

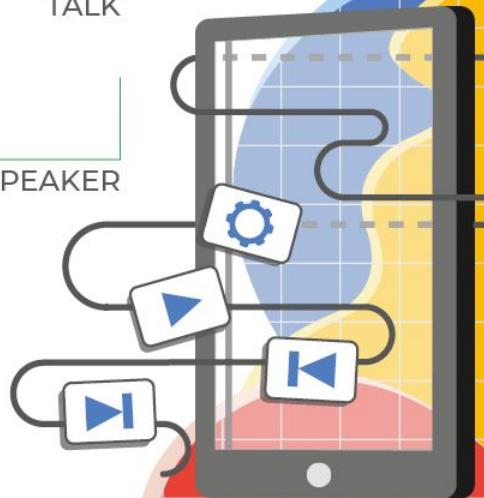


BLOCKCHAIN Smart Contract from Scratch

Alessandro Berti
Luca Corbucci
Eugenio Paluello



TALK
SPEAKER



Who are we?



Alessandro Berti

ICT Solutions Architect

Computer Science Student
@University of Pisa



Luca Corbucci

Computer Science Student
@University of Pisa



Eugenio Paluello

ICT Solutions Architect

Computer Science Student
@University of Pisa

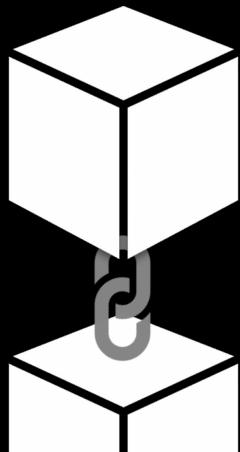
What are we gonna do?



- **Let's Grasp the Idea!** (Theory)
- **Smart Contracts**
- **Deploy your Contract!**

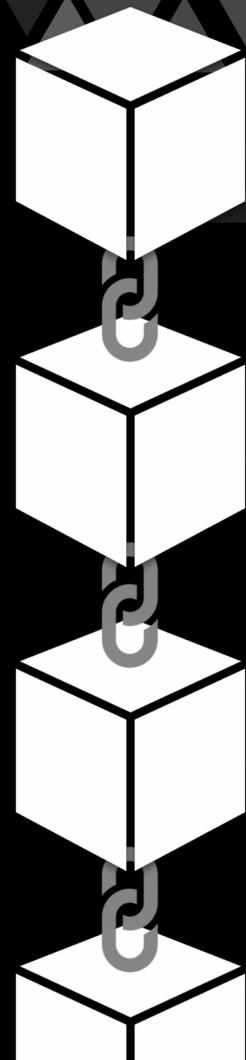


Let's Grasp the Idea!



What is ?

“The Blockchain is
a **shared, trusted, public**
ledger of **transactions**
grouped into **blocks**”



Transaction

Atomic transfer of "some-kind-of-value" between two or more entities.



ADDRESS_1

36mdwq3JZHjpJMhokn4rVeeuzHTXGukX1L

ADDRESS_2

14TDxS9NQhiu1tX4qeFwRkisXn1oDmTuKB



Transaction

Atomic transfer of "some-kind-of-value" between two or more entities.



ADDRESS_1

36mdwq3JZHjpJMhokn4rVeeuzHTXGukX1L

ADDRESS_2

14TDxS9NQhiu1tX4qeFwRkisXn1oDmTuKB



Mining



Mining (PoW)



Which is that value “**nonce**” such that:

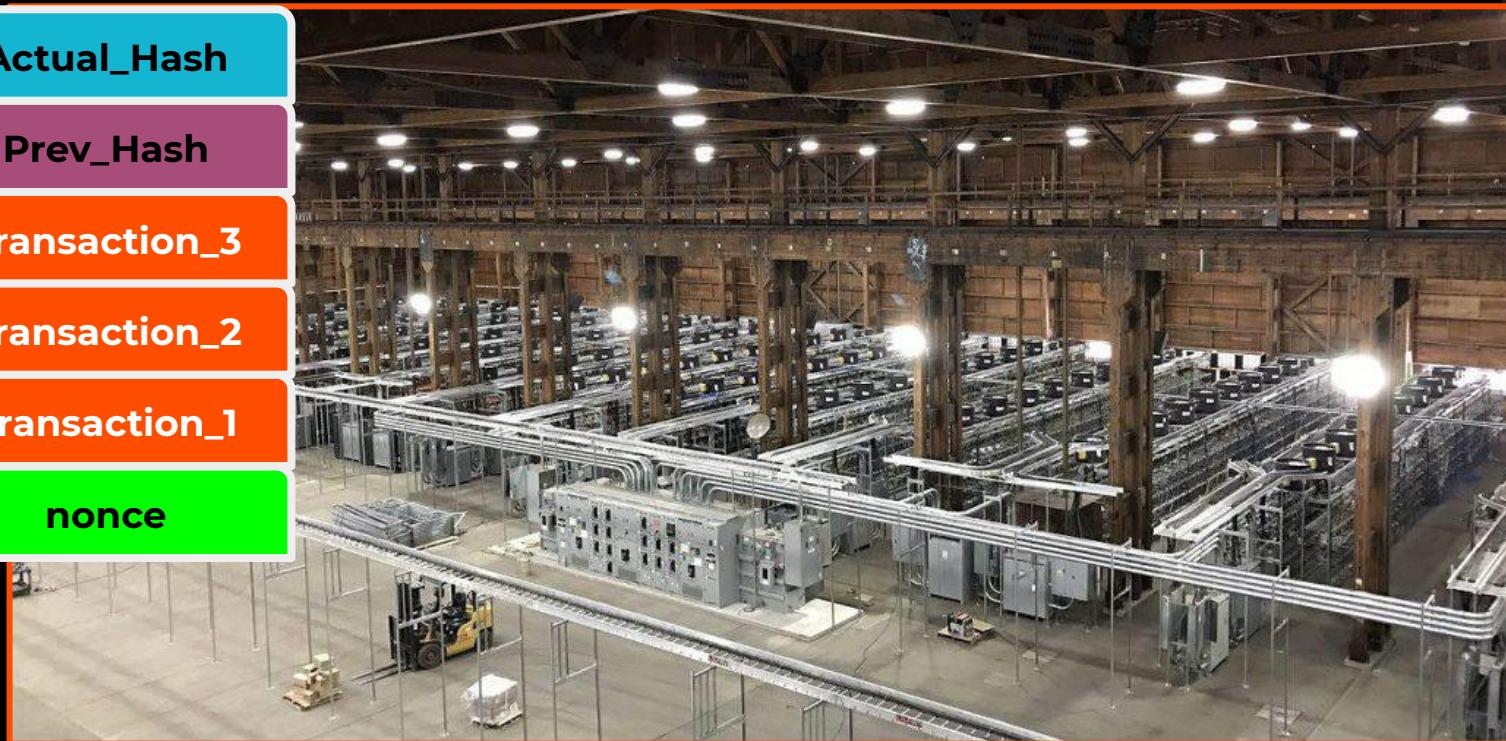
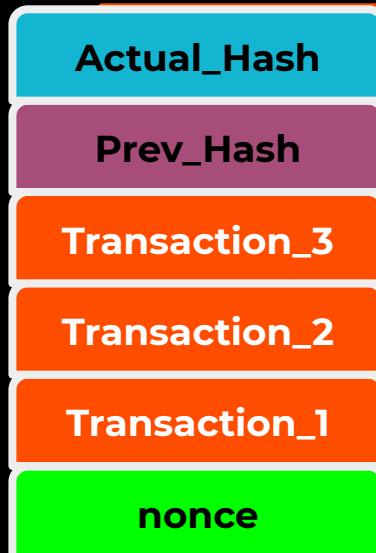
$$\text{Hash}(\text{Tx_1}, \text{Tx_2}, \text{Tx_3}, \text{Prev_Hash}, \text{nonce}) < \text{Difficulty}$$

If found then

$$\text{Hash_ActBlock} = \text{Hash}(\text{Tx_1}, \text{Tx_2}, \text{Tx_3}, \text{Prev_Hash}, \text{nonce})$$

