ORAICHAIN x CYTAST

Blockchain Anyone

Buổi 2: Hợp đồng thông minh và Ứng dụng phi tập trung

(Smart contract & Dapps)

TS. Đỗ Bá Lâm

Nội dung

ORAICHAIN CYTAST

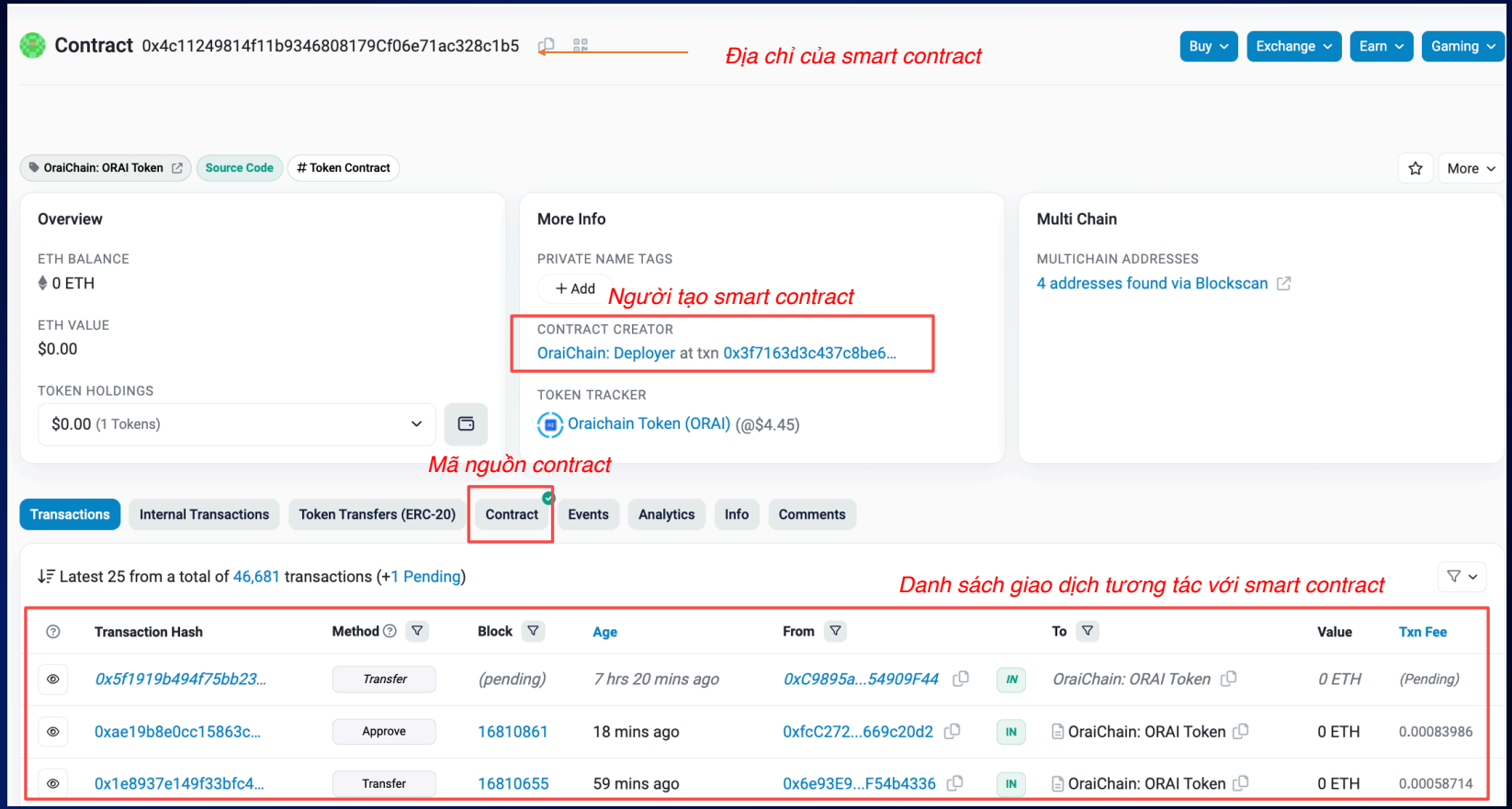
💡 Hợp đồng truyền thống

- Một thỏa thuận hay cam kết giữa hai hoặc nhiều bên để đồng ý thực hiện một điều gì đó
- Các bên tham gia cần có tin tưởng lẫn nhau, hoặc sử dụng một bên trung gian đáng tin cậy hay dựa trên các quy định pháp luật

💡 **Hợp đồng thông minh** là một **chương trình** chạy trên mạng Blockchain, biểu diễn một hợp đồng bằng mã máy và **tự động** thực thi khi các điều khoản được đáp ứng mà không cần các bên tham gia phải tin tưởng nhau hoặc cần một bên trung gian đáng tin cậy nào khác

Hợp đồng truyền thống	Hợp đồng thông minh
Văn bản, lời nói, điện tử	Chương trình máy tính
Cần sự tham gia bên thứ 3 hoặc hệ thống pháp luật	Loại bỏ bên trung gian
Nội dung có thể bị chỉnh sửa	Không thể thay đổi nội dung
Thực thi các thỏa thuận cần thời gian	Thực thi tự động, nhanh chóng khi điều kiện được đáp ứng
Sử dụng định danh hợp pháp của những người liên quan	Sử dụng địa chỉ ví người dùng - cho phép ẩn danh

- Cơ chế hoạt động như một máy bán hàng tự động
  Lựa chọn sản phẩm + Nạp tiền = Sản phẩm được bán
- Hợp đồng thông minh: tự động thực thi các điều khoản khi điều kiện được đáp ứng
  - Có một địa chỉ xác định trên mạng blockchain, là nơi lưu trữ chương trình
  - Là một loại tài khoản trên mạng blockchain: có số dư và cho phép nhận các giao dịch gửi đến



- Thành phần chính của hợp đồng thông minh gồm các biến để lưu dữ liệu và các hàm có thể thực thi để tương tác với dữ liệu đó.

