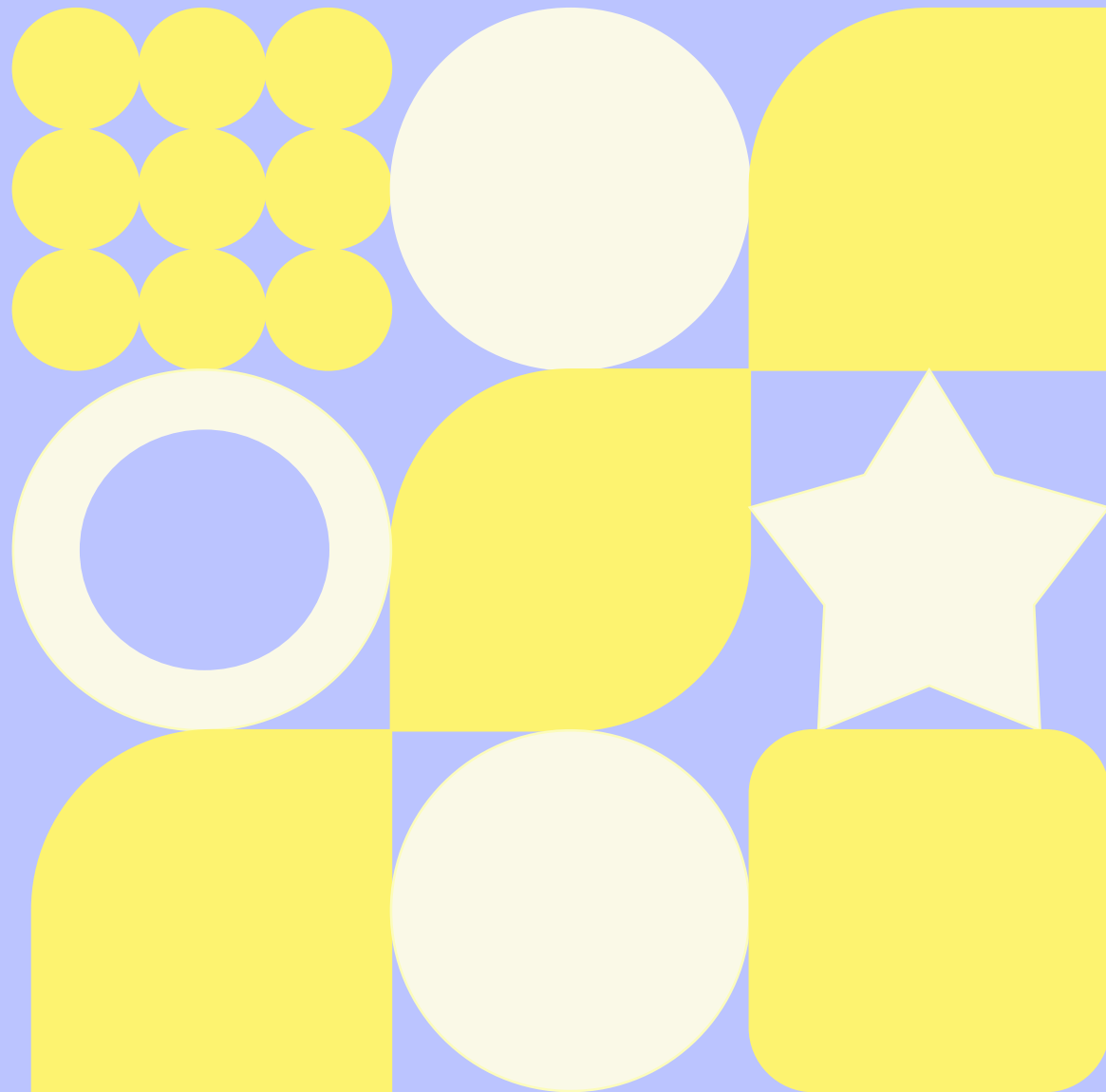
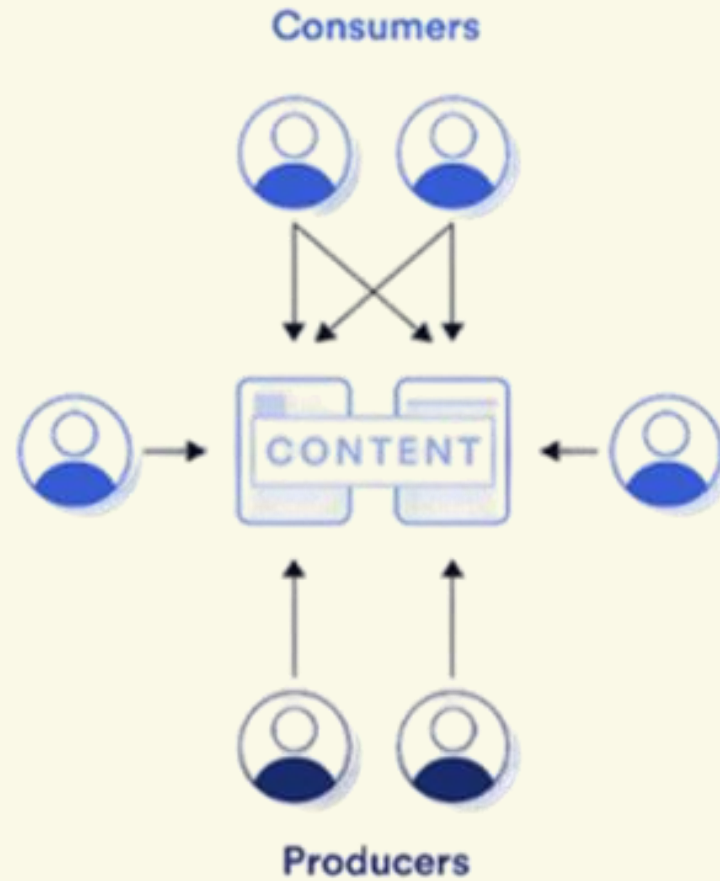