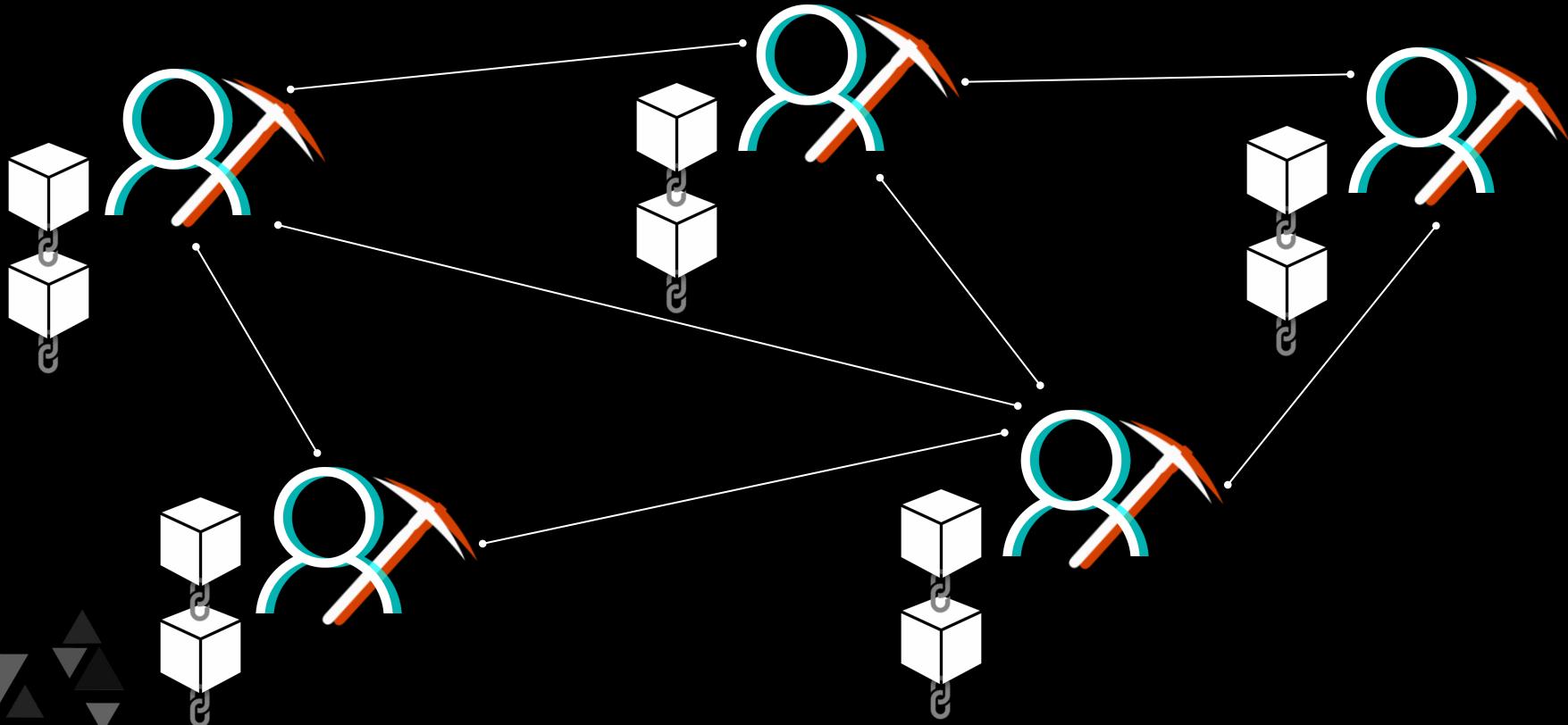


Mining (PoW)

