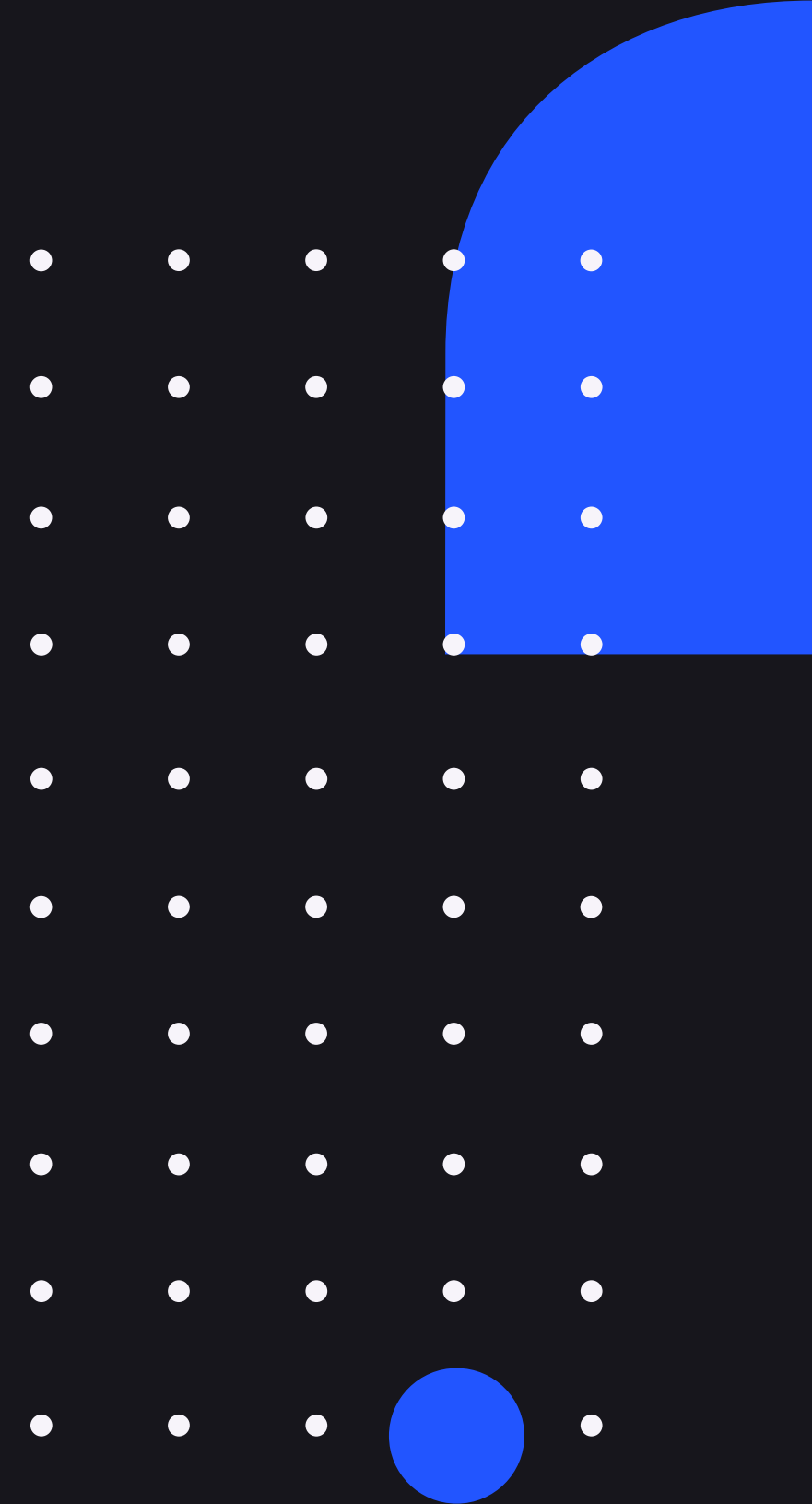


*для простоты визуализации в
Gephi отображали от 3 и более
вызовов

Исходный код и полученные данные доступны на GitHub

<https://github.com/dizpatcher/defianalytics>

Результаты были достигнуты благодаря API Bitquery, Etherscan, языку GraphQL, библиотекам networkx, requests и BeautifulSoup (Python) , а также Gephi.





ФОРМУЛА БУДУЩЕГО РАЗВИТИЯ

К описанному алгоритму

Подключаем ИИ

Учимся правильно

**Интерпретировать
собранные данные**

Проверяем на корректность

**Дополнительные
ВОЗМОЖНОСТИ**

ФОРМУЛА БУДУЩЕГО

Выродов Артём

t.me/vyrodov

Казьмин Олег

t.me/oleg_ator02

Маркович Владимир

t.me/vm5_ball