# Web3 для самых маленьких

Преобразование интернета с помощью  
децентрализации



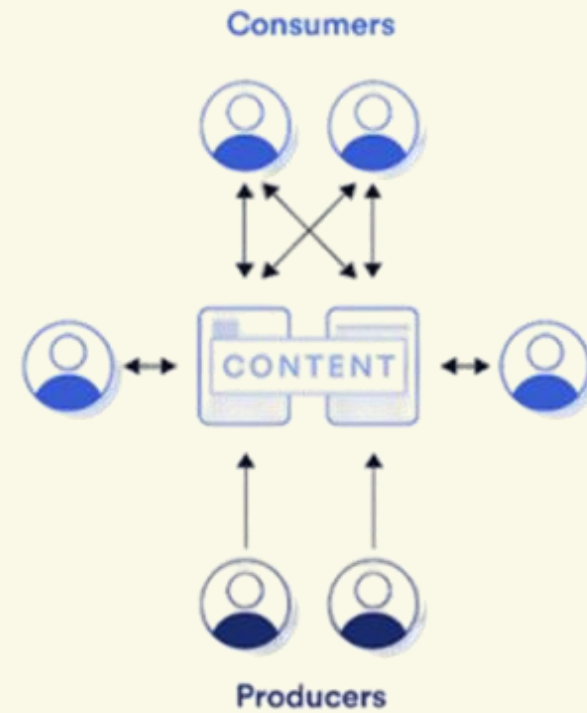
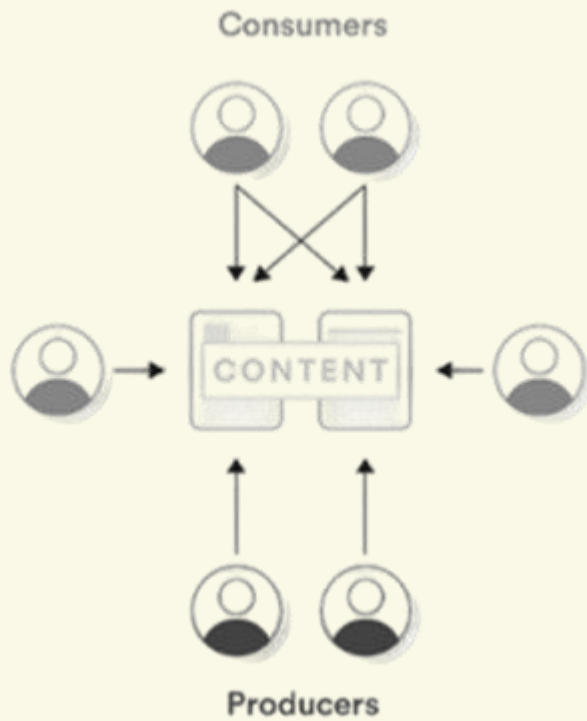
# Web1



# Web1

vs

# Web2

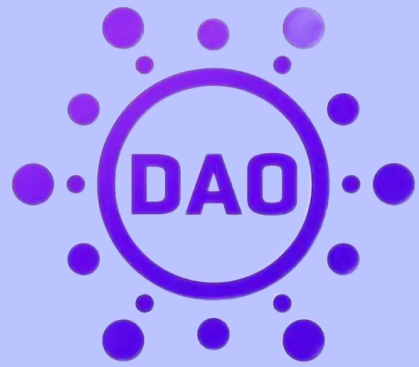


# Web2

# Web3

vs





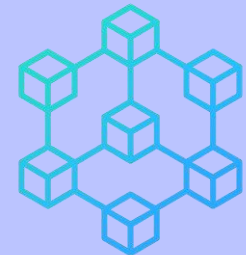
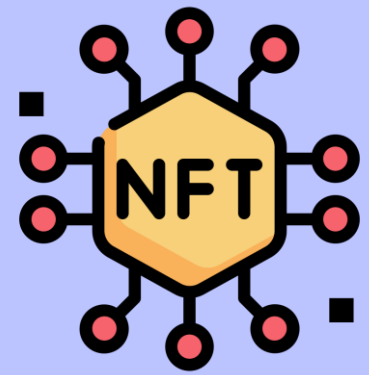
**SELF-  
SOVEREIGN  
IDENTITY**



**arweave**



**BitTorrent**  
File System



**BLOCKCHAIN**



**dCompute**

# Как создать безопасный и свободный обмен цифровыми активами напрямую между участниками?

## Проблема :

Покупатель и продавец не доверяют друг другу. Высокие комиссии и задержки — из-за посредников.

## Решение :

Создание приложения на базе блокчейн технологий

# Требования к приложению

- ◆ Доверие без посредника
- ◆ Отсутствие единой точки отказа
- ◆ Полная прозрачность операций
- ◆ Неизменяемые записи
- ◆ Открытый исходный код
- ◆ Самостоятельное хранение активов
- ◆ Отсутствие блокировок



