Breaking out of the Ethereum maze

osintomático CONFERENCE

C1B3R1NT3LIG3NC1A UTPJ GICC 5 GUARDIA CIVIL

```
jose@drjekyll:~$ whoami
Jose
jose@drjekyll:~$ uname -r
1993-LOE-Plan Bolonia
jose@drjekyll:~$ lsb_release -d
Description:  Capitán Guardia Civil Rolling
jose@drjekyll:~$ pwd
JefaturaPolicíaJudicial/UnidadTécnicaPolicíaJudicial
jose@drjekyll:~$ ls
Grupo_de_Ciberinteligencia_Criminal.md
```

Leo, escribo y monto en bici. No siempre en ese orden, nunca a la vez.

[Disclaimer]

Dejo un repo, pero no soy un crack en Python.
Habrá cosas mal incluso cosas que no funcionen.

No me responsabilizo de pérdidas patrimoniales debidas al hype de la charla y compras impulsivas de Eth.

Son las 15:30, si alguien se duerme no me hago cargo de lesiones cervicales derivadas.

{ [Ethereum]

< *One Ring to rule them all, One Ring to find them, One Ring to bring them all and in the* ERC20 bind them >

ethereum

}

$$\sigma' = \Upsilon(\sigma, T)$$

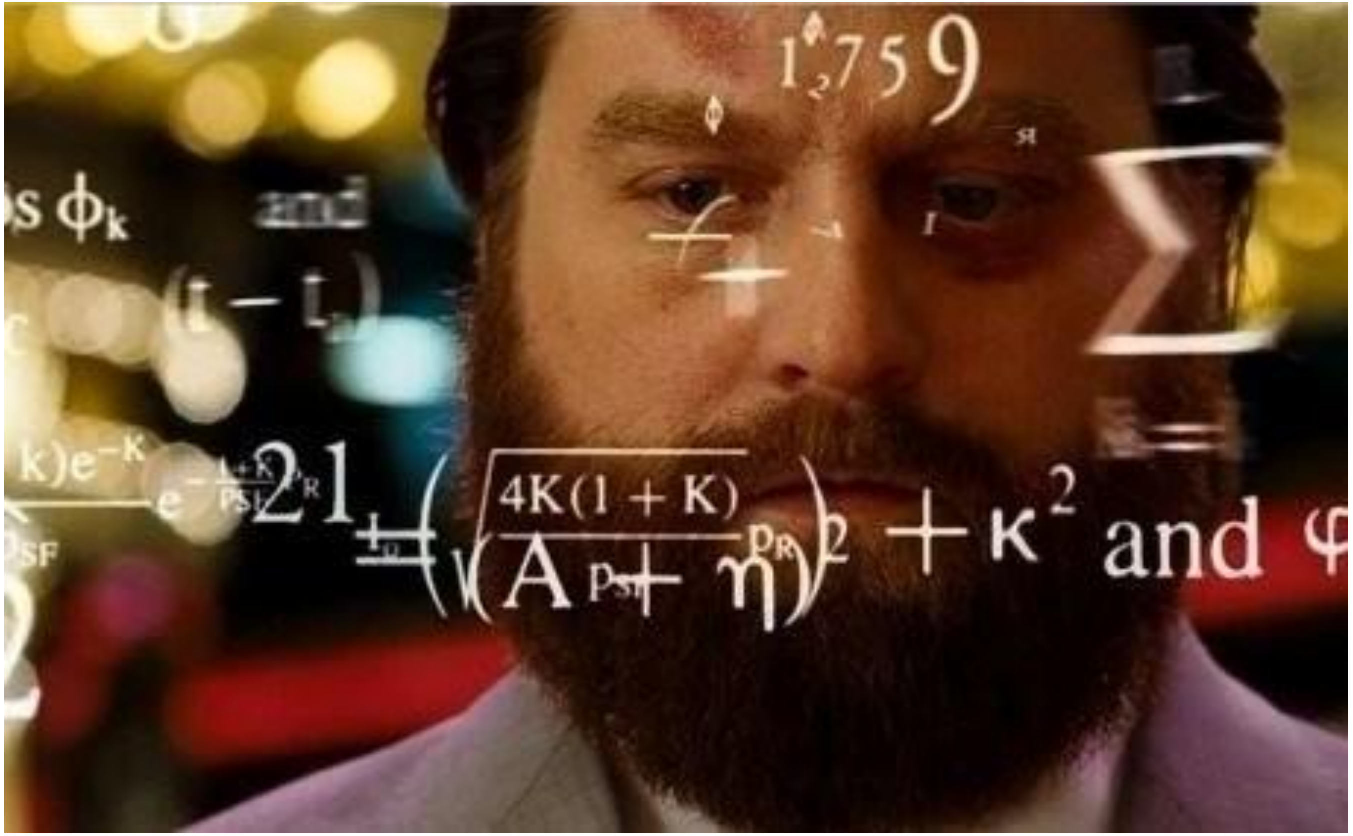
Ethereum, taken as a whole, can be viewed as a transaction-based state machine: we begin with a genesis state and incrementally execute transactions to morph it into some current state.