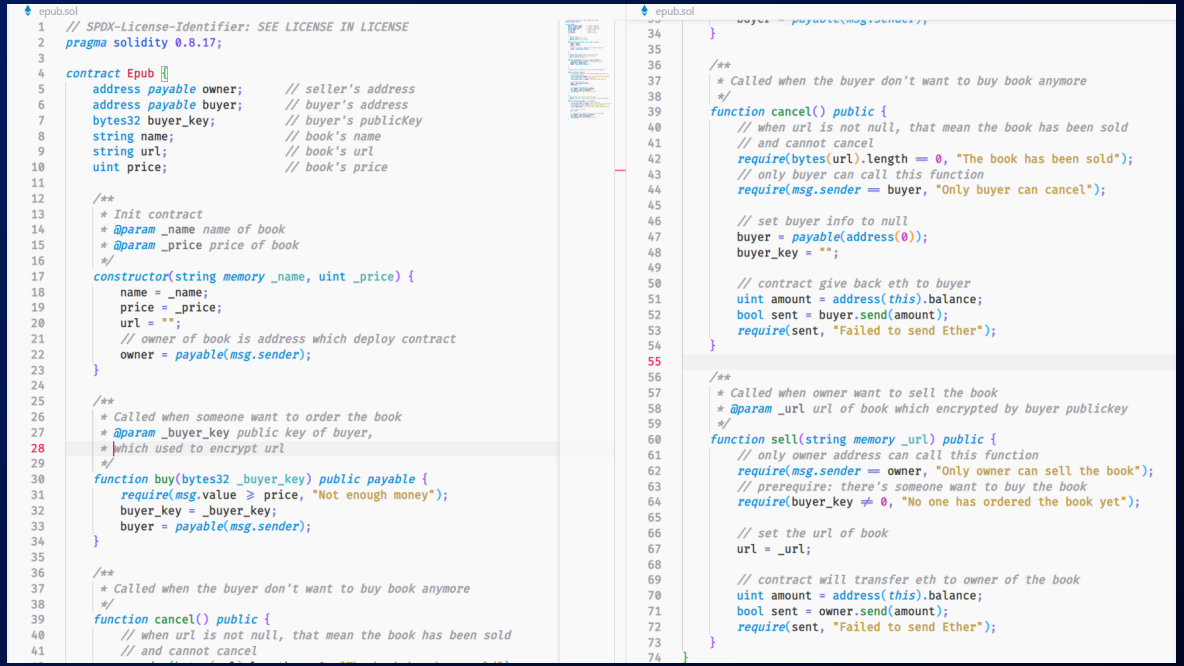
```solidity
// SPDX-License-Identifier: SEE LICENSE IN LICENSE
pragma solidity 0.8.19;

contract ExampleContract {
    string private contract_name; // state variable

    // Called when the contract is deployed and initializes the value
    constructor() public {
        contract_name = "My Example Contract";
    }

    // Get Function
    function read_name() public view returns (string memory) {
        return contract_name;
    }

    // Set Function
    function update_name(string memory new_name) public {
        contract_name = new_name;
    }
}

```

```java
// Java

class ExampleContract {
    private String contractName;

    public ExampleContract() {
        this.contractName = "My Example Contract";
    }

    public String readName() {
        return this.contractName;
    }

    public void updateName(String newName) {
        this.contractName = newName;
    }
}

```

Hợp đồng thông minh còn có một số các biến và hàm đặc biệt để cung cấp các thông tin về blockchain.

VD: ngôn ngữ Solidity chạy trên EVM có các biến như:

- msg.sender: địa chỉ gọi đến hàm của contract
- msg.value: số tiền được gửi đến contract
- x.balance: số dư của địa chỉ x
- x.send(a): gửi a wei đến địa chỉ x
- ...



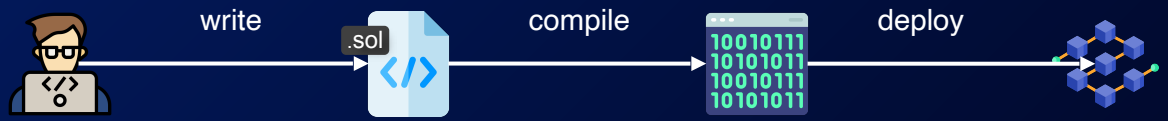
epub.sol

```sol
// SPDX-License-Identifier: SEE LICENSE IN LICENSE
pragma solidity 0.8.17;

contract Epub {
    address payable owner;      // seller's address
    address payable buyer;      // buyer's address
    bytes32 buyer_key;          // buyer's publicKey
    string name;                // book's name
    string url;                 // book's url
    uint price;                 // book's price

    /**
     * Init contract
     * @param _name name of book
     * @param _price price of book
     */
    constructor(string memory _name, uint _price) {
        name = _name;
        price = _price;
        url = "";
        // owner of book is address which deploy contract
        owner = payable(msg.sender);
    }

    /**
     * Called when someone want to order the book
     * @param _buyer_key public key of buyer,
     * which used to encrypt url
     */
    function buy(bytes32 _buyer_key) public payable {
        require(msg.value >= price, "Not enough money");
        buyer_key = _buyer_key;
        buyer = payable(msg.sender);
    }

    /**
     * Called when the buyer don't want to buy book anymore
     */
    function cancel() public {
        // when url is not null, that mean the book has been sold
        // and cannot cancel
```