# Инструменты

**Блокчейн**

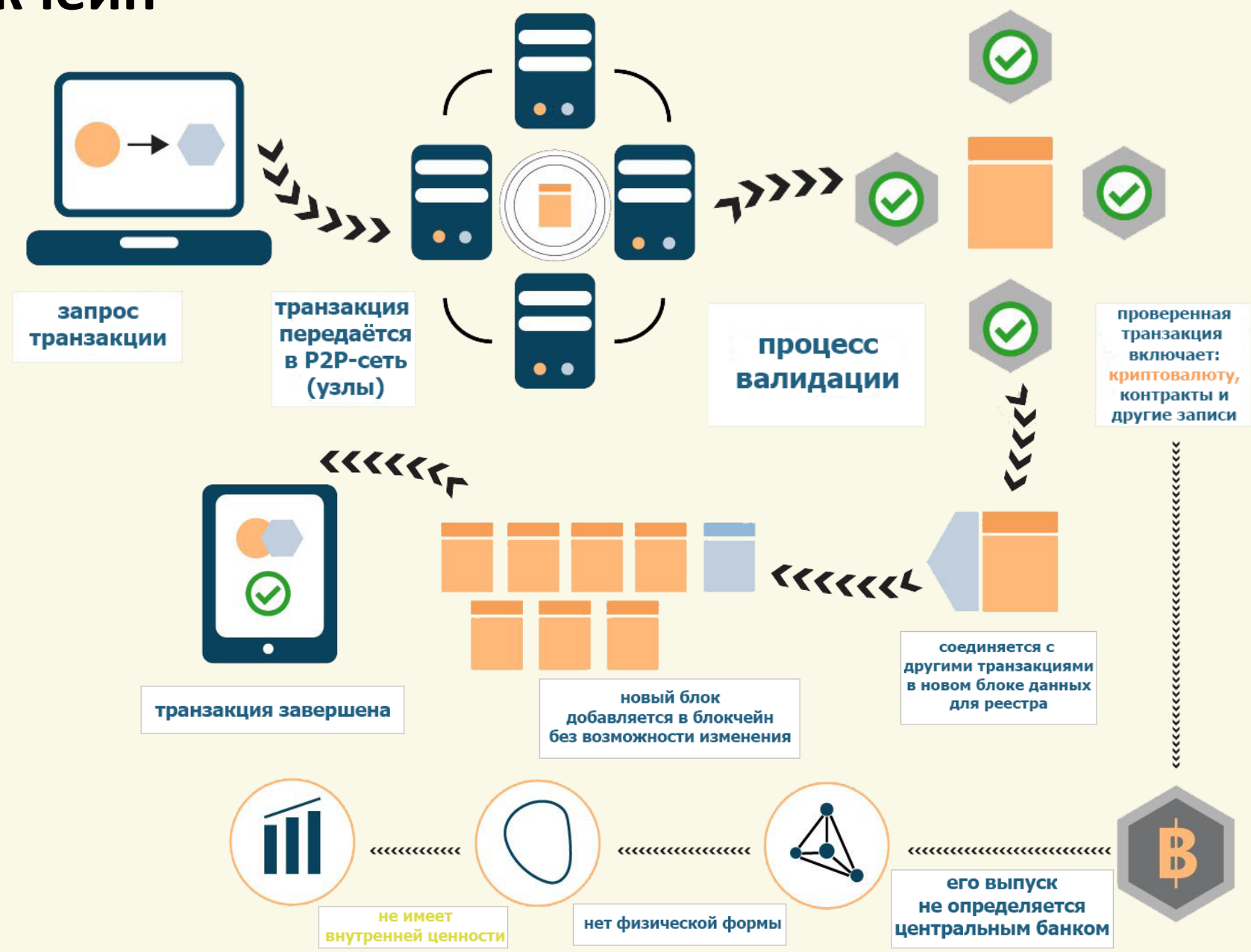
**IPFS**

**Смарт контракт**

**Крипто-кошелёк**



# Блокчейн



# Виды ног

Полная



Легкая



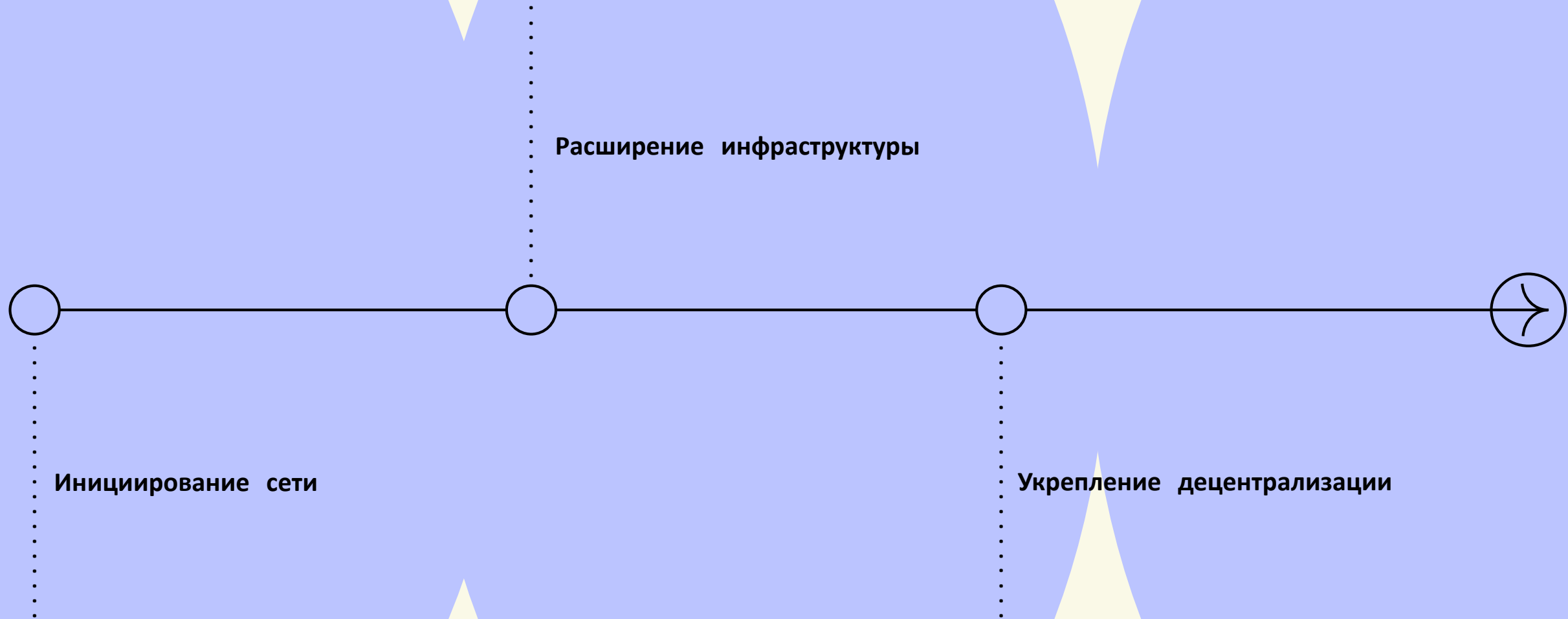
Майнинг нода



Валидационная



# Роль нод в Web3



# Консенсусные модели



**PoW**  
(Proof of Work)



**PoS**  
(Proof of Stake)



**PoH**  
(Proof of History)



**PoB**  
(Proof of Burn)



**PoI**  
(Proof of Identity)



**PoI**  
(Proof of Importance)



**PoA**  
(Proof of Activity)



**PoC**  
(Proof of Capacity)



**PoET**  
(Proof of Elapsed time)



**DPoS**  
(Delegated Proof  
of Stake)



**LPoS**  
(Leased Proof  
of Stake)



**DAG**  
(Directed Acyclic Graph)



**PBFT**  
(Practical Byzantine  
Fault Tolerance)



**DBFT**  
(Delegated Byzantine  
Fault Tolerance)

# Proof of Work vs. Proof of Stake



количество работы, выполняемой конкретным майнером, определяет его возможность добыть один блок и получить вознаграждение в виде монеты.



возможность майнинга конкретного майнера зависит от того, сколько монет у него уже есть.



с течением времени вознаграждение майнеров уменьшается. за счёт снижения стимулов уменьшается риск атаки 51%



атака 51% обходится крайне дорого

# Блокчейн – какие проблемы решил?

## Блокчейн

- ◆ Неизменяемость записей
- ◆ Прозрачность всех операций

## Ноды

- ◆ Децентрализация
- ◆ Устойчивость сети

## Консенсус

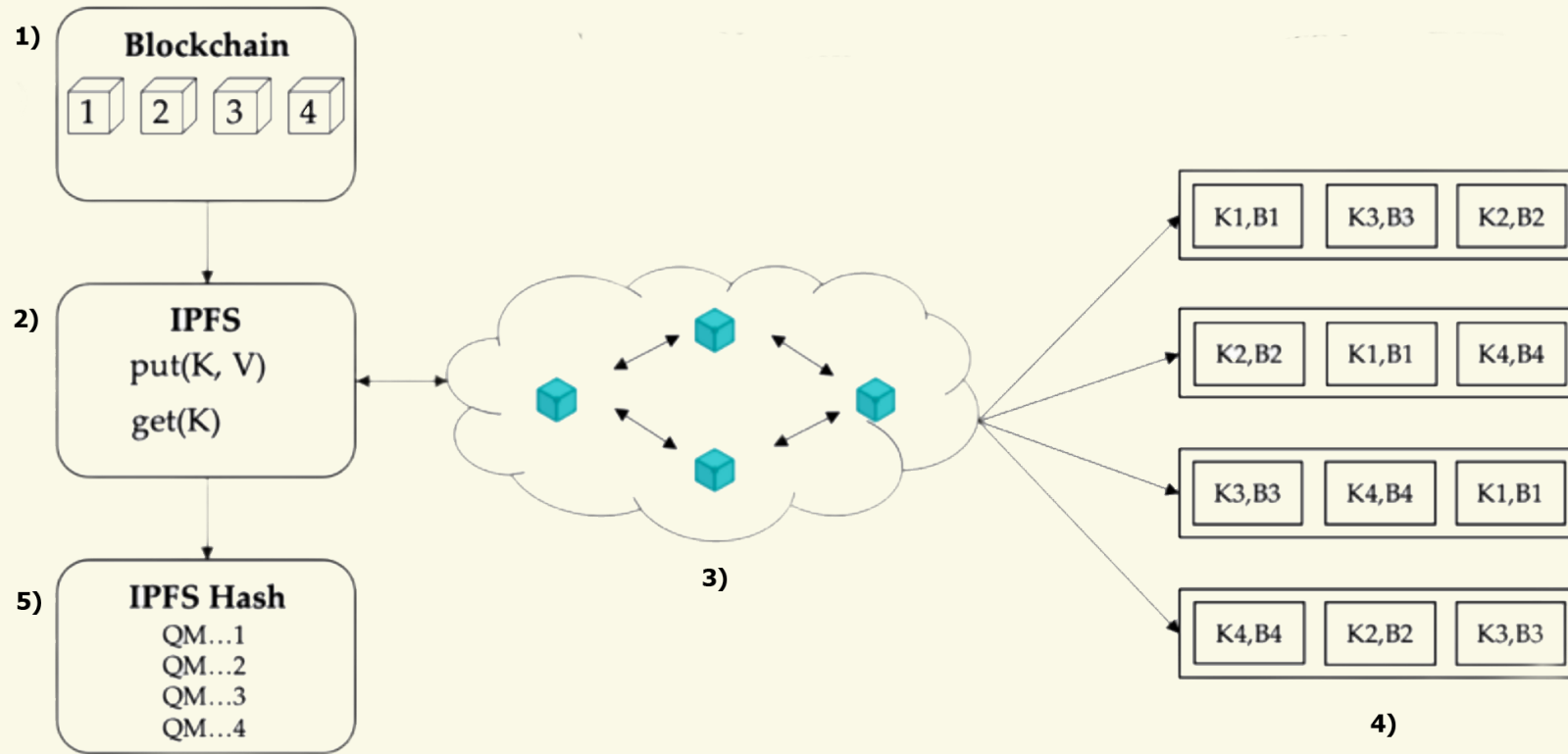
- ◆ Механизм доверия без посредников
- ◆ Коллективное подтверждение транзакций
- ◆ Защита от фальсификации