🤑 Arbitraje & Sandwich 🤑

⚡FlashLoans⚡

MEV
Valor máximo extraíble

🤖FrontRunners


1,759
s φ_k
and
k)e^-K
21
4K(1 + K)
+ κ² and φ

[Datos I] Comercio de datos

Bitquery

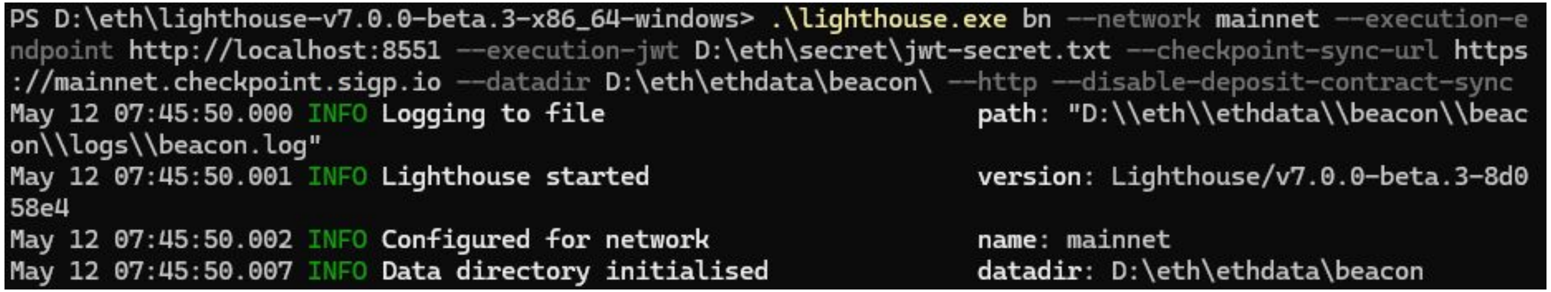
Dune

Etherscan

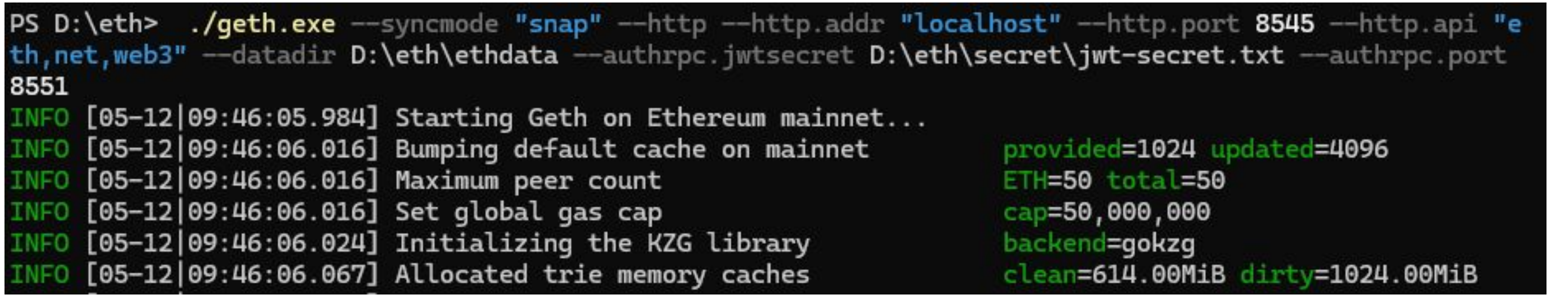
< Consultas limitadas, coste adicional, no 100% personalizables >

[Datos II] Do It Yourself

Geth + LightHouse



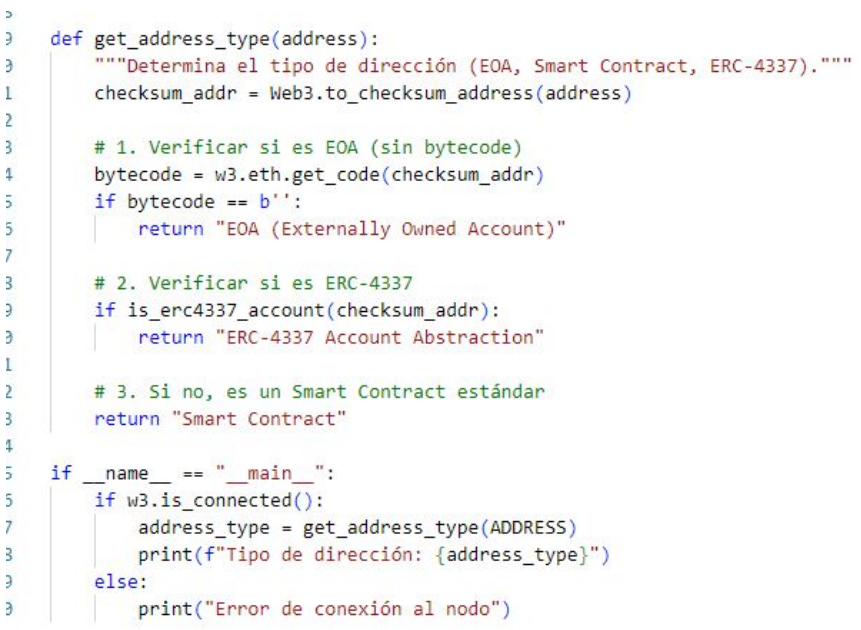
```
PS D:\eth\lighthouse-v7.0.0-beta.3-x86_64-windows> .\lighthouse.exe bn --network mainnet --execution-e
ndpoint http://localhost:8551 --execution-jwt D:\eth\secret\jwt-secret.txt --checkpoint-sync-url https
://mainnet.checkpoint.sigp.io --datadir D:\eth\ethdata\beacon\ --http --disable-deposit-contract-sync
May 12 07:45:50.000 INFO Logging to file                                path: "D:\\eth\\ethdata\\beacon\\beac
on\\logs\\beacon.log"
May 12 07:45:50.001 INFO Lighthouse started                             version: Lighthouse/v7.0.0-beta.3-8d0
58e4
May 12 07:45:50.002 INFO Configured for network                         name: mainnet
May 12 07:45:50.007 INFO Data directory initialised                     datadir: D:\eth\ethdata\beacon
```



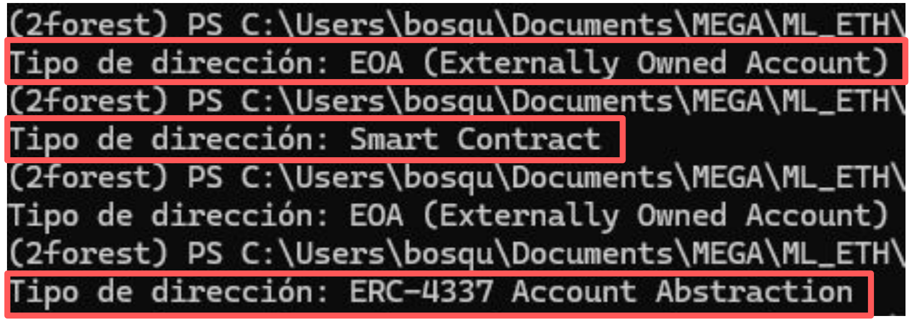
```
PS D:\eth>  ./geth.exe --syncmode "snap" --http --http.addr "localhost" --http.port 8545 --http.api "e
th,net,web3" --datadir D:\eth\ethdata --authrpc.jwtsecret D:\eth\secret\jwt-secret.txt --authrpc.port
8551
INFO [05-12|09:46:05.984] Starting Geth on Ethereum mainnet...
INFO [05-12|09:46:06.016] Bumping default cache on mainnet          provided=1024 updated=4096
INFO [05-12|09:46:06.016] Maximum peer count                        ETH=50 total=50
INFO [05-12|09:46:06.016] Set global gas cap                        cap=50,000,000
INFO [05-12|09:46:06.024] Initializing the KZG library              backend=gokzg
INFO [05-12|09:46:06.067] Allocated trie memory caches              clean=614.00MiB dirty=1024.00MiB
```

Ejecución + Consenso

[Level 0] Tipo



```python
def get_address_type(address):
    """Determina el tipo de dirección (EOA, Smart Contract, ERC-4337)."""
    checksum_addr = Web3.to_checksum_address(address)

    # 1. Verificar si es EOA (sin bytecode)
    bytecode = w3.eth.get_code(checksum_addr)
    if bytecode == b'':
        return "EOA (Externally Owned Account)"

    # 2. Verificar si es ERC-4337
    if is_erc4337_account(checksum_addr):
        return "ERC-4337 Account Abstraction"

    # 3. Si no, es un Smart Contract estándar
    return "Smart Contract"

if __name__ == "__main__":
    if w3.is_connected():
        address_type = get_address_type(ADDRESS)
        print(f"Tipo de dirección: {address_type}")
    else:
        print("Error de conexión al nodo")
```