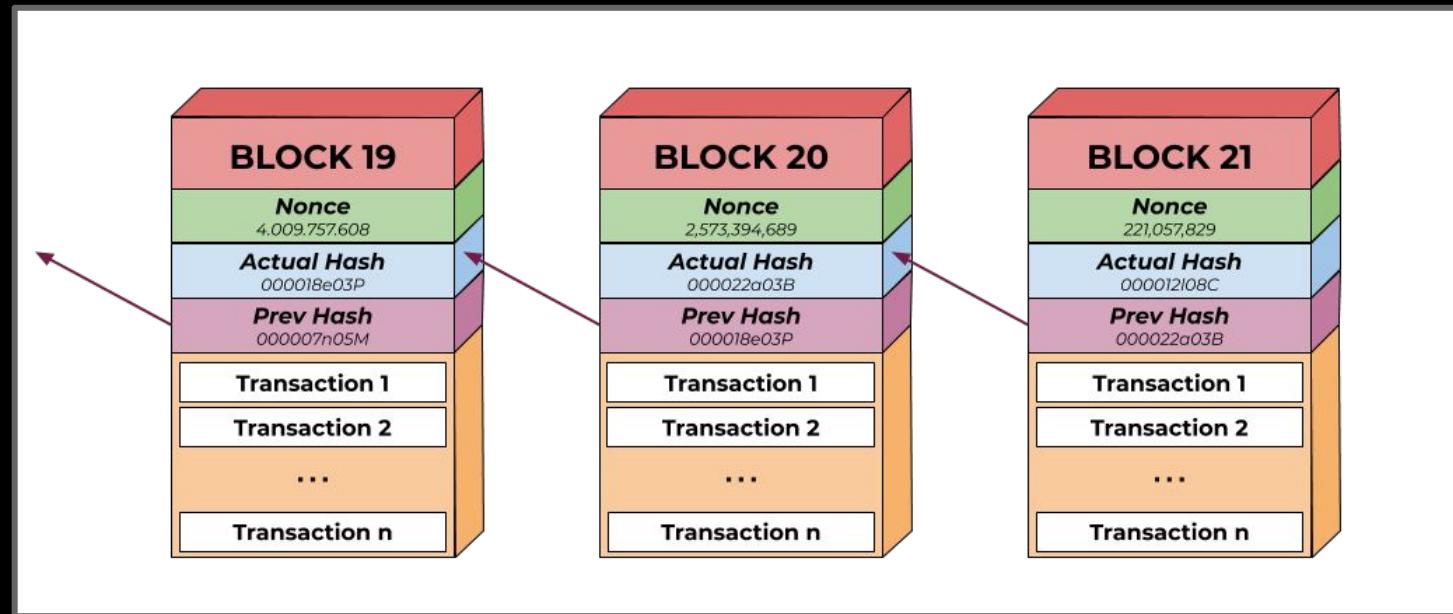


Shared

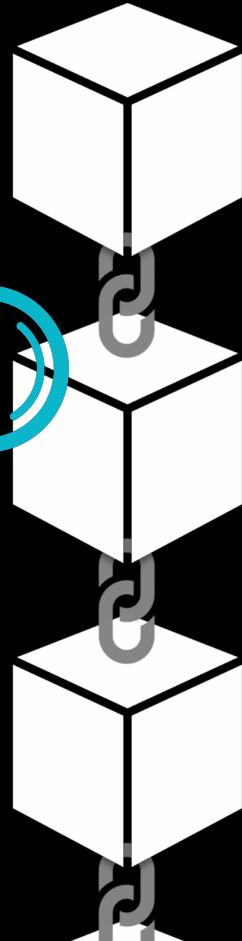
The blockchain is shared across the miners.
Namely: each miner has a consistent *replica* of the blockchain



Trusted



Public



BLOCKCHAIN.COM Prodotti Dati Esploratore Q Accesso Iscriviti

Le transazioni

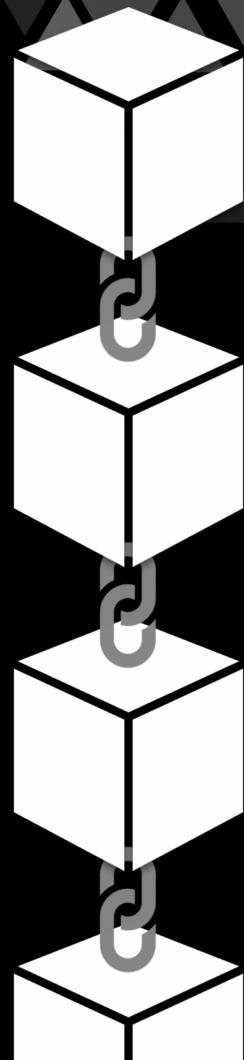
Standard Interno

1 2 3 4 5 Il prossimo +10

hash	tassa	data	status	valore
0x03411439f603b54f4cca767019fa89f645e2682f362591bc77913...	0.000129075 ETH (43025 GAS - 3000000000 WEI)	2020-04-12 23:12	0xa0b86991c6218b36c1d19d4a2e9eb0ce3606eb48	1 Conferma 0.00000000 ETH
0x06fb8a4225b61f7575ae9f03109539daf4c5624641d6406b21b42...	0.00015503 ETH (31006 GAS - 5000000000 WEI)	2020-04-12 23:12	0x693c188e40f780ecf00d2946ef45260b84fbca3e	1 Conferma 0.00000000 ETH
0x0c215222b2237ce2cb03263065753c131341322e95c334d8b7a...	0.00141332026 ETH (242422 GAS - 5830000000 WEI)	2020-04-12 23:12	0x794e6e91555438afc3ccf1c5076a74f42133d08d	1 Conferma 0.00000000 ETH

What is ?

“The Blockchain is
a **shared, trusted, public**
ledger of **transactions**
grouped into **blocks**”



Examples



bitcoin



ethereum

Transaction Scheme Model

UTXO

Account Based

Consensus Algorithm

PoW

PoW¹

Block Validation Time

10 mins

15 secs

Turing Complete

No

Yes²



¹Moving to PoS

²Actually, "Rich-Statefulness"

Smart Contracts



ethereum

What is ?



A **smartcontract** is a computer protocol intended to digitally **facilitate**, **verify**, or **enforce** the negotiation or performance of a **contract**.



Smart contracts help you exchange money, property, shares, or anything of value in a transparent, conflict-free way while avoiding the services of a middleman.

Deploy Your Contract!

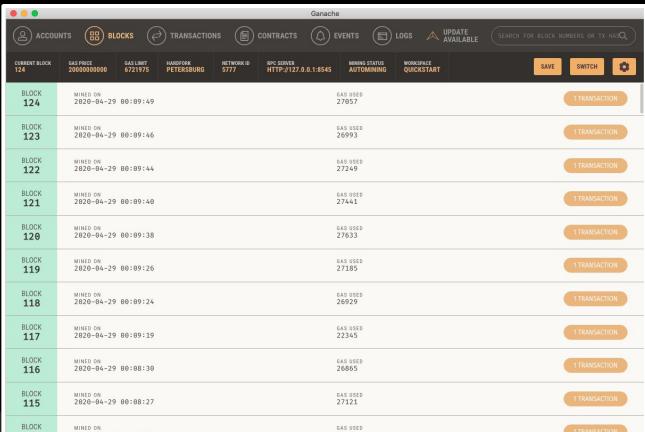


ethereum

Test Net vs Ganache

Ganache

Ganache creates a **fake blockchain** that runs on your machine. All the **transactions are immediately mined**.



