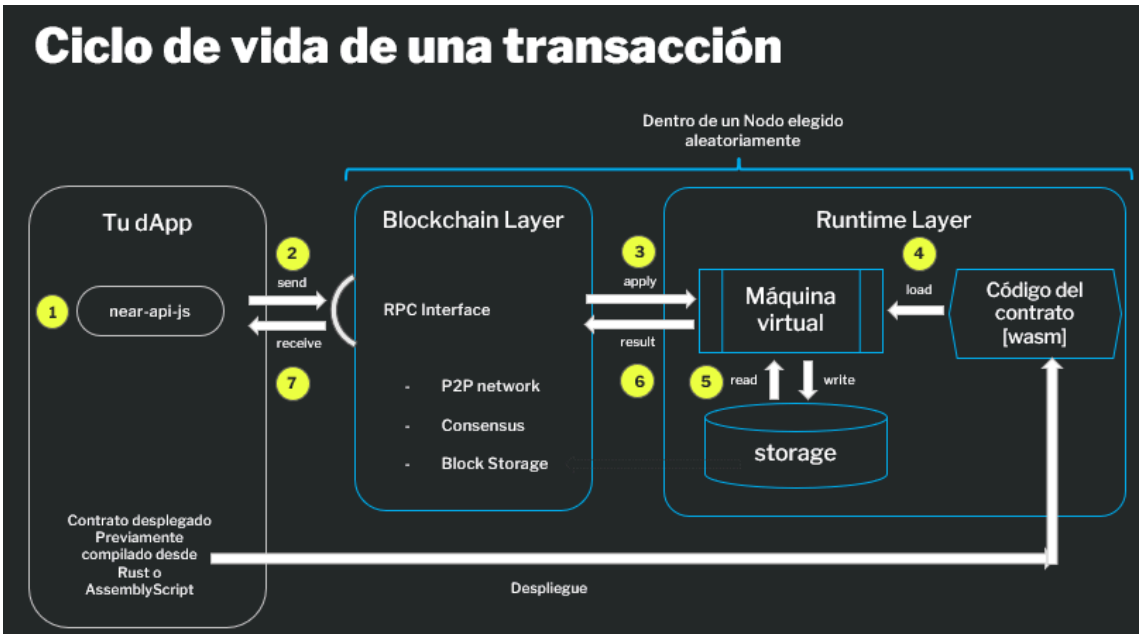
1. Ciclo de vida de una transacción.



Invocar un método de tu DApp sigue aproximadamente los siguientes pasos:

1. Tu dApp usa near-api-js para crear y firmar la transacción.
2. La transacción es enviada a la plataforma de NEAR a través de la interface RPC que valida y verifica la transacción antes de enviarla, basada en la cuenta del contrato, al shard correcto (cada shard es mantenido por al menos un nodo físico en la red).
3. La capa de ejecución (runtime layer) **despierta** una máquina virtual wasm.
4. La máquina virtual carga el código del contrato para invocar la función especificada en la transacción.
5. Se lee o escribe en el storage los datos necesarios.
6. Devuelve el resultado de la ejecución del método y se apaga la máquina virtual.
7. La capa blockchain (blockchain layer) devuelve el resultado a través de la interface RPC a tu DApp.

Definiciones útiles:

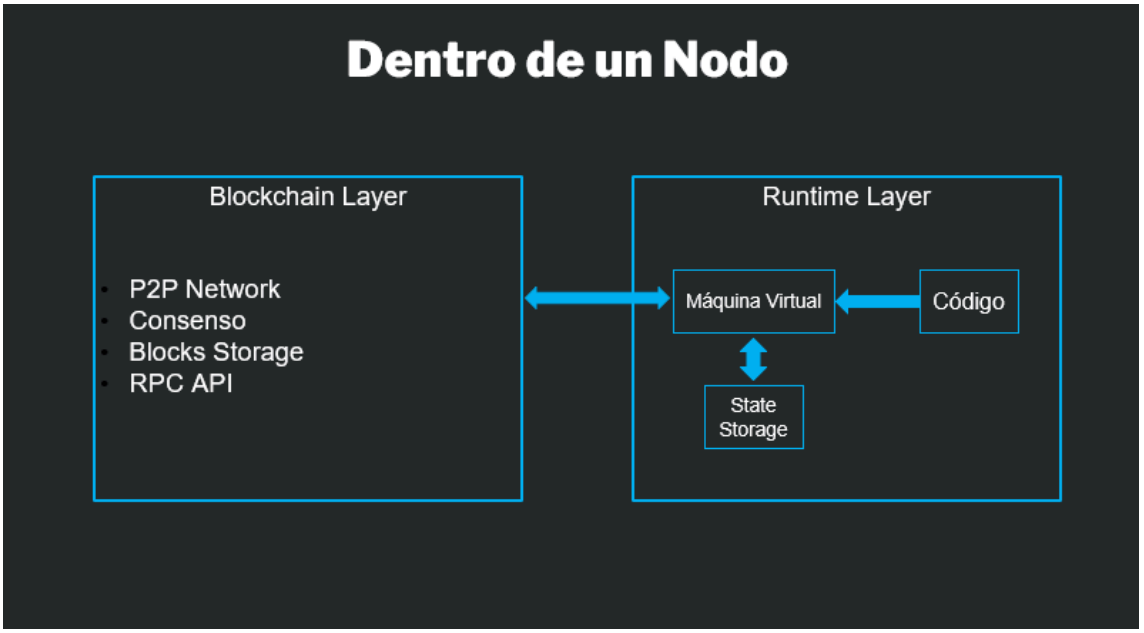
Interface RPC: RPC significa llamada a procedimiento remoto (Remote Procedure Call). Es un protocolo de comunicación utilizado para solicitar un servicio de un programa ubicado en otra computadora o red sin tener que comprender los detalles de la red.

Es una abstracción útil a desarrolladores y usuarios que hace fácil comunicarse con la blockchain a pesar de su complejidad.

Shard: Sharding es una técnica aplicada a ciertas cadenas de bloques o blockchain. Consiste en dividir una red en partes denominadas shards. Esto facilita la escalabilidad y la mayor velocidad de procesamiento de operaciones.

Etimología: Shard se remonta al inglés antiguo (donde se deletreaba sceard) y está relacionado con la palabra scieran, que significa "cortar".

Runtime layer vs Blockchain layer:



Un nodo de near consta de manera simplificada de una blockchain layer (capa de cadena de bloques) y una runtime layer (capa de tiempo de ejecución).

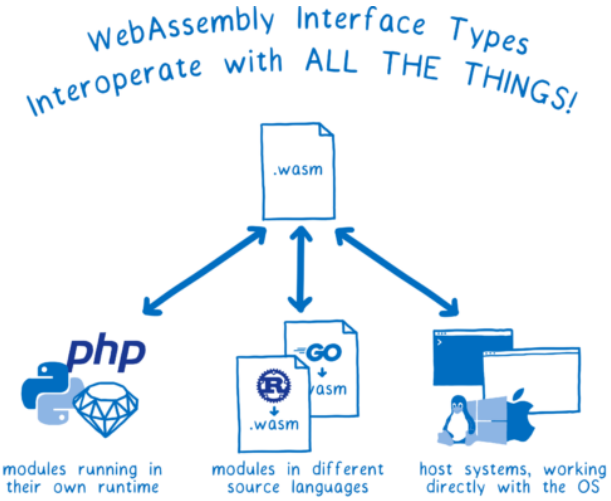
La capa blockchain lleva adelante tareas relacionadas con mantener la cadena de bloques de datos.

La capa de ejecución se encarga de las tareas relacionadas a la ejecución de los métodos en contratos inteligentes de la red.

Máquina virtual:



Una máquina virtual es como tener un ordenador dentro de otro. No de forma física pero sí emulado dentro de una ventana, como si fuera un programa. Cuando el usuario pone en marcha una máquina virtual, arranca un programa que se comporta como si fuera un ordenador independiente dentro de su equipo.
Wasm:



Wasm es la abreviatura de WebAssembly. Es un formato de instrucción para una máquina virtual. Es utilizado como lenguaje al que se compilan los contratos inteligentes porque alcanza un buen desempeño, el archivo wasm es pequeño, tiene un ecosistema de desarrollo sólido, entre otras razones.

Detalle a resaltar sobre el ciclo de vida de una transacción:

- Las Máquinas Virtuales despiertan solo al ejecutarse una transacción y se levanta en el nodo asignado. Por ello los contratos inteligentes son reactivos.

  Esta característica de un contrato inteligente es similar a una función lambda de AWS.