IPFS – межпланетная файловая система





# IPFS – межпланетная файловая система



# IPFS – какие проблемы решил?

---

Отсутствие централизованного хранения

---

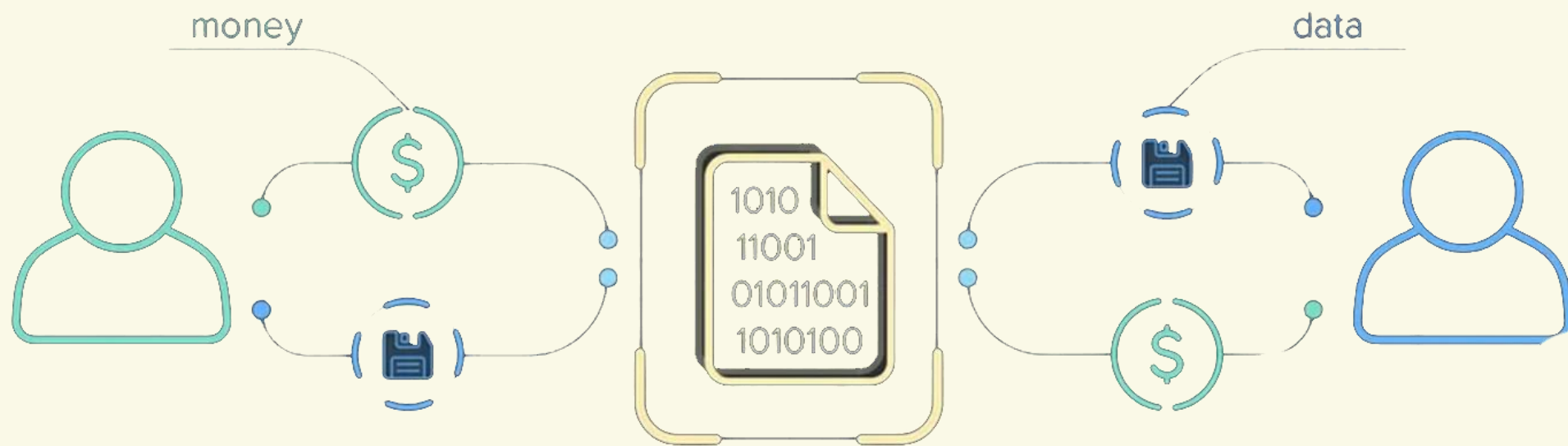
Защита от удаления и цензуры

---

Неизменяемые ссылки по хэшу

---

# Смарт-контракты



# Интеграция с Web 2



# Смарт-контракты – какие проблемы решил?

---

Автоматизация выполнения условий

---

Исключение человеческого фактора

---

Прозрачность и проверяемость логики

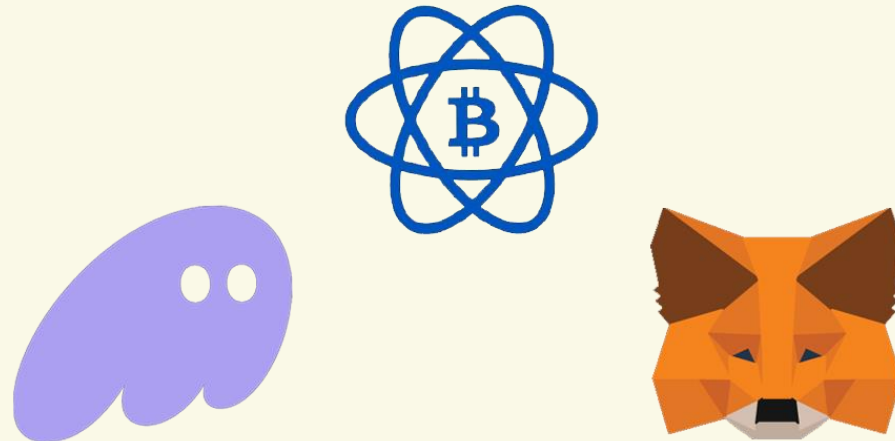
---

# Крипто-кошельки

Холодный кошелек



Горячий кошелек



# Кошелек – какие проблемы решил?

---

Самостоятельное хранение активов

---

Контроль над приватными ключами

---

Безопасные транзакции без участия платформ

---

## Итог решения :

- ◆ Доверие без посредника
- ◆ Отсутствие единой точки отказа
- ◆ Полная прозрачность операций
- ◆ Неизменяемые записи
- ◆ Открытый исходный код
- ◆ Самостоятельное хранение активов
- ◆ Отсутствие блокировок



# Как взаимодействовать с блокчейном?

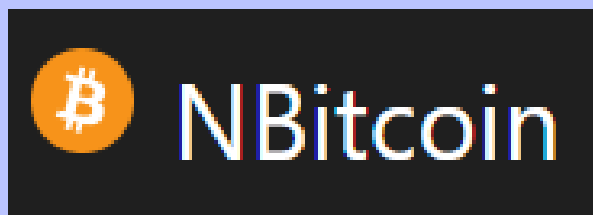
RPC (Remote Procedure Call) – взаимодействие через удалённые вызовы функций.



# Как попробовать в C#?

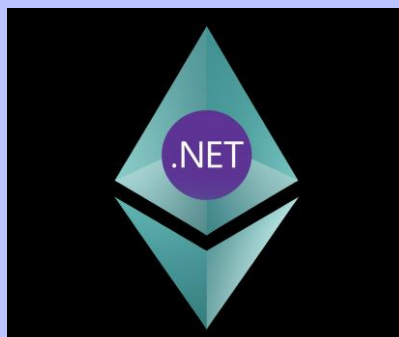
NBitcoin

C# Bitcoin library



Nethereum

Клиент для Ethereum



SolNet

SDK для Solana





# Nethereum

Добавляем RPC провайдера для дальнейшей работы

```
"Ethereum" : {  
    "RpcUrl" :  
    "https://ethereum.publicnode.com",  
    "PrivateKey" : ""  
}
```