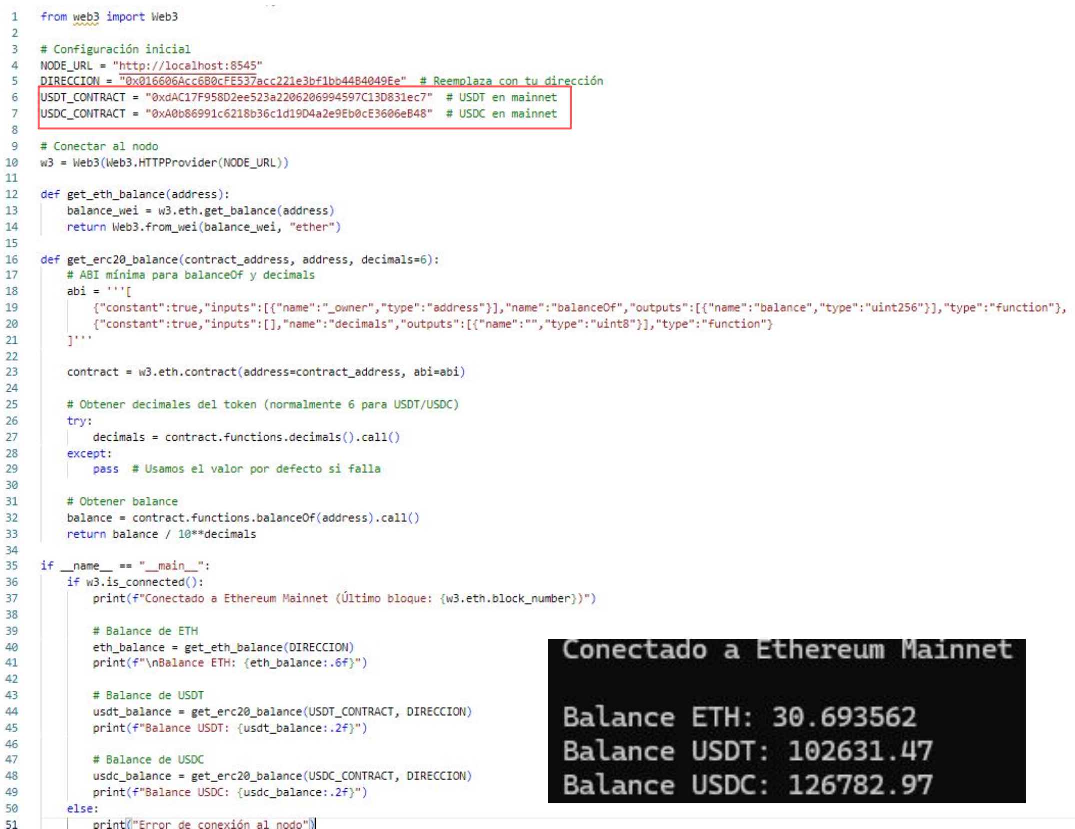
```
(2forest) PS C:\Users\bosqu\Documents\MEGA\ML_ETH\
Tipo de dirección: EOA (Externally Owned Account)
(2forest) PS C:\Users\bosqu\Documents\MEGA\ML_ETH\
Tipo de dirección: Smart Contract
(2forest) PS C:\Users\bosqu\Documents\MEGA\ML_ETH\
Tipo de dirección: EOA (Externally Owned Account)
(2forest) PS C:\Users\bosqu\Documents\MEGA\ML_ETH\
Tipo de dirección: ERC-4337 Account Abstraction
```

[Level I] Balance

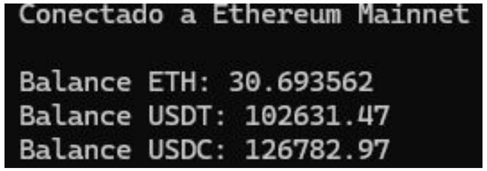
Solo ETH

```python
import requests
from decimal import Decimal

# Atacando al nodo
NODE_URL = "http://localhost:8545"
DIRECCION = "0x...."  # Dirección a consultar

# JSON-RPC
payload = {
    "jsonrpc": "2.0",
    "method": "eth_getBalance",
    "params": [DIRECCION, "latest"],
    "id": 1
}

response = requests.post(NODE_URL, json=payload).json()

# El balance está en wei
if "result" in response:
    balance_wei = int(response["result"], 16)
    balance_eth = balance_wei / 10**18  # 1 ETH = 10^18 wei
    print(f"Balance: {balance_eth:.6f} ETH")
else:
    print("Error:", response.get("error", "Desconocido"))
```

ETH, USDT, USDC

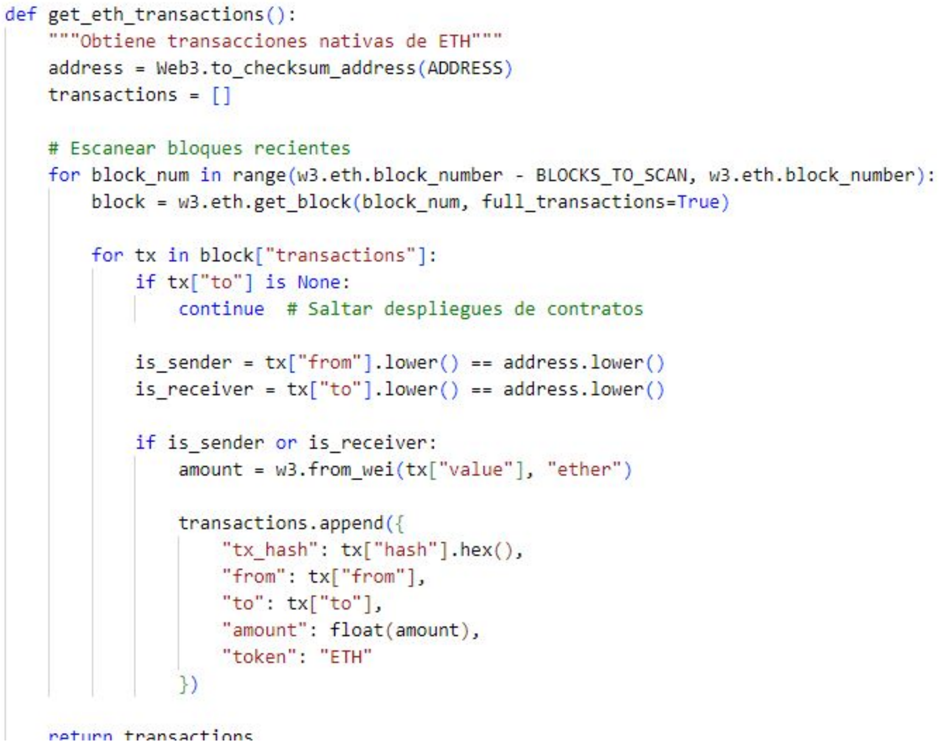


```python
from web3 import Web3

# Configuración inicial
NODE_URL = "http://localhost:8545"
DIRECCION = "0x016606Acc6B0cFE537acc221e3bf1bb44B4049Ee"  # Reemplaza con tu dirección
USDT_CONTRACT = "0xdAC17F958D2ee523a2206206994597C13D831ec7"  # USDT en mainnet
USDC_CONTRACT = "0xA0b86991c6218b36c1d19D4a2e9Eb0cE3606eB48"  # USDC en mainnet

# Conectar al nodo
w3 = Web3(Web3.HTTPProvider(NODE_URL))

def get_eth_balance(address):
    balance_wei = w3.eth.get_balance(address)
    return Web3.from_wei(balance_wei, "ether")

def get_erc20_balance(contract_address, address, decimals=6):
    # ABI mínima para balanceOf y decimals
    abi = '''[
        {"constant":true,"inputs":[{"name":"_owner","type":"address"}],"name":"balanceOf","outputs":[{"name":"balance","type":"uint256"}],"type":"function"},
        {"constant":true,"inputs":[],"name":"decimals","outputs":[{"name":"","type":"uint8"}],"type":"function"}
    ]'''

    contract = w3.eth.contract(address=contract_address, abi=abi)

    # Obtener decimales del token (normalmente 6 para USDT/USDC)
    try:
        decimals = contract.functions.decimals().call()
    except:
        pass  # Usamos el valor por defecto si falla

    # Obtener balance
    balance = contract.functions.balanceOf(address).call()
    return balance / 10**decimals

if __name__ == "__main__":
    if w3.is_connected():
        print(f"Conectado a Ethereum Mainnet (Último bloque: {w3.eth.block_number})")

        # Balance de ETH
        eth_balance = get_eth_balance(DIRECCION)
        print(f"\nBalance ETH: {eth_balance:.6f}")

        # Balance de USDT
        usdt_balance = get_erc20_balance(USDT_CONTRACT, DIRECCION)
        print(f"Balance USDT: {usdt_balance:.2f}")

        # Balance de USDC
        usdc_balance = get_erc20_balance(USDC_CONTRACT, DIRECCION)
        print(f"Balance USDC: {usdc_balance:.2f}")
    else:
        print("Error de conexión al nodo")
```