The screenshot shows the Ganache interface with a list of recently mined blocks. The current block is 124. The table includes columns for Block Number, Mined On, Gas Used, and a '1 TRANSACTION' button.

Block	Mined On	Gas Used	Action
124	2020-04-29 00:09:49	27957	1 TRANSACTION
123	2020-04-29 00:09:46	26993	1 TRANSACTION
122	2020-04-29 00:09:44	27249	1 TRANSACTION
121	2020-04-29 00:09:40	27442	1 TRANSACTION
120	2020-04-29 00:09:38	27633	1 TRANSACTION
119	2020-04-29 00:09:26	27385	1 TRANSACTION
118	2020-04-29 00:09:24	26929	1 TRANSACTION
117	2020-04-29 00:09:19	22345	1 TRANSACTION
116	2020-04-29 00:08:39	26865	1 TRANSACTION
115	2020-04-29 00:08:27	27121	1 TRANSACTION
Block	Mined On	Gas Used	All Transactions

Test Net

If you want to try your contract in a **“real” world** you can use the Ethereum Test Net.

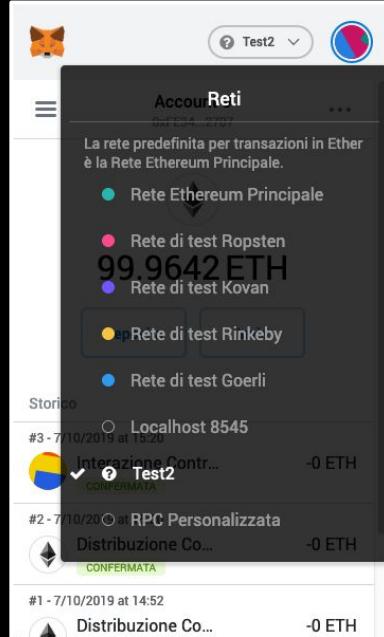
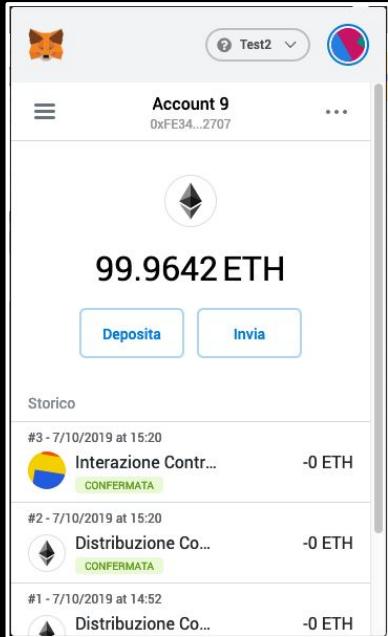
The **transactions are mined after a few seconds**.

There exists several Test net:

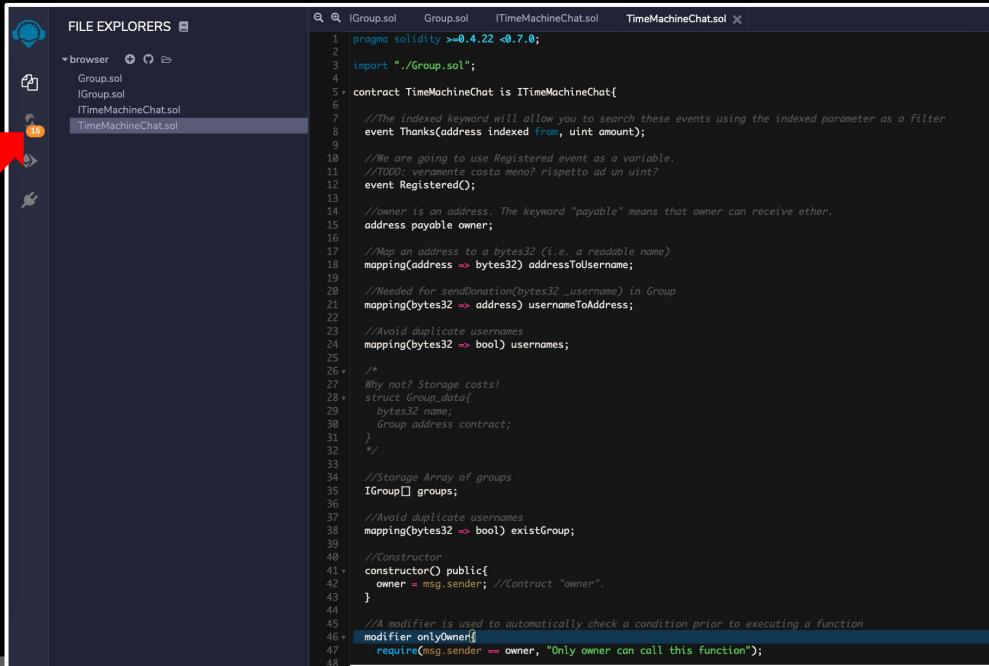
- Ropsten
- Kovan
- Rinkeby
- Gorli



MetaMask is an extension for accessing Ethereum enabled distributed applications, in your browser.