epub.sol

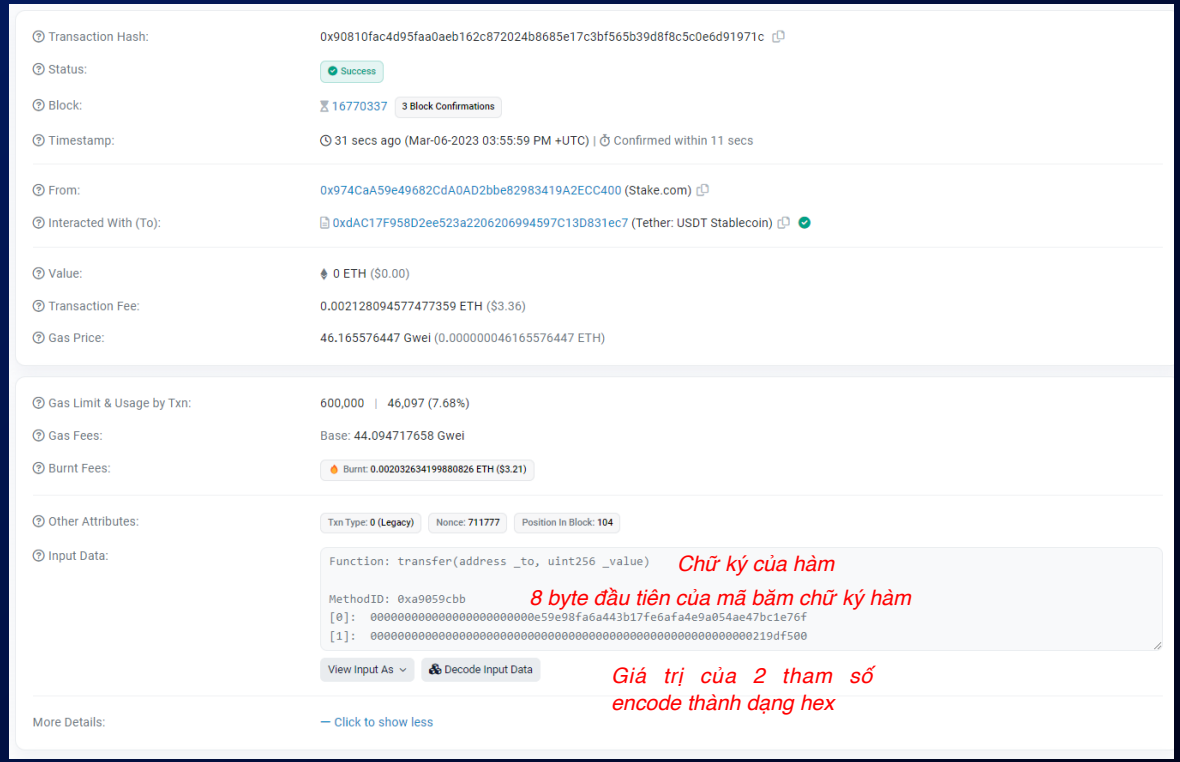
```sol
    }

    /**
     * Called when the buyer don't want to buy book anymore
     */
    function cancel() public {
        // when url is not null, that mean the book has been sold
        // and cannot cancel
        require(bytes(url).length == 0, "The book has been sold");
        // only buyer can call this function
        require(msg.sender == buyer, "Only buyer can cancel");

        // set buyer info to null
        buyer = payable(address(0));
        buyer_key = "";

        // contract give back eth to buyer
        uint amount = address(this).balance;
        bool sent = buyer.send(amount);
        require(sent, "Failed to send Ether");
    }

    /**
     * Called when owner want to sell the book
     * @param _url url of book which encrypted by buyer publickey
     */
    function sell(string memory _url) public {
        // only owner address can call this function
        require(msg.sender == owner, "Only owner can sell the book");
        // prerequire: there's someone want to buy the book
        require(buyer_key != 0, "No one has ordered the book yet");

        // set the url of book
        url = _url;

        // contract will transfer eth to owner of the book
        uint amount = address(this).balance;
        bool sent = owner.send(amount);
        require(sent, "Failed to send Ether");
    }
}
```

- Nhà phát triển cần định nghĩa các logic của hợp đồng thông qua việc viết mã bằng các ngôn ngữ lập trình
    - Ethereum: Solidity, Vyper
    - Cosmos: Rust
    - Cardano: Haskell
- Sau đó biên dịch thành mã máy bằng các phần mềm tương ứng
- Thực hiện một giao dịch để đẩy mã máy lên Blockchain



Trên Ethereum, deploy contract là việc một tài khoản tạo một giao dịch trên mạng nhưng không có địa chỉ đến (chỉ có "**from**" là địa chỉ người deploy nhưng không có "**to**") kèm theo dữ liệu **input data** là nội dung contract và tham số khởi tạo (được compile sang bytecode), giao dịch này trả về **result** là **địa chỉ của contract mới được deploy.**

- Người dùng có thể tương tác với contract bằng cách gọi các hàm được khai báo trong contract đó. Việc này cũng được thực hiện bằng cách tạo một giao dịch với địa chỉ đến là địa chỉ contract và input data là chữ ký của hàm và các tham số của hàm đó.