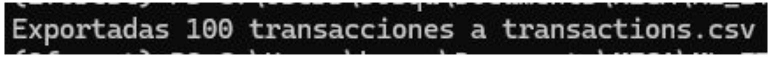


```
Conectado a Ethereum Mainnet

Balance ETH: 30.693562
Balance USDT: 102631.47
Balance USDC: 126782.97
```

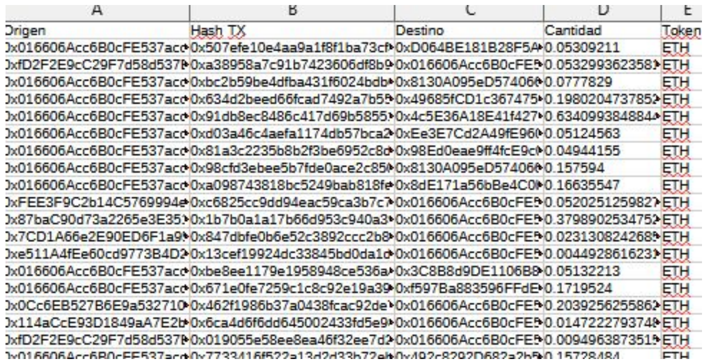
[Level II] Transacciones



```python
def get_eth_transactions():
    """Obtiene transacciones nativas de ETH"""
    address = Web3.to_checksum_address(ADDRESS)
    transactions = []

    # Escanear bloques recientes
    for block_num in range(w3.eth.block_number - BLOCKS_TO_SCAN, w3.eth.block_number):
        block = w3.eth.get_block(block_num, full_transactions=True)

        for tx in block["transactions"]:
            if tx["to"] is None:
                continue  # Saltar despliegues de contratos

            is_sender = tx["from"].lower() == address.lower()
            is_receiver = tx["to"].lower() == address.lower()

            if is_sender or is_receiver:
                amount = w3.from_wei(tx["value"], "ether")

                transactions.append({
                    "tx_hash": tx["hash"].hex(),
                    "from": tx["from"],
                    "to": tx["to"],
                    "amount": float(amount),
                    "token": "ETH"
                })

    return transactions
```

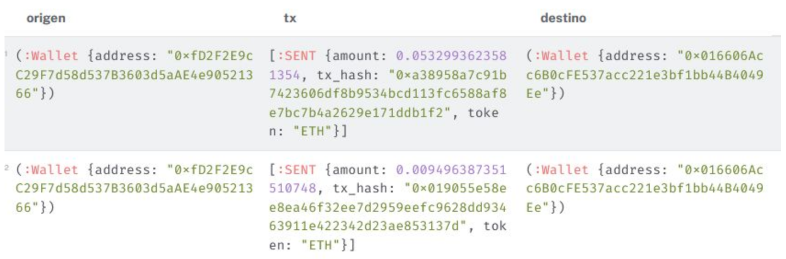


```
Exportadas 100 transacciones a transactions.csv
```