Remix is an **open source tool** that helps you **write Solidity contracts** from the **browser**.



```
FILE EXPLORERS
browser
Group.sol
IGroup.sol
ITimeMachineChatsol
TimeMachineChat.sol

pragma solidity >=0.4.22 <0.7.0;

import "./Group.sol";

contract TimeMachineChat is ITimeMachineChat{
    //The indexed keyword will allow you to search these events using the indexed parameter as a filter
    event Thanks(address indexed from, uint amount);

    //We are going to use Registered event as a variable.
    //TODO: veramente costa meno? rispetto ad un uint?
    event Registered();

    //owner is an address. The keyword "payable" means that owner can receive ether.
    address payable owner;

    //Map an address to a bytes32 (i.e. a readable name)
    mappingAddress => bytes32 addressToUsername;
    //Needed for sendDonation(bytes32 _username) in Group
    mapping(bytes32 => address) usernameToAddress;

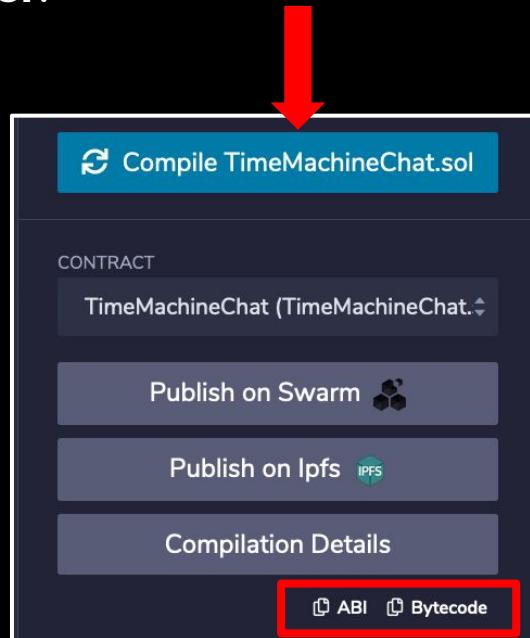
    //Avoid duplicate usernames
    mapping(bytes32 => bool) usernames;

    /**
     * Why not? Storage costs!
     */
    struct Group_data{
        bytes32 name;
        Group address contract;
    }

    //Storage Array of groups
    IGroup[] groups;
    //Avoid duplicate usernames
    mapping(bytes32 => bool) existGroup;

    //Constructor
    constructor() public{
        owner = msg.sender; //Contract "owner".
    }

    //A modifier is used to automatically check a condition prior to executing a function
    modifier onlyOwner{
        require(msg.sender == owner, "Only owner can call this function");
    }
}
```



Remix also supports **testing, debugging and deploying** of smart contracts.

The screenshot shows the Remix IDE interface. On the left, there's a sidebar with "DEPLOY & RUN TRANSACTIONS" settings: Environment (Web3 Provider, Custom (5777) network), Account (0x88a...e5312 (99.9433453799904)), Gas Limit (3000000), Value (0 wei), and a Deploy button. Below it are options to Publish to IPFS or Deploy at Address. The Deployed Contracts section shows "TIMEMACHINECHAT AT 0X84F...BB5DC (BLOCKCHAIN)". The main area has tabs for IGroup.sol, Group.sol, iTimeMachineChat.sol, and TimeMachineChat.sol. The TimeMachineChat.sol tab displays the Solidity code for a smart contract. At the bottom, the Transaction History shows two transactions: one successful deployment and one pending creation of the TimeMachineChat contract.

```
// SPDX-License-Identifier: MIT
pragma solidity ^0.8.0;

//Storage Array of groups
IGroup[] groups;

//Avoid duplicate usernames
mapping(bytes32 => bool) existGroup;

//Constructor
constructor() public{
    owner = msg.sender; //Contract "owner".
}

//A modifier is used to automatically check a condition prior to executing a function
modifier onlyOwner{
    require(msg.sender == owner, "Only owner can call this function");
    -
}

//check whether msg.sender is already registered
modifier addressAlreadyRegistered(){
    require(AddressToUsername[msg.sender] == bytes4(0x0), "Address already registered");
    -
}

//check that "username" is not already taken
modifier usernameTaken(bytes32 _username){
    require(usernames[_username], "Username not available");
    -
}

modifier isRegisteredOf{
    require(AddressToUsername[tx.origin] != bytes4(0x0), "Address is not registered");
    -
}
```