Создание кошелька

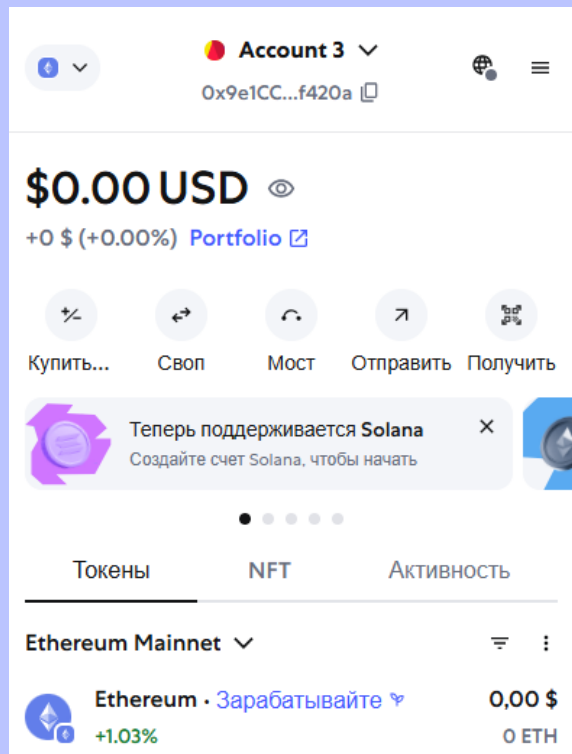
```
var wallet = new Wallet(Wordlist.English,  
    WordCount.Twelve);  
var account = wallet.GetAccount(0);  
  
return Task.FromResult(new GeneratedWallet  
{  
    SeedPhrase = string.Join(" ", wallet.Words),  
    PrivateKey = account.PrivateKey,  
    Address = account.Address  
});
```



## Response body

```
{
  "seedPhrase": "laundry follow soul diamond useless festival inform goddess possible album fee enroll",
  "privateKey": "0xe1c9b809d95c79d266d75145d7b6b0fe2dd54a08451d41c9e40a5a7451989d6f",
  "address": "0x3c3218bfD6d5019a3490B865F465A6e373f7B2b4"
}
```

Проверка – импорт в MetaMask





## Получение баланса токена по адресу кошелька

```
var contract = new StandardTokenService(_web3, tokenAddress);  
var balance = await contract.BalanceOfQueryAsync(walletAddress);  
var decimals = await contract.DecimalsQueryAsync();  
var symbol = await contract.SymbolQueryAsync();  
  
var formattedBalance = Web3.Convert.FromWei(balance, decimals);  
return (formattedBalance, symbol);
```

GET /api/Etherium/balance	
Parameters	
Name	Description
tokenAddress string (query)	0xA0b86991c6218b36c1d19D4a2e9Eb0cE3l
walletAddress string (query)	0x52aF8119BAFDE2aE55b66B1c4cED8076:

### Response body

```
{  
  "balance": 0.8,  
  "symbol": "USDC"  
}
```

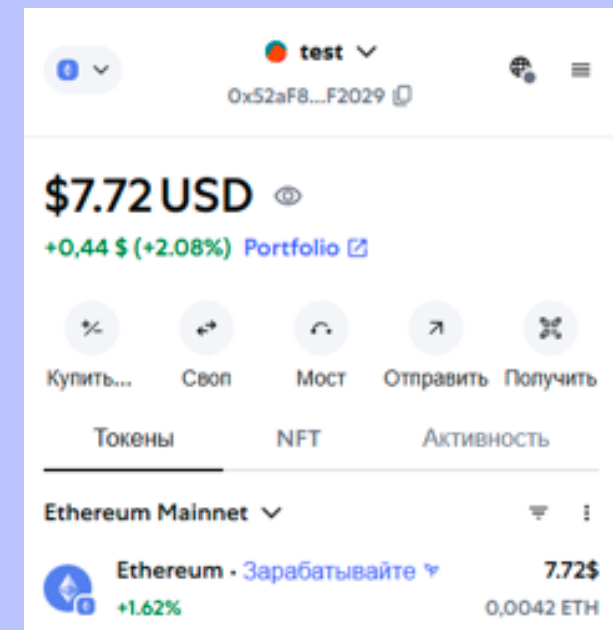
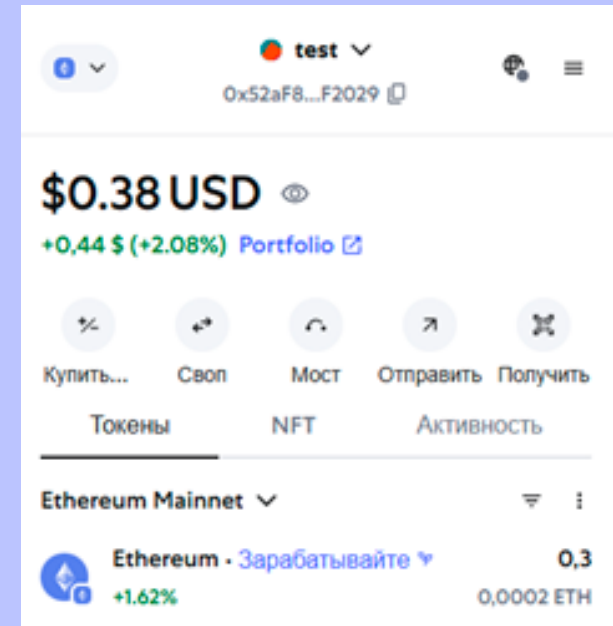
## Перевод ETH

```
var transferService =  
_web3.Eth.GetEtherTransferService();  
  
var transaction = await  
transferService.TransferEtherAndWaitForReceiptAsync(  
    toAddress: toAddress,  
    etherAmount: amount);
```

**POST** /api/Ethereum/eth/transfer

**Parameters**

Name	Description
toAddress string (query)	0x52aF8119BAFDE2aE55b66B1c4cED8076:
amount number(\$double) (query)	0.004



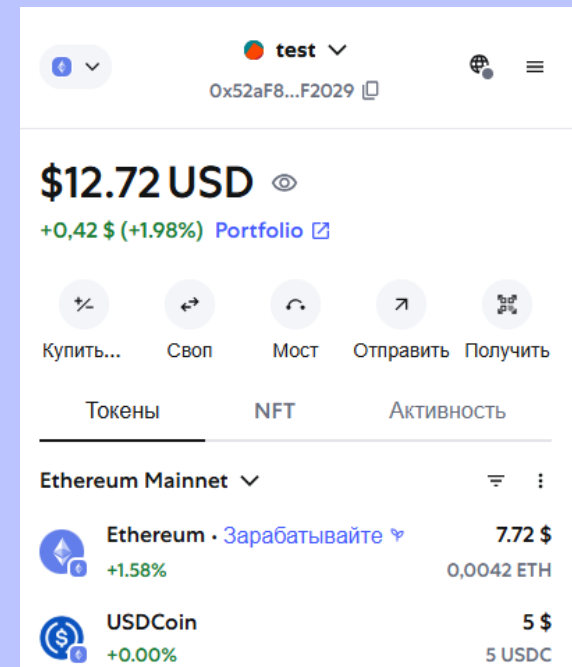
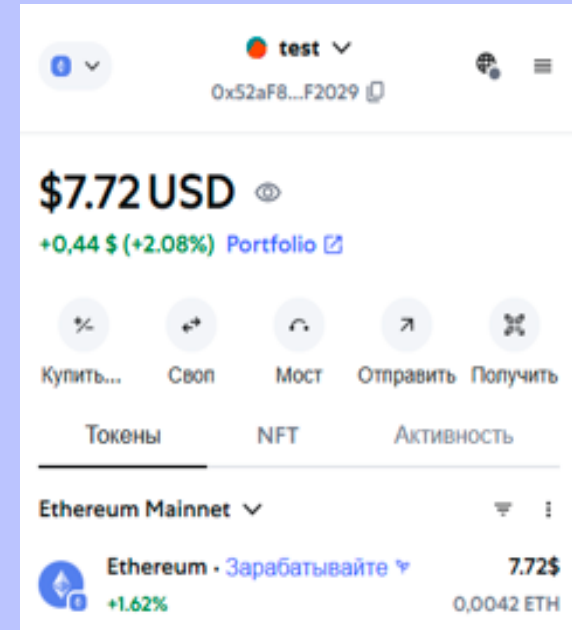
## Перевод токена

```
var contract = new StandardTokenService(_web3, tokenAddress);  
  
var decimals = await contract.DecimalsQueryAsync();  
var weiAmount = Web3.Convert.ToWei(amount, decimals);  
  
var receipt = await  
contract.TransferRequestAndWaitForReceiptAsync(toAddress,  
weiAmount);
```