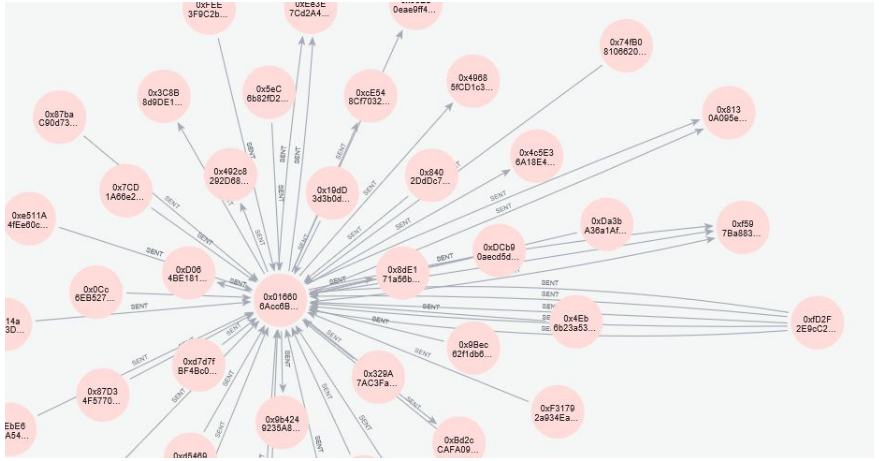


A	B	C	D	E
Origen	Hash TX	Destino	Cantidad	Token
0x016606Acc6B0cFE537acc	0x507efe10e4aa9a1f8f1ba73cf	0xD064BE181B28F5A	0.05309211	ETH
0xfD2F2E9cC29F7d58d537f	0xa38958a7c91b7423606df8b9	0x016606Acc6B0cFE5	0.053299362358	ETH
0x016606Acc6B0cFE537acc	0xbc2b59be4dfba431f6024bdb	0x8130A095eD574060	0.0777829	ETH
0x016606Acc6B0cFE537acc	0x634d2beed66fcad7492a7b55	0x49685fCD1c367475	0.198020473785	ETH
0x016606Acc6B0cFE537acc	0x91db8ec8486c417d69b5855	0x4c5E36A18E41f427	0.634099384884	ETH
0x016606Acc6B0cFE537acc	0xd03a46c4aefa1174db57bca2	0xEe3E7Cd2A49fE960	0.05124563	ETH
0x016606Acc6B0cFE537acc	0x81a3c2235b8b2f3be6952c8d	0x98Ed0eae9ff4fcE9c	0.04944155	ETH
0x016606Acc6B0cFE537acc	0x98cfd3ebee5b7fde0ace2c850	0x8130A095eD574060	0.157594	ETH
0x016606Acc6B0cFE537acc	0xa098743818bc5249bab818fe	0x8dE171a56bBe4C0	0.16635547	ETH
0xFEE3F9C2b14C5769994e	0xc6825cc9dd94eac59ca3b7c7	0x016606Acc6B0cFE5	0.052025125982	ETH
0x87baC90d73a2265e3E35	0x1b7b0a1a17b66d953c940a3	0x016606Acc6B0cFE5	0.379890253475	ETH
0x7CD1A66e2E90ED6F1a9	0x847dbfe0b6e52c3892ccc2b8	0x016606Acc6B0cFE5	0.023130824268	ETH
0xe511A4fEe60cd9773B4D2	0x13cef19924dc33845bd0da1d	0x016606Acc6B0cFE5	0.004492861623	ETH
0x016606Acc6B0cFE537acc	0xbe8ee1179e1958948ce536a	0x3C8B8d9DE1106B8	0.05132213	ETH
0x016606Acc6B0cFE537acc	0x671e0fe7259c1c8c92e19a39	0xf597Ba883596FFdE	0.1719524	ETH
0x0Cc6EB527B6E9a532710	0x462f1986b37a0438fcac92de	0x016606Acc6B0cFE5	0.203925625586	ETH
0x114aCcE93D1849aA7E2b	0x6ca4d6f6dd645002433fd5e9	0x016606Acc6B0cFE5	0.014722279374	ETH
0xfD2F2E9cC29F7d58d537f	0x019055e58ee8ea46f32ee7d2	0x016606Acc6B0cFE5	0.009496387351	ETH
0x016606Acc6B0cFE537acc	0x773416f522a13d2d33b72eb	0x492c8292D682a2b5	0.15728484	ETH

[Level III] Graph database



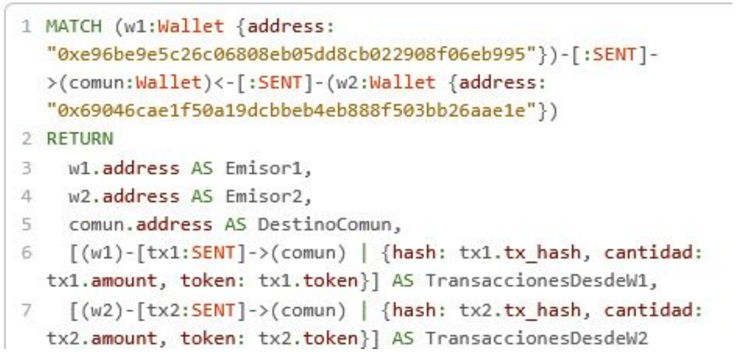
	origen	tx	destino
1	(:Wallet {address: "0×fD2F2E9cC29F7d58d537B3603d5aAE4e90521366"})	[:SENT {amount: 0.0532993623581354, tx_hash: "0×a38958a7c91b7423606df8b9534bcd113fc6588af8e7bc7b4a2629e171ddb1f2", token: "ETH"}]	(:Wallet {address: "0×016606Acc6B0cFE537acc221e3bf1bb44B4049Ee"})
2	(:Wallet {address: "0×fD2F2E9cC29F7d58d537B3603d5aAE4e90521366"})	[:SENT {amount: 0.009496387351510748, tx_hash: "0×019055e58ee8ea46f32ee7d2959eefc9628dd93463911e422342d23ae853137d", token: "ETH"}]	(:Wallet {address: "0×016606Acc6B0cFE537acc221e3bf1bb44B4049Ee"})



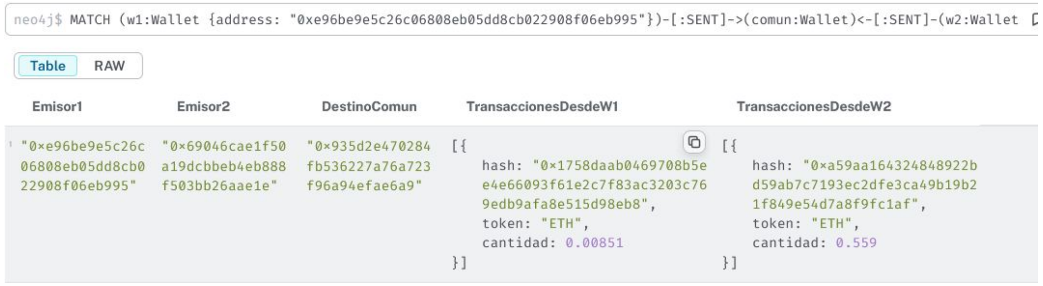
[Level Padawan]

[Level Padawan]

Dadas dos o más address, ¿qué destinatarios comunes encontramos?



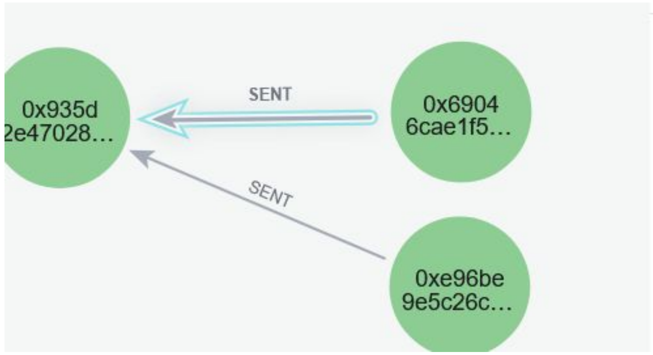
```
MATCH (w1:Wallet {address:
"0xe96be9e5c26c06808eb05dd8cb022908f06eb995"})-[:SENT]-
>(comun:Wallet)<-[:SENT]-(w2:Wallet {address:
"0x69046cae1f50a19dcbbeb4eb888f503bb26aae1e"})
RETURN
  w1.address AS Emisor1,
  w2.address AS Emisor2,
  comun.address AS DestinoComun,
  [(w1)-[tx1:SENT]->(comun) | {hash: tx1.tx_hash, cantidad:
tx1.amount, token: tx1.token}] AS TransaccionesDesdeW1,
  [(w2)-[tx2:SENT]->(comun) | {hash: tx2.tx_hash, cantidad:
tx2.amount, token: tx2.token}] AS TransaccionesDesdeW2
```