💡Việc tạo giao dịch để chuyển tiền bình thường hay để deploy và tương tác với contract đều có thể thực hiện bằng các chương trình Blockchain client (Ethereum có Geth, Besu; Tezos có Octez, Cardano có Cardano-node…) hoặc thông qua các thư viện web3 của các ngôn ngữ lập trình phổ biến (JS, Python, Java…)

ORAICHAIN CYTAST

Lợi ích

TỰ ĐỘNG THỰC THI

Lợi ích quan trọng nhất của hợp đồng thông minh khi so với hợp đồng thường đó là được tự động thực thi khi các điều kiện đã được thỏa mãn.

KẾT QUẢ RÕ RÀNG

Hợp đồng truyền thống có thể gây ra một số tranh cãi bởi những cách lý giải câu từ khác nhau. Hợp đồng thông minh được thực thi chính xác dựa trên các điều kiện được viết trong mã.

CÔNG KHAI & MINH BẠCH

Hợp đồng thông minh được lưu trên mạng blockchain có nghĩa là ai cũng có thể xem được nội dung hợp đồng và các giao dịch liên quan, do đó rất hữu ích trong việc kiểm toán, theo dõi và kiểm tra hợp đồng.

BẢO VỆ QUYỀN RIÊNG TƯ

Được thừa hưởng đặc tính ẩn danh của blockchain, các tài khoản có thể giao dịch và tương tác với hợp đồng mà không sợ bị lộ danh tính.

Hạn chế

Còn một số hạn chế khác an toàn bảo mật, sự chấp nhận của pháp luật

Giới hạn kích thước

Các contract thường bị giới hạn kích thước do đó các chức năng cũng bị giới hạn.

Thụ động

Các contract được thiết kế để không thể kết nối và tự lấy thông tin từ thế giới bên ngoài vì những thông tin đó có thể phá vỡ sự đồng thuận.

ORAICHAIN CYTAST

TOKEN

Token là một hợp đồng thông minh lưu số dư của một địa chỉ và một hàm để chuyển tiền (tăng giảm số dư của 2 địa chỉ giao dịch), cùng với các thông tin và một số chức năng khác.

NON-FUNGIBLE TOKEN

Mỗi NFT là một hợp đồng thông minh chứa thông tin về một tài sản (có thể là hiện vật hay kỹ thuật số) và địa chỉ chủ sở hữu của nó, để thể hiện quyền sở hữu đối với các vật phẩm duy nhất.

LƯU TRỮ PHI TẬP TRUNG

Hợp đồng thông minh có thể được sử dụng để lưu trữ các bản ghi như một cơ sở dữ liệu phi tập trung, mọi thao tác cập nhật đều được blockchain lưu lại qua các giao dịch và có thể truy vết.

BỎ PHIẾU ĐIỆN TỬ

Bỏ phiếu từ xa thông qua hợp đồng thông minh tiết kiệm được chi phí, trong khi đảm bảo được tính an toàn và minh bạch do các phiếu bầu được lưu trữ bất biến và có ghi dấu thời gian, đồng thời danh tính của cử tri được giữ bí mật nhờ tính ẩn danh của blockchain.

Ví dụ về Token Smart contract

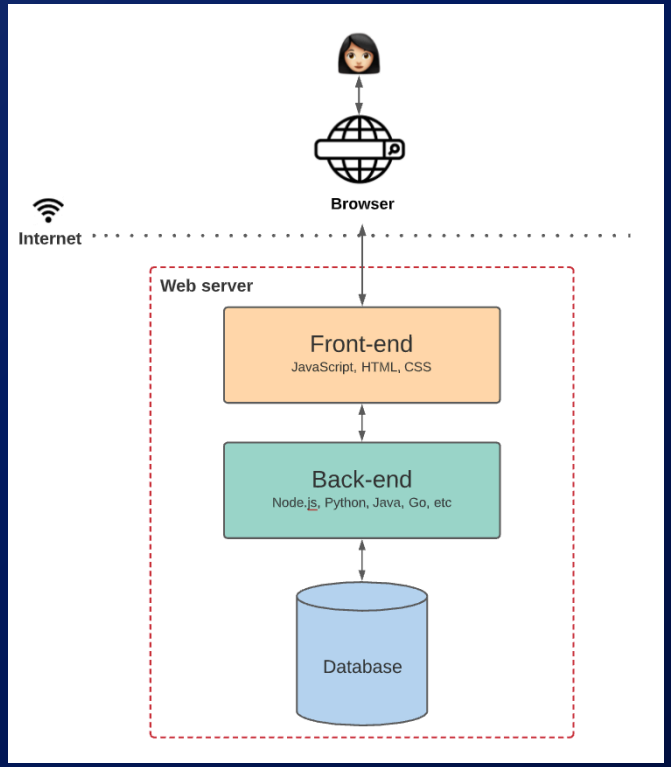
ORAICHAIN CYTAST

```
1 contract ExampleCoin {
2   mapping(address → uint) public balances;
3
4   function transfer(address to, uint amount) public {
5     // ... validate balances and other things
6     balances[msg.sender] -= amount;
7     balances[to] += amount;
8   }
9 }
```