**POST** /api/Etherium/token/transfer

**Parameters**

Name	Description
tokenAddress string (query)	0xA0b86991c6218b36c1d19D4a2e9Eb0cE3f
toAddress string (query)	0x52aF8119BAFDE2aE55b66B1c4cED8076:
amount number(\$double) (query)	5





## Получение событий

```
[Event("Transfer")]
public record TransferEventDTO : IEventDTO
{
    [Parameter("address", "from", 1, true)]
    public string? From { get; set; }

    [Parameter("address", "to", 2, true)]
    public string? To { get; set; }

    [Parameter("uint256", "value", 3, false)]
    public BigInteger Value { get; set; }

    public DateTime TransactionTime { get; set; }
}
```



Создаем фильтр для получения события перевода USDC



```
private const string UsdcContractAddress =  
"0xA0b86991c6218b36c1d19D4a2e9Eb0cE3606eB48";
```

```
var transferEvent =  
web3.Eth.GetEvent<TransferEventDTO>(UsdcContractAddress);
```

```
var filterInput = transferEvent.CreateFilterInput(  
    fromBlock: BlockParameter.CreateLatest(),  
    toBlock: BlockParameter.CreateLatest()  
);
```

```
var filterId = await transferEvent.CreateFilterAsync(filterInput);
```



```
var eventLogs = await transferEvent.GetFilterChangesAsync(filterId);
foreach (var eventLog in eventLogs)
{
    var block = await web3.Eth.Blocks
        .GetBlockWithTransactionsByNumber
        .SendRequestAsync(eventLog.Log.BlockNumber);
}
```

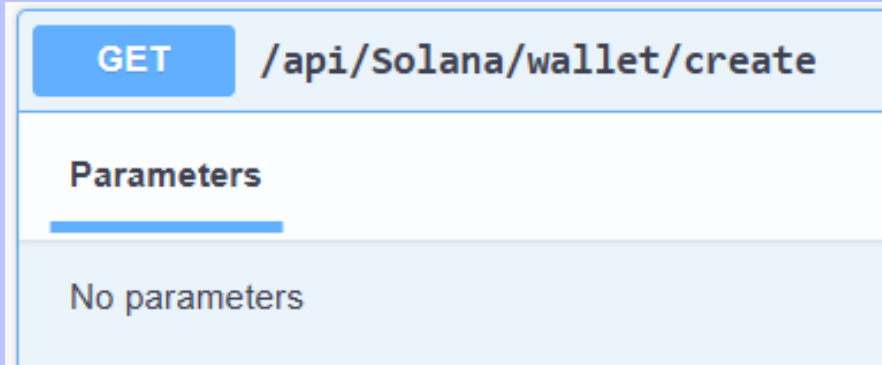
```
Transfer event:
info: PoCWeb3.Services.TokenTransferListener[0]
      Transfer from 0x7176f0F071379FeE51668Eb6387ddA9129E5CA6B to 0xC965BC2cf23D54ccF41a7d3aF10572580E9B8078, amount 0,000000000158998644 at 26.04.2025 13:23:35
info: PoCWeb3.Services.TokenTransferListener[0]
      Transfer event:
info: PoCWeb3.Services.TokenTransferListener[0]
      Transfer from 0xe35Bfbf439D7C37E2Df41BF1236cCf1dEc0543fd to 0x73a8a6F5d9762EA5f1de193eC19CdF476C7e86b1, amount 0,000000006010519315 at 26.04.2025 13:23:35
```



# Solnet

Создание кошелька

```
private readonly IRpcClient _rpcClient =  
ClientFactory.GetClient(Cluster.DevNet);
```



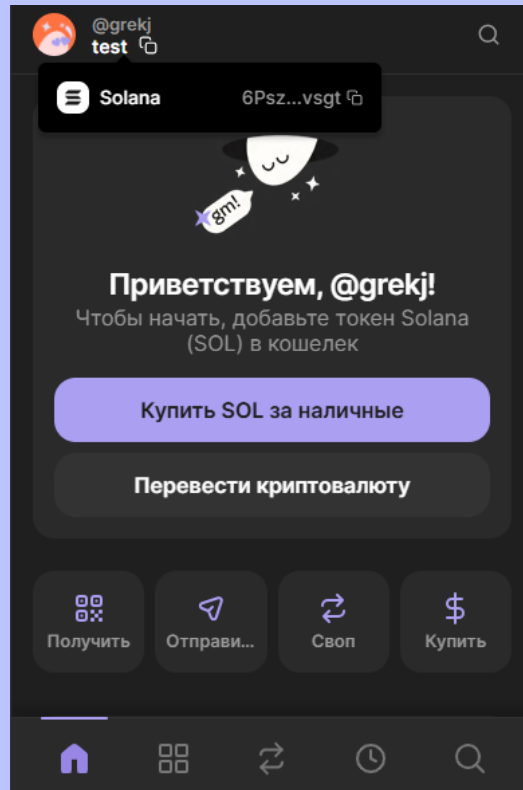
```
var mnemonic = new Mnemonic(WordList.English, WordCount.Twelve);  
var wallet = new Wallet(mnemonic);  
var account = wallet.GetAccount(0);  
  
return new GeneratedWallet  
{  
    SeedPhrase = mnemonic.ToString(),  
    Address    = account.PublicKey,  
    PrivateKey =  
Encoders.Base58.EncodeData(account.PrivateKey.KeyBytes)  
};
```



### Response body

```
{
  "seedPhrase": "leaf rib design lawn will mobile dry brief vapor access ensure tribe",
  "privateKey": "4qTXjPUA4QJeHwLYJovusYCx5V5CVwr7a5T3w6zh8iBTx4fMSSAsnPHuVhM4pSCess1s4mUQvGbNMyiDkGC3rCEg",
  "address": "3eAhJb3oi5JL8XtHbCkA1quPxvWCjJ3KJbZGEHSM9Wg4"
}
```