Transactions recorded: 1

Deployed Contracts: TIMEMACHINECHAT AT 0X84F...BB5DC (BLOCKCHAIN)

creation of TimeMachineChat pending...

block:1 txIndex:0 from:0xF23...D2707 to:TimeMachineChat.(constructor) value:0 wei data:0x608...10033 logs:0 hash:0x98a...19598 Debug

block:125 txIndex:0 from:0x88a...e5312 to:TimeMachineChat.(constructor) value:0 wei data:0x608...10033 logs:0 hash:0xdb...c38e6 Debug

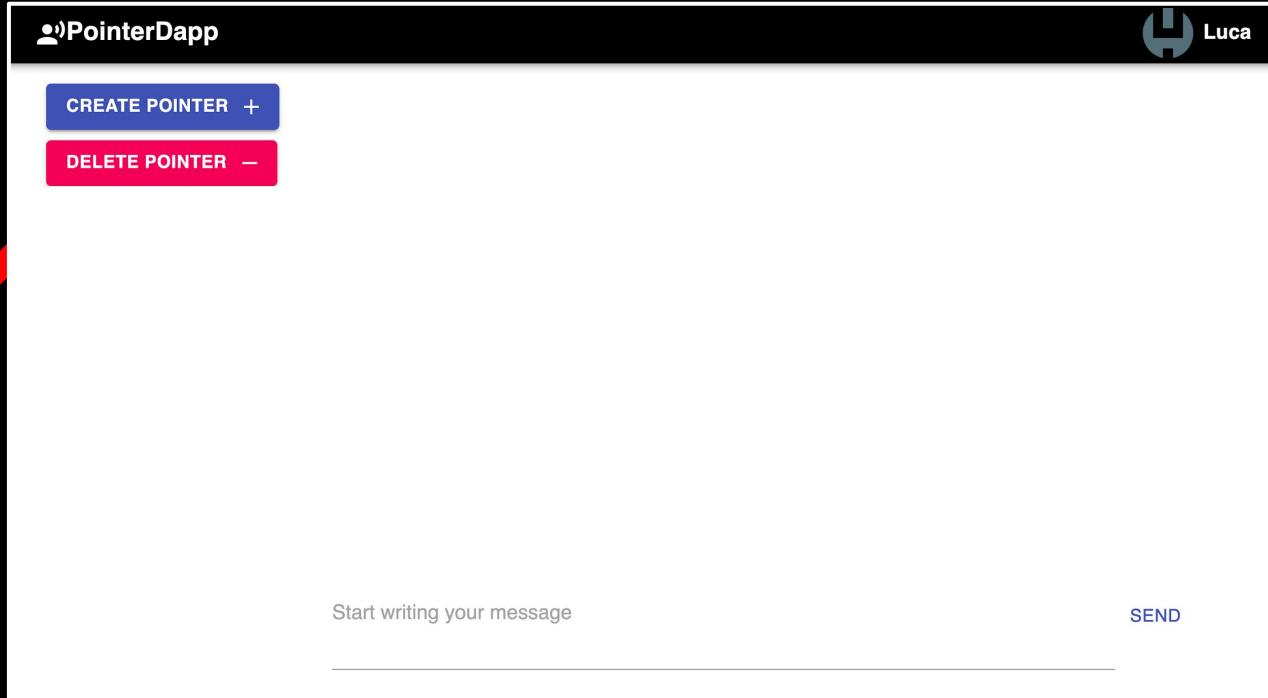
Configure the frontend

To interact with your contract from the **frontend** you need two informations:

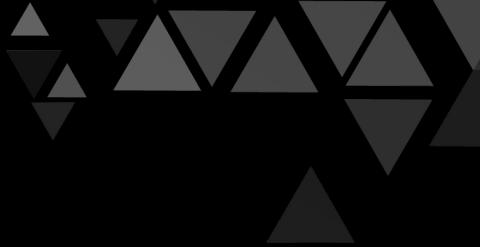
- **The address of your contract**
- **The ABI of your contract**

```
web3.current = new Web3(new Web3.providers.WebsocketProvider('ws://localhost:8545'))
accounts = await web3.current.eth.getAccounts()
contratto = new web3.current.eth.Contract(
    TIME_MACHINE_ABI,
    TIME_MACHINE_ADDRESS
)
.....
[
```