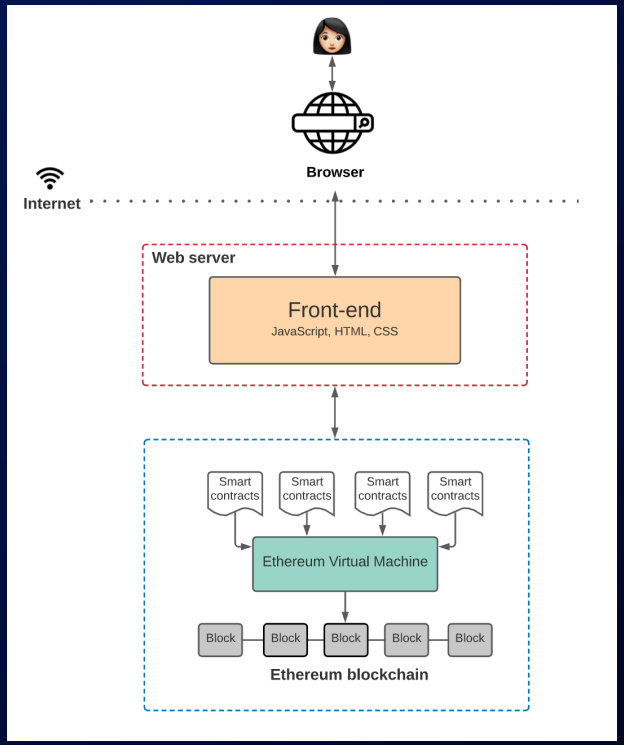
Định nghĩa

Ứng dụng phi tập trung (Decentralized Applications – dApps) là một ứng dụng mà phần code backend của nó chạy trên một mạng ngang hàng phi tập trung, thay vì chạy trên một server trung tâm được quản lý bởi một thực thể cụ thể. Còn phần giao diện người dùng của ứng dụng vẫn có thể là viết bằng bất kỳ ngôn ngữ frontend thông thường nào.

Để chạy một ứng dụng với phần code backend là phi tập trung, cách tốt nhất đó là sử dụng các hợp đồng thông minh cho phần backend, vì chúng được lưu trữ và thực thi thông qua các giao dịch trên mạng ngang hàng blockchain.



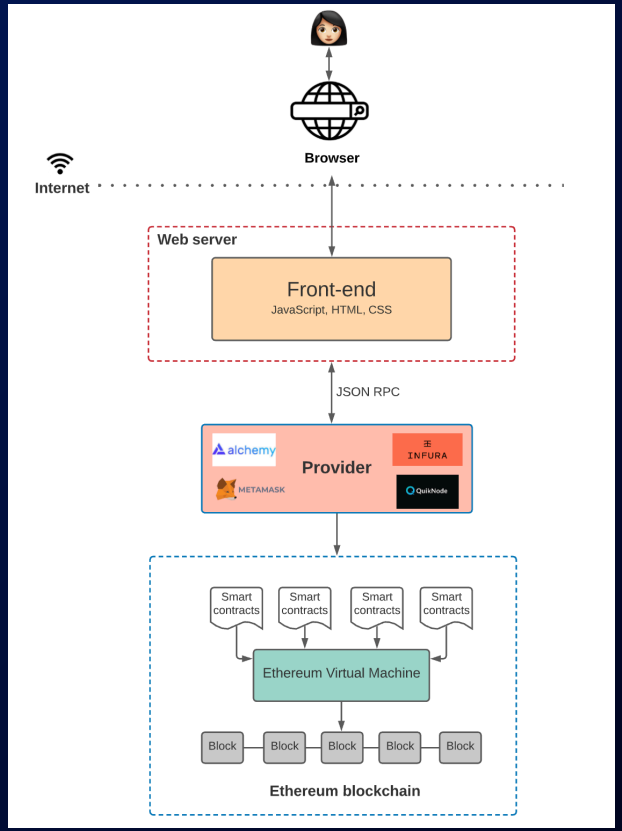
Kiến trúc ứng dụng web truyền thống



Kiến trúc ứng dụng Dapp

Giao tiếp giữa Front-end và Smart contract

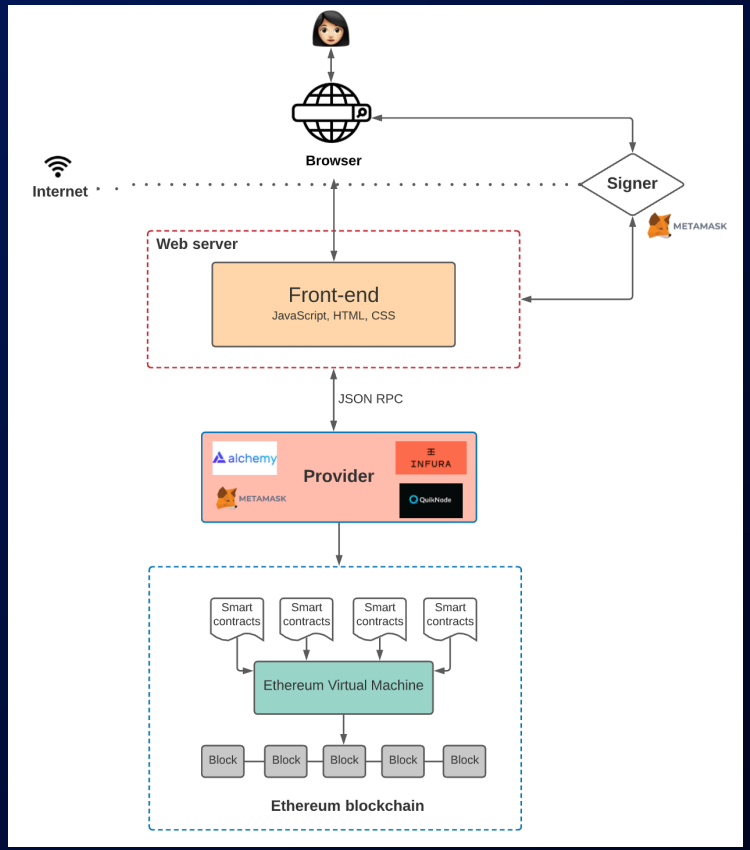
- Thiết lập một node trên mạng
- Sử dụng các bên trung gian như Infura, Alchemy, Quicknode
- Client giao tiếp với nút mạng blockchain thông qua chuẩn JSON-RPC (Cosmos hỗ trợ thêm chuẩn REST API)