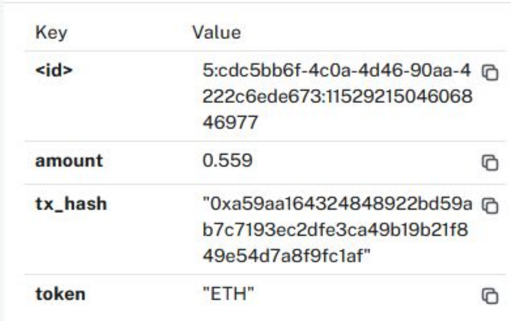


```
neo4j$ MATCH (w1:Wallet {address: "0xe96be9e5c26c06808eb05dd8cb022908f06eb995"})-[:SENT]->(comun:Wallet)<-[:SENT]-(w2:Wallet
```

Table RAW

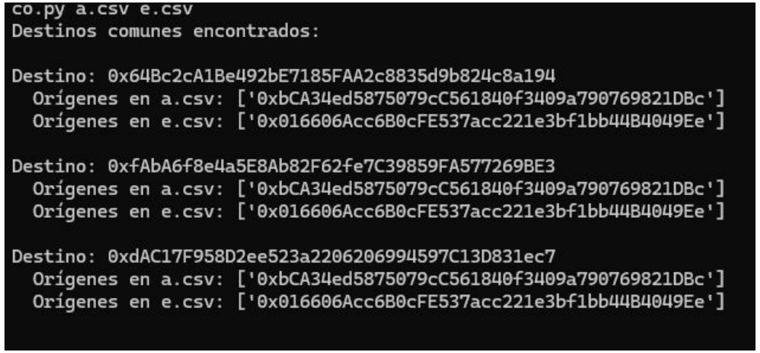
Emisor1	Emisor2	DestinoComun	TransaccionesDesdeW1	TransaccionesDesdeW2
"0xe96be9e5c26c06808eb05dd8cb022908f06eb995"	"0x69046cae1f50a19dcbbeb4eb888f503bb26aae1e"	"0x935d2e470284fb536227a76a723f96a94efae6a9"	[{ hash: "0x1758daab0469708b5ee4e66093f61e2c7f83ac3203c769edb9afa8e515d98eb8", token: "ETH", cantidad: 0.00851 }]	[{ hash: "0xa59aa164324848922bd59ab7c7193ec2dfe3ca49b19b21f849e54d7a8f9fc1af", token: "ETH", cantidad: 0.559 }]





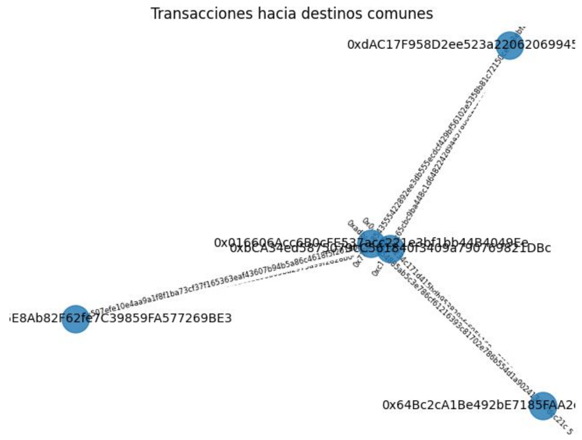
Key	Value
<id>	5:cdc5bb6f-4c0a-4d46-90aa-4222c6ede673:1152921504606846977
amount	0.559
tx_hash	"0xa59aa164324848922bd59ab7c7193ec2dfe3ca49b19b21f849e54d7a8f9fc1af"
token	"ETH"

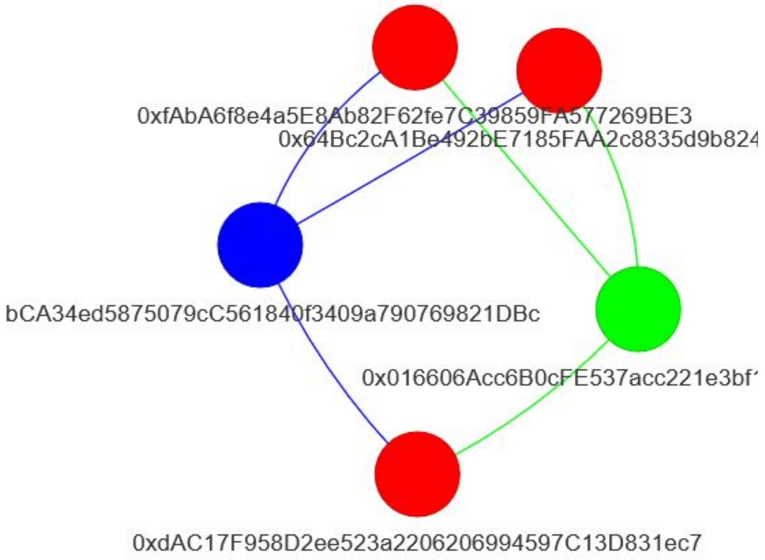
[Level Padawan] Python



```
co.py a.csv e.csv
Destinos comunes encontrados:

Destino: 0x64Bc2cA1Be492bE7185FAA2c8835d9b824c8a194
  Orígenes en a.csv: ['0xbCA34ed5875079cC561840f3409a790769821DBc']
  Orígenes en e.csv: ['0x016606Acc6B0cFE537acc221e3bf1bb44B4049Ee']

Destino: 0xfAbA6f8e4a5E8Ab82F62fe7C39859FA577269BE3
  Orígenes en a.csv: ['0xbCA34ed5875079cC561840f3409a790769821DBc']
  Orígenes en e.csv: ['0x016606Acc6B0cFE537acc221e3bf1bb44B4049Ee']

Destino: 0xdAC17F958D2ee523a2206206994597C13D831ec7
  Orígenes en a.csv: ['0xbCA34ed5875079cC561840f3409a790769821DBc']
  Orígenes en e.csv: ['0x016606Acc6B0cFE537acc221e3bf1bb44B4049Ee']
```

pyvis + pandas en html

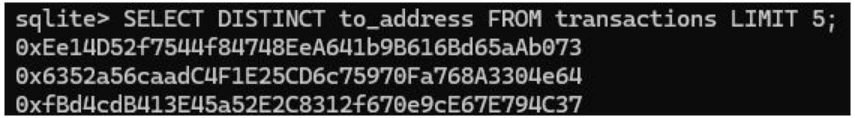

Transacciones hacia destinos comunes
0xdAC17F958D2ee523a2206206994
0x016606Acc6B0cFE537acc221e3bf1bb44B4049Ee
0xbCA34ed5875079cC561840f3409a790769821DBc
E8Ab82F62fe7C39859FA577269BE3
0x64Bc2cA1Be492bE7185FAA2