## Импорт в Phantom



## Получение тестовых токенов



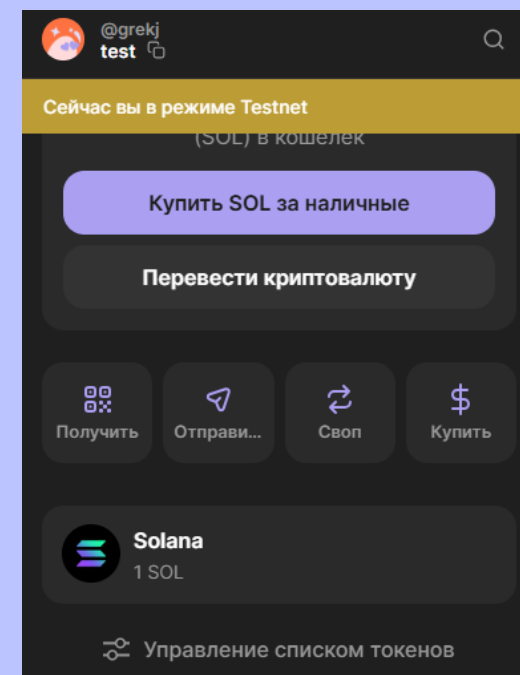
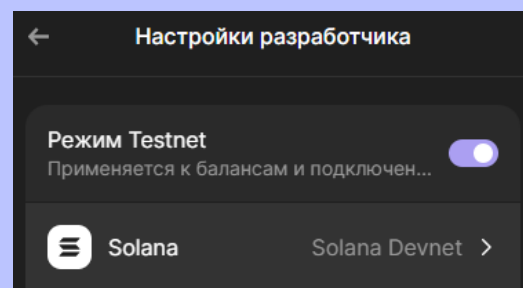
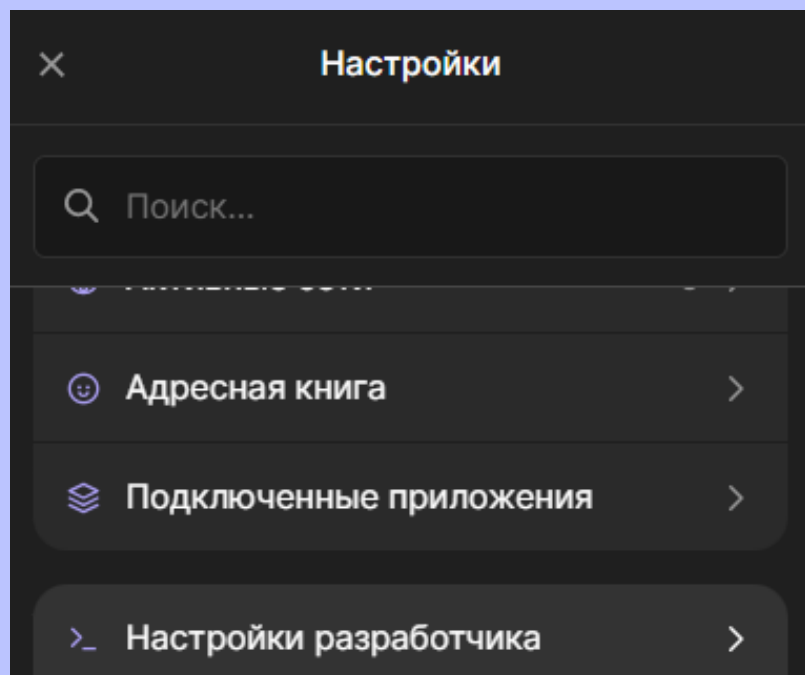
GET /api/Solana/faucet	
Parameters	
Name	Description
walletAddress string (query)	6PsZzeUY2XaxNNKArp3use1RGnV99umfNI

```
public async Task<IActionResult> GetTestSol([FromQuery] string  
walletAddress)  
{  
    await solana.RequestTestSol(walletAddress);  
}
```

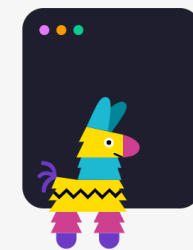


## Response body

```
{  
  "message": "Test SOL has been sent",  
  "walletAddress": "6PszZeUY2XaxNNKArp3use1RGnV99umfNMie44Dtvsgt"  
}
```



## Добавление файлов в IPFS - Pinata



```
public async Task<string> AddFile(Stream fileStream, string fileName, string name, string network)
{
    using var content = new MultipartFormDataContent();
    content.Add(new StreamContent(fileStream), "file", fileName);
    var metadata = JsonSerializer.Serialize(new { name });
    content.Add(new StringContent(metadata, Encoding.UTF8, "application/json"), "pinataMetadata");
    content.Add(new StringContent(network), "public");

    var response = await _http.PostAsync("/pinning/pinFileToIPFS", content, ct);

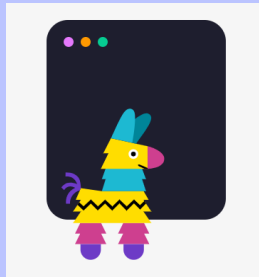
    await using var jsonStream = await response.Content.ReadAsStreamAsync(ct);
    using var doc = await JsonDocument.ParseAsync(jsonStream, cancellationTokens: ct);
    return doc.RootElement.GetProperty("IpfsHash").GetString()!;
}
```

```
{
  "cid": "QmcmaLcfss8vEGh6bhCqCSCQpBwsH5hdzWA2Q4BeVrMkBY",
  "name": "spbDotNetLogo",
  "url": "https://gateway.pinata.cloud/ipfs/QmcmaLcfss8vEGh6bhCqCSCQpBwsH5hdzWA2Q4BeVrMkBY"
}
```

Собираем файл метадаты для токена:


```
{
  "name": "SpbDotNet Token",
  "symbol": "SPBDN",
  "description": "Utility token for the SpbDotNet community.",
  "image":
    "https://gateway.pinata.cloud/ipfs/QmcmaLcfss8vEGh6bhCqCSCQpBwsH5hdzWA2Q4BeVrMkBY",
  "seller_fee_basis_points": 0,
  "external_url": "https://t.me/spbdotnet",
  "attributes": [],
  "collection": null,
  "creators": [
    {
      "address": "6PszZeUY2XaxNNKArp3use1RGnV99umfNMie44Dtvsgt",
      "share": 100
    }
  ]
}
```

```
{
  "cid": "QmV4yLfenHZdt8Y9dh63K5PZQw1vW7Y5NWmXRJirRMTd5Y",
  "name": "tokenMetadata",
  "url": "https://gateway.pinata.cloud/ipfs/QmV4yLfenHZdt8Y9dh63K5PZQw1vW7Y5NWmXRJirRMTd5Y"
}
```







Pinata

Workspace

Sidiropulos's Workspace

▼

IPFS

Files

Groups

Gateways

Analytics

DEVELOPER

API Keys

Webhooks

Access Controls

EXTENSIONS

Marketplace

Integrations

FILES

PUBLIC

PRIVATE

Public files are accessible via IPFS

salmon-quiet-mockingbird-54.myp...