Kiến trúc ứng dụng Dapp

Giao tiếp giữa người dùng và Front-end

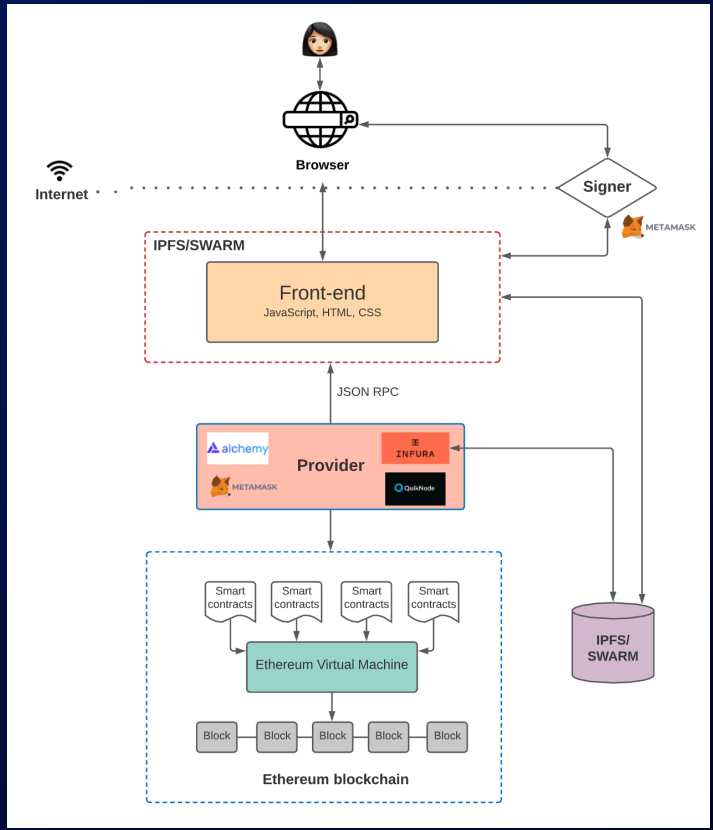
- Để ghi dữ liệu vào Dapp, người dùng cần ký (sign) một giao dịch
- Sử dụng khóa riêng tư (private key) được lưu trong ví Meta Mask



Kiến trúc ứng dụng Dapp

Lưu trữ dữ liệu trên mạng blockchain

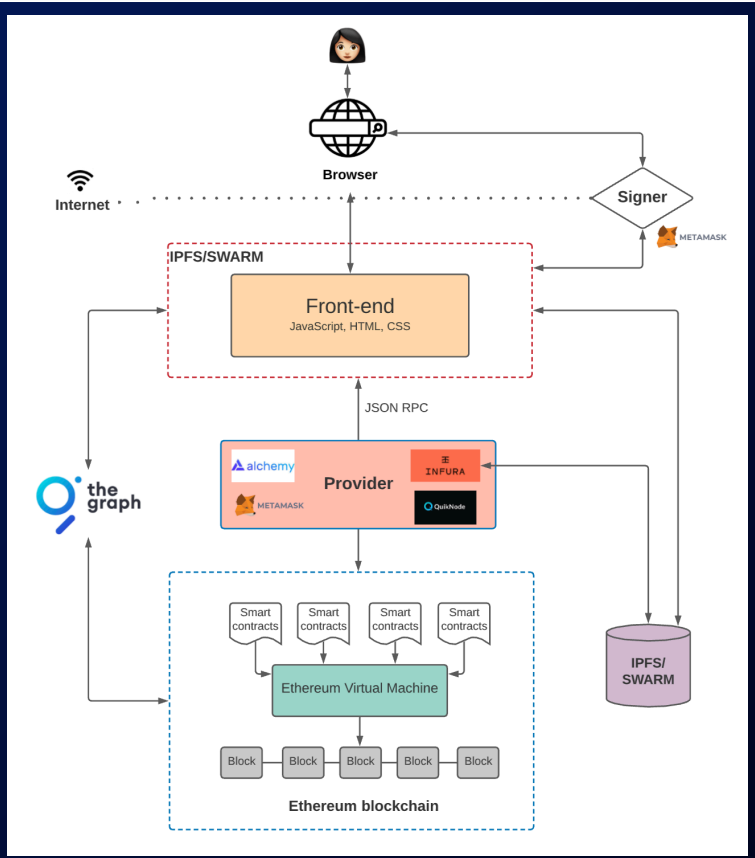
- Sử dụng IPFS, SWARM - hệ thống file phi tập trung để lưu trữ dữ liệu
- Mã băm của dữ liệu sẽ được lưu trên mạng blockchain
- Front-end cũng có thể lưu trên IPFS/SWARM thay vì web server