0xfAbA6f8e4a5E8Ab82F62fe7C39859FA577269BE3
0x64Bc2cA1Be492bE7185FAA2c8835d9b824
bCA34ed5875079cC561840f3409a790769821DBc
0x016606Acc6B0cFE537acc221e3bf
0xdAC17F958D2ee523a2206206994597C13D831ec7

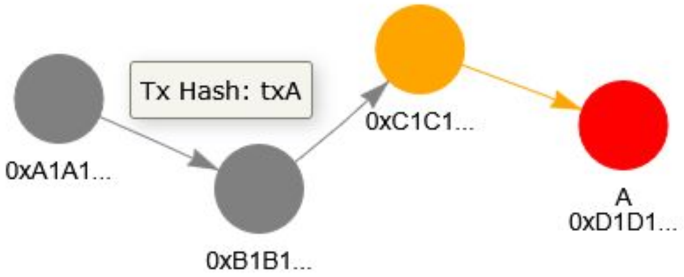
[Level Jedi]

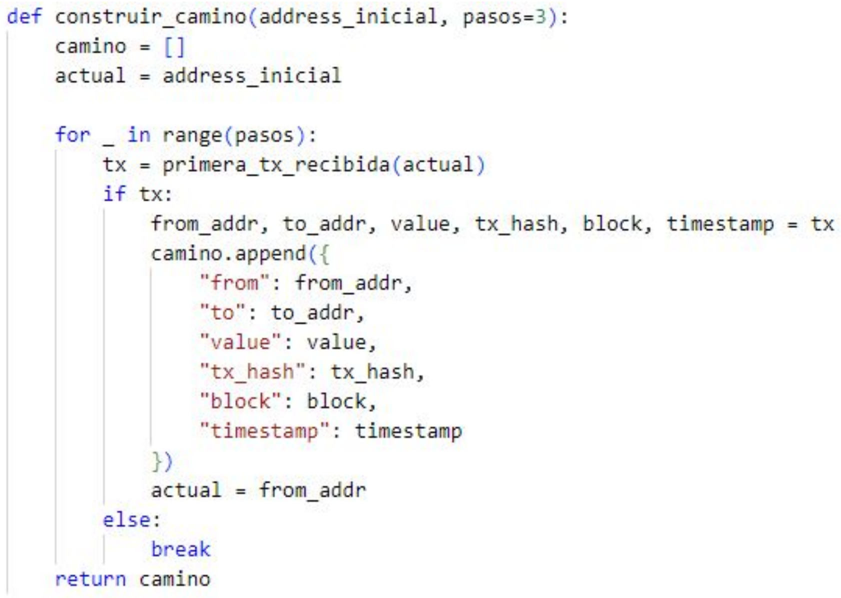
[Level Jedi] Track tx

Full node OR Index *(Index is for tiesos)*



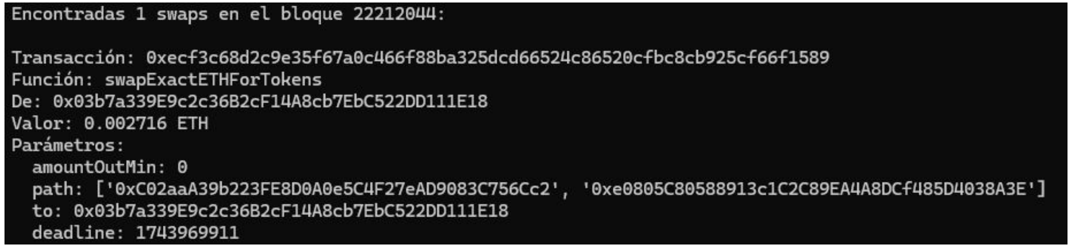
```
sqlite> SELECT DISTINCT to_address FROM transactions LIMIT 5;
0xEe14D52f7544f84748EeA641b9B616Bd65aAb073
0x6352a56caadC4F1E25CD6c75970Fa768A3304e64
0xfBd4cdB413E45a52E2C8312f670e9cE67E794C37
```



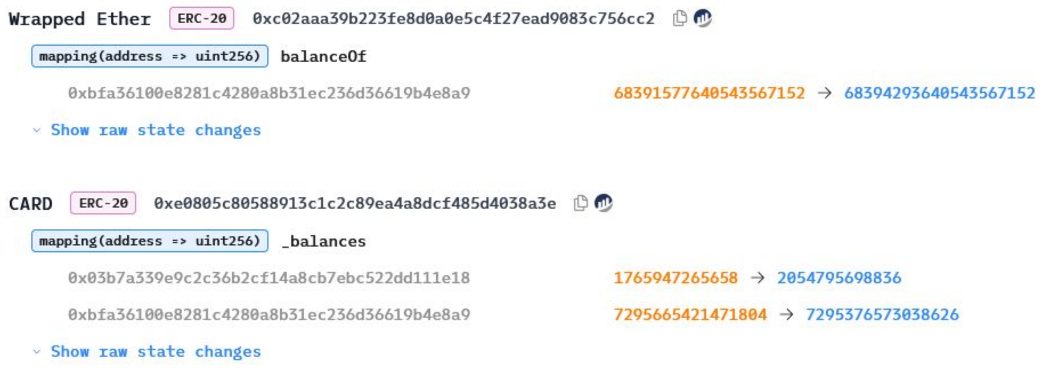


```python
def construir_camino(address_inicial, pasos=3):
    camino = []
    actual = address_inicial

    for _ in range(pasos):
        tx = primera_tx_recibida(actual)
        if tx:
            from_addr, to_addr, value, tx_hash, block, timestamp = tx
            camino.append({
                "from": from_addr,
                "to": to_addr,
                "value": value,
                "tx_hash": tx_hash,
                "block": block,
                "timestamp": timestamp
            })
            actual = from_addr
        else:
            break
    return camino
```

[Level Jedi] Caza Unicornios



```
Encontradas 1 swaps en el bloque 22212044:

Transacción: 0xecf3c68d2c9e35f67a0c466f88ba325dcd66524c86520cfbc8cb925cf66f1589
Función: swapExactETHForTokens
De: 0x03b7a339E9c2c36B2cF14A8cb7EbC522DD111E18
Valor: 0.002716 ETH
Parámetros:
  amountOutMin: 0
  path: ['0xC02aaA39b223FE8D0A0e5C4F27eAD9083C756Cc2', '0xe0805C80588913c1C2C89EA4A8DCf485D4038A3E']
  to: 0x03b7a339E9c2c36B2cF14A8cb7EbC522DD111E18
  deadline: 1743969911
```



Wrapped Ether ERC-20 0xc02aaa39b223fe8d0a0e5c4f27ead9083c756cc2

mapping(address => uint256) balanceOf

0xbfa36100e8281c4280a8b31ec236d36619b4e8a9 68391577640543567152 → 68394293640543567152