Let's make some calls



Get Data from the Blockchain



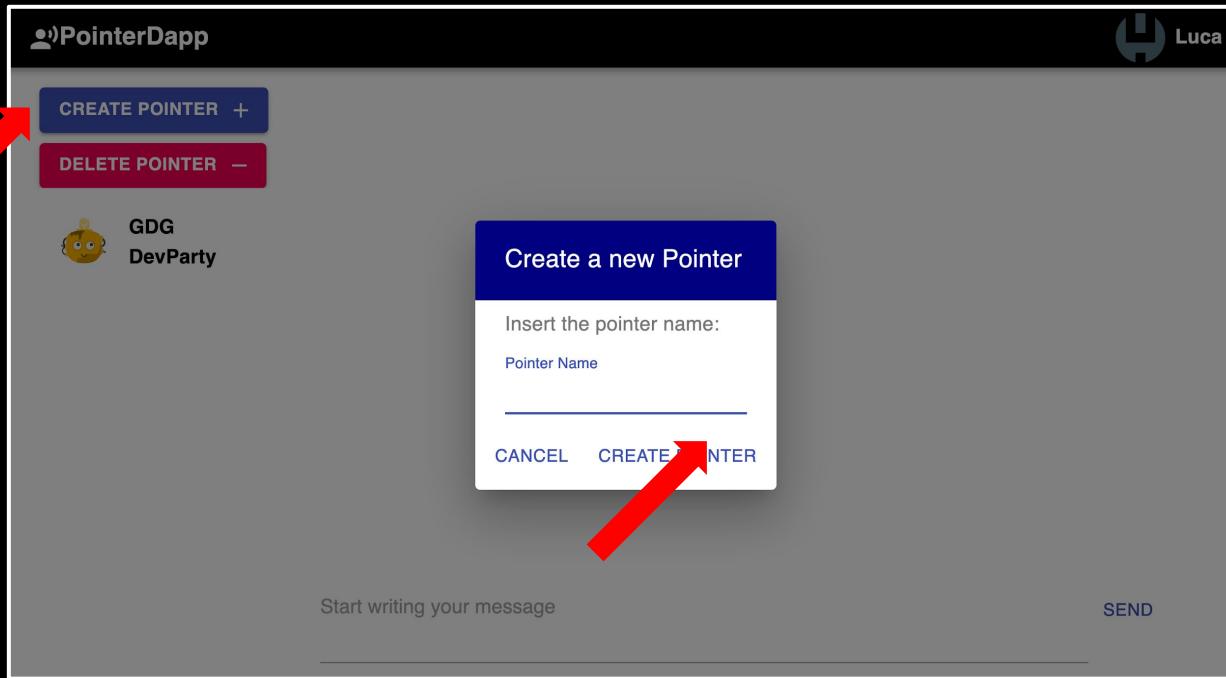
How to invoke a call to the contract?

- We use the function “**call**”
- This is a **free interaction** with the contract



```
contratto.current.methods  
    .getGroups( )  
    .call({ from: address })  
    .then((result) => {  
        // Do something  
    });
```

Let's make some calls



Send data to the Blockchain

How to send a transaction to the contract?

- We use the function “**send**”
- This is a **paid interaction** with the contract.
- We use the function `createGroup` and we could specify:
 - **Gas**
 - **GasPrice**



```
contratto.current.methods
  .createGroup(web3.current.utils.fromAscii(groupName))
  .send({
    from: address,
    gas: 1000000,
    gasPrice: '1'
  })
  .on("confirmation", (confirmationNumber, receipt) => {
    // Do something
  })
  .on("error", (error) => {
    // Do something else
  });
}
```

Let's make some calls

The screenshot shows a user interface for a decentralized application called "PointerDapp". At the top left is the app logo, which is a person icon followed by "PointerDapp". At the top right is a profile picture of a person named "Luca". Below the header, there are two main buttons: "CREATE POINTER +", which is blue, and "DELETE POINTER -", which is pink. A red arrow points from the text "DELETE POINTER" to the pink button. Below these buttons is a list of pointers. The first pointer is for "Alessandro", represented by a green cross icon, with the text "Benvenut* al nostro talk! :D". The second pointer is for "GDG DevParty", represented by a yellow bee icon, with the text "GDG DevParty". At the bottom of the screen is a message input field with the placeholder "Start writing your message" and a "SEND" button.

Start writing your message

SEND

Handle Events

How to listen for events's contract?

- We can listen for an event using the name of the event.
- We can also retrieve the past events specifying the “fromBlock” field.



```
groupContract.events.Message({  
    fromBlock: 0  
}, function (error, event) {  
    // Do something  
})
```

Pro/Cons



PRO

- **Pseudo-anonymity**
 - No personal data, only address
- **Transparency**
- **Decentralized**
- **Tampering-Free**
- **User owns his data**
- **Has Legal Value**
 - It's more important than you think

CONS

- **High Response Time**
- **Transactions cost**
- **Scalability issues**

Warning

Blockchain is not applicable in any context.
You have to carefully analyze your problem.

Mantra

"If it fits then I use it"

Don't follow the hype.



Thanks!



github.com/PointerPodcast/PointerDapp

PointerDapp.it



Alessandro Berti



Luca Corbucci



Eugenio Paluello



PointerPodcast.it