Kiến trúc ứng dụng Dapp

Truy vấn dữ liệu

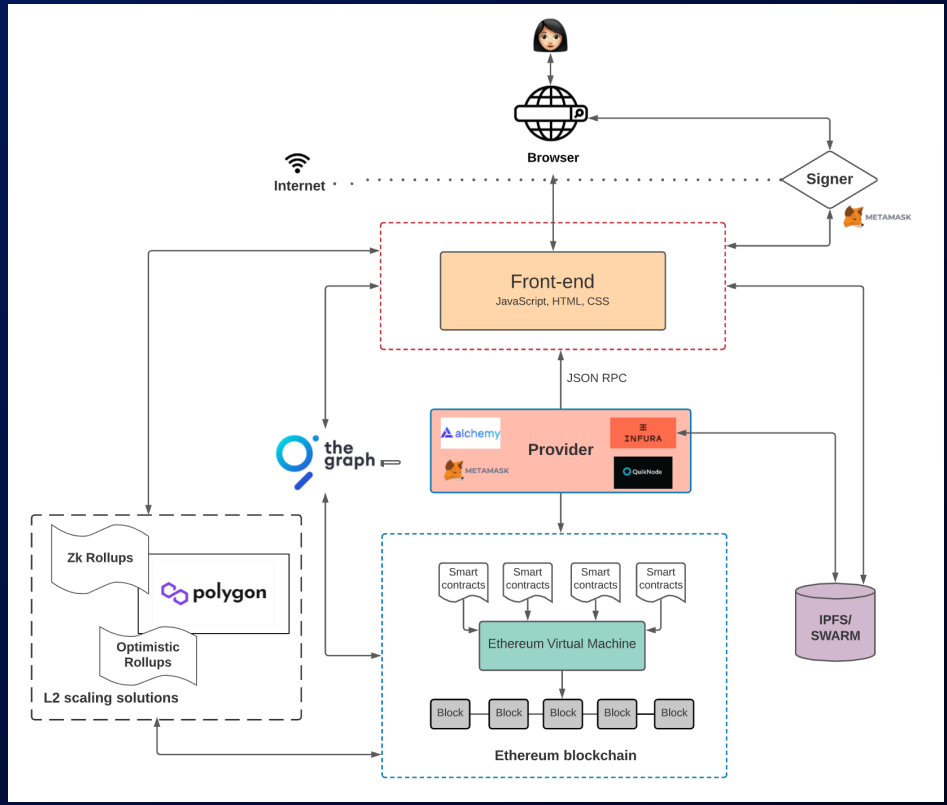
- Lắng nghe các sự kiện - event trong hợp đồng thông minh
- Sử dụng The Graph - giải pháp đánh chỉ mục dữ liệu, cho phép định nghĩa sự kiện, hàm cần lắng nghe



Kiến trúc ứng dụng Dapp

Mở rộng mạng

- Layer 2 đề cập tới các giải pháp thực thi giao dịch ngoài chuỗi chính, nhằm tăng tốc độ xử lý và giảm chi phí giao dịch
- Sidechains: một mạng blockchain có cơ chế đồng thuật, tham số khối riêng. Chúng được kết nối với Layer 1 qua cầu nối (bridge) - hợp đồng thông minh triển khai trên cả hai mạng
- Rollups: thực thi các giao dịch ngoài chuỗi chính, sau đó tổng hợp lại bằng chứng để ghi lên chuỗi chính



Kiến trúc ứng dụng Dapp

ORAICHAIN CYTAST

Lợi ích

- **Luôn hoạt động** – Sau khi hợp đồng thông minh được deploy, toàn bộ mạng đều có thể phục vụ các client muốn tương tác với hợp đồng, do đó không bị tấn công từ chối dịch vụ vào dapp đơn lẻ.
- **Riêng tư** – Không cần cung cấp danh tính trong thế giới thực để triển khai hoặc tương tác với các dapp, nhờ vào đặc tính ẩn danh của blockchain.
- **Chống kiểm duyệt** – Mạng ngang hàng phi tập trung giúp người dùng có thể tự do tương tác mà không lo bị chặn bởi bất cứ thực thể đơn lẻ nào.
- **Toàn vẹn dữ liệu** – Các tương tác với hợp đồng thông minh từ dapp được ghi vào trong các giao dịch được lưu vĩnh viễn trên blockchain nên không thể bị sửa đổi tự do từ các tác nhân độc hại.
- **Không cần bên thứ 3 đáng tin** – Hợp đồng thông minh tự động thực thi theo các logic được đề ra khiến kết quả luôn có thể dự đoán trước được, thay vì mô hình truyền thống cần phải tin tưởng vào bên cung cấp dịch vụ (vd phải tin vào các hệ thống ngân hàng không thay đổi dữ liệu của mình).

ORAICHAIN CYTAST

Hạn chế

- **Khó bảo trì** – Hợp đồng thông minh đã được triển khai trên mạng blockchain không thể sửa đổi, do đó khó để bảo trì, kể cả khi phát hiện các lỗi bảo mật trong những phiên bản cũ.

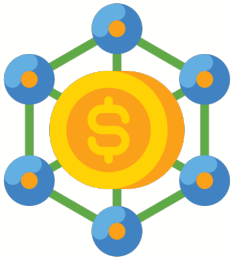
- **Chi phí** – Vấn đề chung và lớn nhất của các mạng blockchain, dapp càng được sử dụng nhiều thì càng sinh ra nhiều giao dịch phải lưu trên toàn bộ các nút mạng khiến tiêu tốn nhiều chi phí cho việc lưu trữ, xác minh dữ liệu và tốn nhiều lưu lượng mạng dẫn đến tình trạng tắc nghẽn.

- **Trải nghiệm người dùng** – Người dùng cuối sẽ gặp khó khăn để thiết lập các công cụ cần thiết để tương tác với mạng blockchain một cách an toàn (tải ví, sinh và quản lý cặp khóa….).

Game phi tập trung

Sử dụng các NFT đại diện cho các item duy nhất trong game, người chơi sẽ sở hữu các vật phẩm độc nhất vô nhị của mình mà không lo bị nhà phát hành thao túng.