Show raw state changes

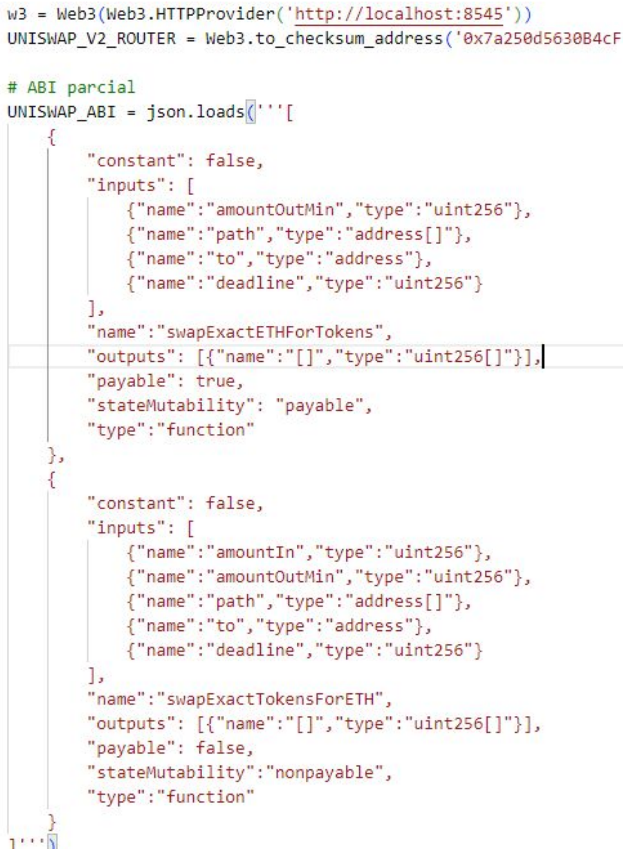
CARD ERC-20 0xe0805c80588913c1c2c89ea4a8dcf485d4038a3e

mapping(address => uint256) _balances

0x03b7a339e9c2c36b2cf14a8cb7ebc522dd111e18 1765947265658 → 2054795698836

0xbfa36100e8281c4280a8b31ec236d36619b4e8a9 7295665421471804 → 7295376573038626

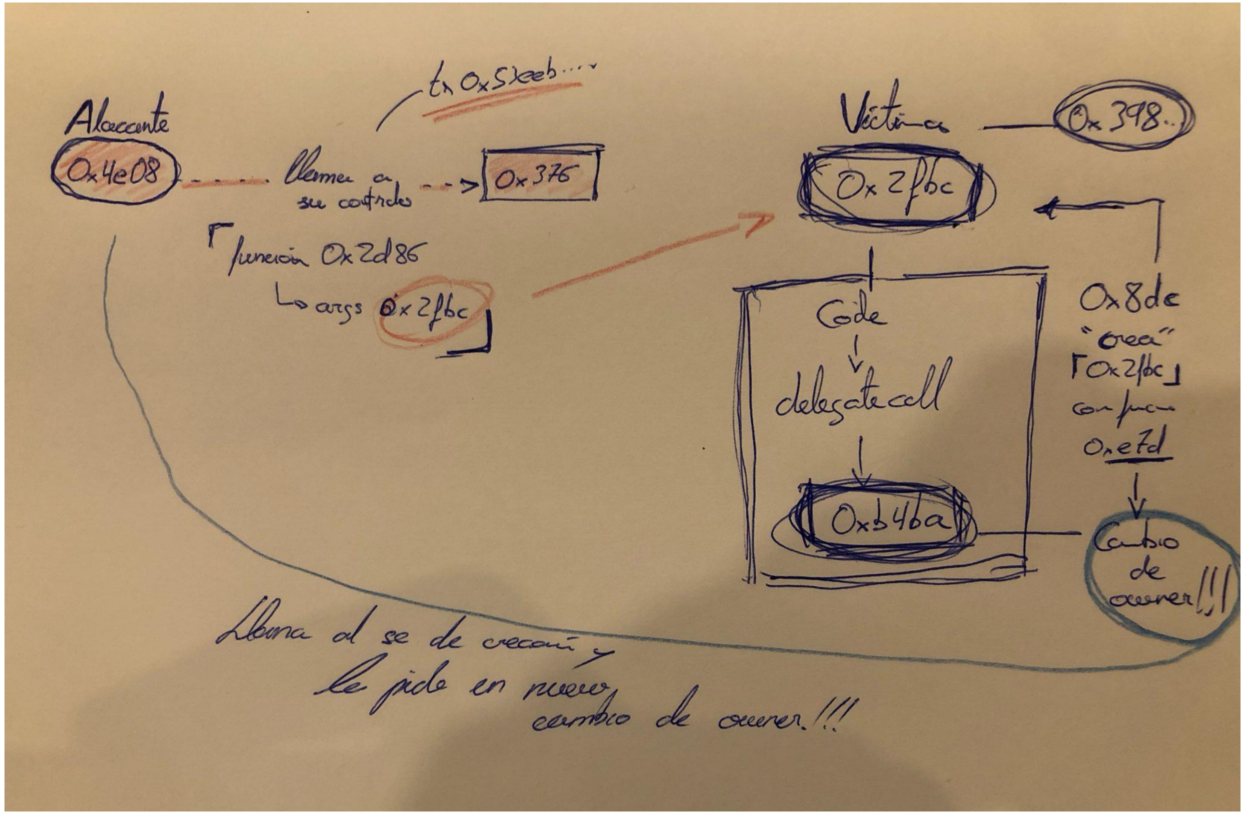
Show raw state changes



```
w3 = Web3(Web3.HTTPProvider('http://localhost:8545'))
UNISWAP_V2_ROUTER = Web3.to_checksum_address('0x7a250d5630B4cF

# ABI parcial
UNISWAP_ABI = json.loads('''[
    {
        "constant": false,
        "inputs": [
            {"name":"amountOutMin","type":"uint256"},
            {"name":"path","type":"address[]"},
            {"name":"to","type":"address"},
            {"name":"deadline","type":"uint256"}
        ],
        "name":"swapExactETHForTokens",
        "outputs": [{"name":"[]","type":"uint256[]"}],
        "payable": true,
        "stateMutability": "payable",
        "type":"function"
    },
    {
        "constant": false,
        "inputs": [
            {"name":"amountIn","type":"uint256"},
            {"name":"amountOutMin","type":"uint256"},
            {"name":"path","type":"address[]"},
            {"name":"to","type":"address"},
            {"name":"deadline","type":"uint256"}
        ],
        "name":"swapExactTokensForETH",
        "outputs": [{"name":"[]","type":"uint256[]"}],
        "payable": false,
        "stateMutability":"nonpayable",
        "type":"function"
    }
]''')
```

[Lord Sith]

[Lord Sith] Papel + Bic



[Más…]

Análisis de tx complejas p.e smart contracts + membots

Predicción de comportamientos (ML)

Varios nodos, varias redes → Bridges

Y todo lo que se pueda imaginar… TODO ESTÁ AHÍ, SOLO HAY QUE INTERPRETARLO

[Thanks!]

GitHub → **drj3ky11/AliceInWeb3land**

LinkedIn https://www.linkedin.com/in/josete/