+ Add

Q

Search files and CIDs

	NAME	CID	SIZE	CREATION DATE	FILE ID
<input type="checkbox"/>	tokenMetadata	QmV4y...MTd5Y	471 B	7/12/2025	<div><div></div><div></div></div>
<input type="checkbox"/>	spbDotNetLogo	Qmcma...rMkBY	74.08 KB	7/12/2025	<div><div></div><div></div></div>
<input type="checkbox"/>	metadata.json	bafkr...nib4q	475 B	5/18/2025	<div><div></div><div></div></div>
<input type="checkbox"/>	logo.png	bafkr...phdhm	74.06 KB	5/18/2025	<div><div></div><div></div></div>

Rows per page: 10

41

## Создание собственного токена



**POST** `/api/Solana/token/create-token`

**Parameters**

Name	Description
privateKey string (query)	<input type="text" value="5sZFdmvJqEDMcKpnotqumBtUXz9p1AEHd2"/>
decimals integer(\$int32) (query)	<input type="text" value="9"/>
amount integer(\$int64) (query)	<input type="text" value="1000"/>
metadataUri string (query)	<input type="text" value="https://gateway.pinata.cloud/ipfs/QmV4yLfenl"/>



```
var rentRes = await
_rpcClient.GetMinimumBalanceForRentExemptionAsync(TokenProgram.MintAccountDataSize);
var recent = await _rpcClient.GetLatestBlockHashAsync();
var blockHash = recent.Result.Value.Blockhash;
```

```
var createMintIx = SystemProgram.CreateAccount(
    fromAccount: account.PublicKey,
    newAccountPublicKey: mintKey.PublicKey,
    lamports: rentRes.Result,
    space: TokenProgram.MintAccountDataSize,
    programId: TokenProgram.ProgramIdKey
);
```

```
var initMintIx = TokenProgram.InitializeMint(
    mint: mintKey.PublicKey,
    decimals: (byte)decimals,
    mintAuthority: account.PublicKey,
    freezeAuthority: account.PublicKey
);
```



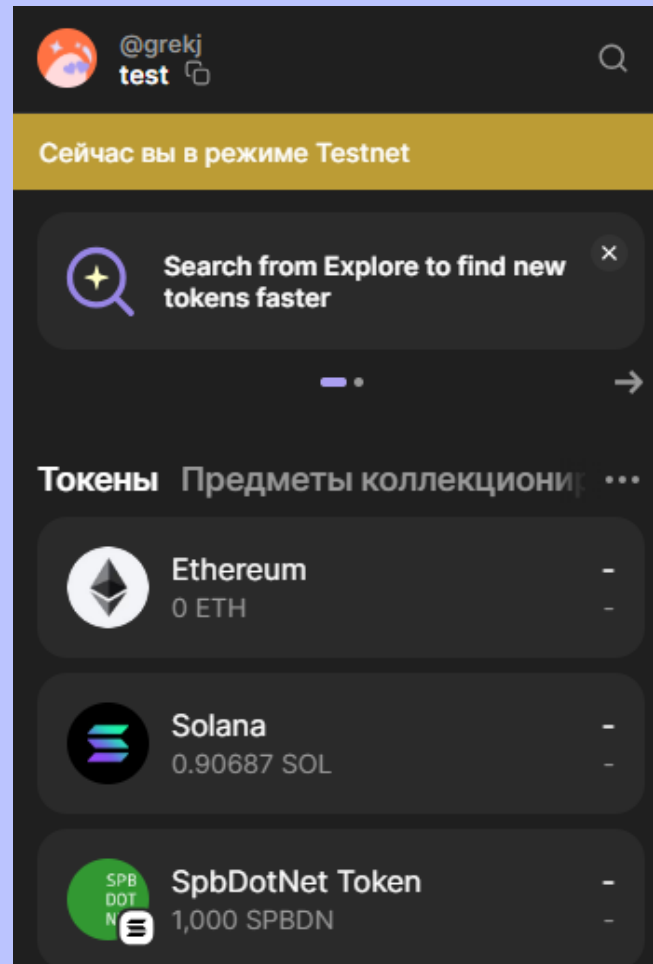
```
var metadata = new Metadata
{
    uri = metadataUri,
    // name добавляется в блокчейн
    // кошельки читают из метадаты
    name = "SpbDotNet Token",
    symbol = "SPBDN",
    sellerFeeBasisPoints = 0,
    creators =
    [
        new Creator(
            key: account.PublicKey,
            share: 100,
            verified: true
        )
    ],
    collection = null,
    uses = null,
    programmableConfig = null
};
```



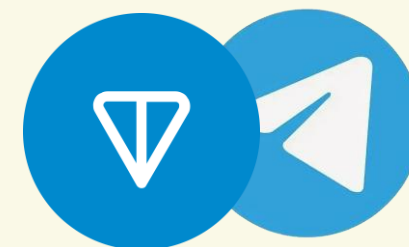
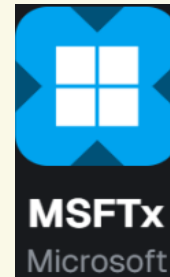
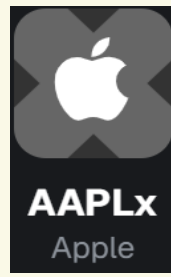
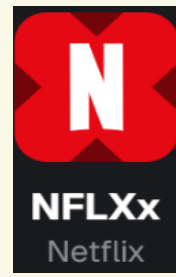
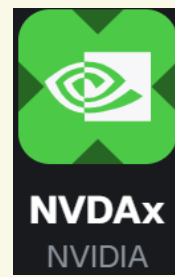
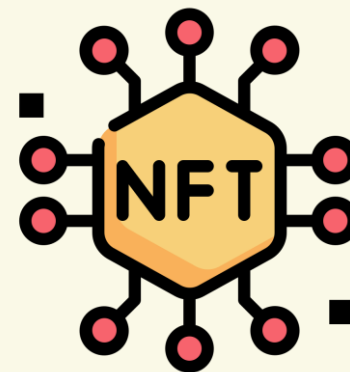
```
var metadataIx =  
MetadataProgram.CreateMetadataAccount(  
  metadataKey: metadataAccount,  
  mintKey: mintKey.PublicKey,  
  authorityKey: account.PublicKey,  
  payerKey: account.PublicKey,  
  updateAuthority: account.PublicKey,  
  data: metadata,  
  tokenStandard: TokenStandard.FungibleAsset,  
  isMutable: true,  
  updateAuthorityIsSigner: true  
);
```



```
var createToken = new TransactionBuilder()  
    .SetRecentBlockHash(recentBlockHash: blockHash)  
    .SetFeePayer(account: account.PublicKey)  
    .AddInstruction(instruction: createMintIx)  
    .AddInstruction(instruction: initMintIx)  
    .AddInstruction(instruction: createAtalx)  
    .AddInstruction(instruction: mintToIx)  
    .AddInstruction(instruction: metadataIx)  
    .Build(signers: new List<Account> { account, mintKey });
```



## Заключение: Потенциал Web3





Сделать своими руками



# Список источников

## Для общего понимания:

- <https://academy.binance.com/ru> — простые объяснения Web3, DeFi, NFT, блокчейна и смарт-контрактов от Binance
- <https://chain.link/education> — Введение в ключевые концепты блокчейн-технологий
- <https://www.coinbase.com/learn> — материалы об основах блокчейна и DeFi.

## Для разработки:

- Nethereum Docs — <https://docs.nethereum.com/en/latest/>
- Solnet GitHub — <https://github.com/bmresearch/Solnet>
- Moralis Web3 Docs — <https://docs.moralis.io/>
- Infura API — <https://www.infura.io/docs>
- Pinata API — <https://docs.pinata.cloud/>

## Попробовать самому:

- Merkle tree — <https://github.com/lontivero/MerkleTools> & <https://github.com/cliftonm/MerkleTree>
- ZK Proofs — <https://github.com/asagynbaev/ZkpSharp>
- Mini Dao — <https://www.binance.com/ru/square/post/97946>

# Исходный код



<https://github.com/sidiropulo>