Tài chính phi tập trung



Các hoạt động tài chính như trao đổi, vay mượn tiền tệ có thể thực hiện nhanh chóng và chính xác trên các hợp đồng thông minh thay vì cần các thủ tục phức tạp như ngân hang truyền thống.

Tổ chức tự trị phi tập trung

Các quy tắc của tổ chức xây dựng dưới dạng các quy tắc được biểu diễn bởi các hợp đồng thông minh được kiểm soát bởi các thành viên của tổ chức có quyền hạn ngang nhau.

TS. Đỗ Bá Lâm

Giảng viên Trường Công nghệ thông tin và truyền thông, Đại học Bách khoa Hà Nội

XIN CẢM ƠN!

Tài liệu tham khảo

1. https://ethereum.org/en/developers/docs/smart-contracts/
2. https://docs.soliditylang.org/en/v0.8.17/introduction-to-smart-contracts.html
3. https://www.preethikasireddy.com/post/the-architecture-of-a-web-3-0-application

Phụ lục

- Các thành phần:
  - Biến trạng thái (state variable)
  - Hàm (function)
  - Kiểm tra điều kiện hàm (function modifier)
  - Sự kiện (event)
  - Lỗi (error)
  - Cấu trúc (struct)
  - Kiểu dữ liệu tự định nghĩa enum types)

- Các thành phần:
  - Biến trạng thái (state variable)
  - Hàm (function)
  - Kiểm tra điều kiện hàm (function modifier)
  - Sự kiện (event)
  - Lỗi (error)
  - Cấu trúc (struct)
  - Kiểu dữ liệu tự định nghĩa enum types)

```
// SPDX-License-Identifier: GPL-3.0
pragma solidity >=0.4.0 <0.9.0;

contract SimpleStorage {
    uint storedData; // State variable
    // ...
}
```

- Các thành phần:
  - Biến trạng thái (state variable)
  - Hàm (function)
  - Kiểm tra điều kiện hàm (function modifier)
  - Sự kiện (event)
  - Lỗi (error)
  - Cấu trúc (struct)
  - Kiểu dữ liệu tự định nghĩa enum types)

```
// SPDX-License-Identifier: GPL-3.0
pragma solidity >=0.7.1 <0.9.0;

contract SimpleAuction {
    function bid() public payable { // Function
        // ...
    }
}

// Helper function defined outside of a contract
function helper(uint x) pure returns (uint) {
    return x * 2;
}
```

- Các thành phần:
  - Biến trạng thái (state variable)
  - Hàm (function)
  - Kiểm tra điều kiện hàm (function modifier)
  - Sự kiện (event)
  - Lỗi (error)
  - Cấu trúc (struct)
  - Kiểu dữ liệu tự định nghĩa enum types)

```solidity
// SPDX-License-Identifier: GPL-3.0
pragma solidity >=0.4.22 <0.9.0;

contract Purchase {
    address public seller;

    modifier onlySeller() { // Modifier
        require(
            msg.sender == seller,
            "Only seller can call this."
        );
        _;
    }

    function abort() public view onlySeller { // Modifier usage
        // ...
    }
}
```

- Các thành phần:
  - Biến trạng thái (state variable)
  - Hàm (function)
  - Kiểm tra điều kiện hàm (function modifier)
  - Sự kiện (event)
  - Lỗi (error)
  - Cấu trúc (struct)
  - Kiểu dữ liệu tự định nghĩa enum types)

```
// SPDX-License-Identifier: GPL-3.0
pragma solidity >=0.4.21 <0.9.0;

contract SimpleAuction {
    event HighestBidIncreased(address bidder, uint amount); // Event

    function bid() public payable {
        // ...
        emit HighestBidIncreased(msg.sender, msg.value); // Triggering event
    }
}
```

- Các thành phần:
  - Biến trạng thái (state variable)
  - Hàm (function)
  - Kiểm tra điều kiện hàm (function modifier)
  - Sự kiện (event)
  - Lỗi (error)
  - Cấu trúc (struct)
  - Kiểu dữ liệu tự định nghĩa enum types)

```solidity
// SPDX-License-Identifier: GPL-3.0
pragma solidity ^0.8.4;

/// Not enough funds for transfer. Requested `requested`,
/// but only `available` available.
error NotEnoughFunds(uint requested, uint available);

contract Token {
    mapping(address => uint) balances;
    function transfer(address to, uint amount) public {
        uint balance = balances[msg.sender];
        if (balance < amount)
            revert NotEnoughFunds(amount, balance);
        balances[msg.sender] -= amount;
        balances[to] += amount;
        // ...
    }
}
```

- Các thành phần:
  - Biến trạng thái (state variable)
  - Hàm (function)
  - Kiểm tra điều kiện hàm (function modifier)
  - Sự kiện (event)
  - Lỗi (error)
  - Cấu trúc (struct)
  - Kiểu dữ liệu tự định nghĩa enum types)

```
// SPDX-License-Identifier: GPL-3.0
pragma solidity >=0.4.0 <0.9.0;

contract Ballot {
    struct Voter { // Struct
        uint weight;
        bool voted;
        address delegate;
        uint vote;
    }
}
```

- Các thành phần:
  - Biến trạng thái (state variable)
  - Hàm (function)
  - Kiểm tra điều kiện hàm (function modifier)
  - Sự kiện (event)
  - Lỗi (error)
  - Cấu trúc (struct)
  - Kiểu dữ liệu tự định nghĩa enum types)

```
// SPDX-License-Identifier: GPL-3.0
pragma solidity >=0.4.0 <0.9.0;

contract Purchase {
    enum State { Created, Locked, Inactive } // Enum
}
```