Recursos adicionales:

RPC:

https://www.ionos.es/digitalguide/servidores/know-how/que-es-rpc/#:~:text=La%20RPC%20o%20Remote%20Procedure,redes%20y%20arquitecturas%20cliente%2Dservidor.

https://morioh.com/p/a930f99d5dfd

https://docs.near.org/docs/api/rpc

Shard:

https://economipedia.com/definiciones/sharding.html#:~:text=Sharding%20es%20una%20t%C3%A9cnica%20aplicada,velocidad%20de%20procesamiento%20de%20operaciones.

https://www.merriam-webster.com/dictionary/shard

Runtime Layer vs Blockchain layer:

https://nomicon.io/RuntimeSpec/Runtime#:~:text=Runtime%20layer%20is%20used%20to,the%20state%20between%20the%20executions.

https://pt.slideshare.net/maksym_zavershynskyi/rust-smart-contracts

Máquina virtual:

https://www.consumer.es/tecnologia/software/maquinas-virtuales-que-son-y-para-que-sirven.html

WebAssembly:

https://webassembly.org/

https://ink.substrate.io/why-webassembly-for-smart-contracts/

2. Lenguajes de programación Rust vs AssemblyScript.

Los contratos inteligentes en NEAR son escritos en Rust o AssemblyScript antes de ser compilados a WebAssembly. El código legible para nosotros (en Rust o AssemblyScript) no puede ser ejecutado directamente, necesita ser compilado a código máquina.

El siguiente cuadro es una comparación entre AssemblyScript y Rust.

AssemblyScript vs Rust		
	AssemblyScript	Rust
Pro	- Fácil uso para prototipar - Fácil de aprender para desarrollares de JS y TS. - Binaries en Wasm más pequeños. - Binaries son más fáciles de leer/desbugear.	- Compilador maduro y probado en batalla. - Próspero ecosistema. - Casos de uso reales. - El near-sdk-rs te hace la vida fácil.
Contras	- Compilador y ecosistema inmaduros. - Herramientas de desbugeo aun inmaduras.	- Empinada curva de aprendizaje. - Incluso expertos batallan con el compilador.

Rust es preferido sobre AssemblyScript. La razón principal para esto es que el proceso de compilación de Rust hacia wasm tiene herramientas más maduras, lo que garantiza mayor seguridad en el desarrollo de aplicaciones que transacciona valores.

Aunque AssemblyScript es más fácil de aprender. Iniciarse con AS en el desarrollo de contratos inteligentes permite enfocarse en comprender cómo funciona el protocolo NEAR en lugar de pasar la mayor parte del tiempo entendiendo la sintaxis de Rust.

Recursos Adicionales:

https://blog.suborbital.dev/assemblyscript-vs-rust-for-your-wasm-app

3. ¿Qué es un Contrato Inteligente?



"Si me das $2.50 y presionas este botón, obtendrás una Coca-Cola Light."

La mejor metáfora para un contrato inteligente es una máquina expendedora: con los inputs correctos, se garantiza un resultado.

Para obtener un snack de una máquina expendedora:

$$dinero \; + \; elección\ de\ un\ snack \; = \; snack\ entregado$$

Listemos algunas de las características de una máquina expendedora:

- Es neutral.
- Se ejecuta automáticamente (automático pero reactivo).
- No es tan inteligente: hace lo que se le dice (dame una coca-cola).
- Cumple con un acuerdo previo (Usted ingresa el dinero. Presione el botón. ¡Listo!).

Todo esto es en realidad un pequeño programa ("contrato") codificado ("escrito") en la máquina de antemano que se ejecuta cuando presionas el botón ("cuando firmas una transacción").

El código de computadora es como un contrato. Los contratos inteligentes son código.

A diferencia de un "contrato" en el sentido legal, tanto humanos como máquinas lo pueden leer.

Al igual que una máquina expendedora elimina la necesidad de un empleado vendedor, los contratos inteligentes pueden reemplazar a los intermediarios en muchas industrias.

Entonces, ¿Si los contratos inteligentes son como máquinas expendedoras, qué los hace especiales?

Piensa en lo siguiente:

> ¿Por qué las máquinas expendedoras están dentro de edificios de oficinas o centros comerciales y no en la vía pública?
>
> ¿Qué pasaría si quisiéramos hacer una máquina expendedora de joyas valuadas en miles de dólares? ¿Confiarías en que te entregará la joya al ingresar el dinero?
>
> ¿Si fuera una máquina de cambio de divisas? ¿preguntarías antes de usarla si la maquina está respaldada por alguna institución?

Las respuestas a estas preguntas giran alrededor del concepto de **Confianza**. Y este es un problema que ya visitamos y resolvimos con anterioridad.

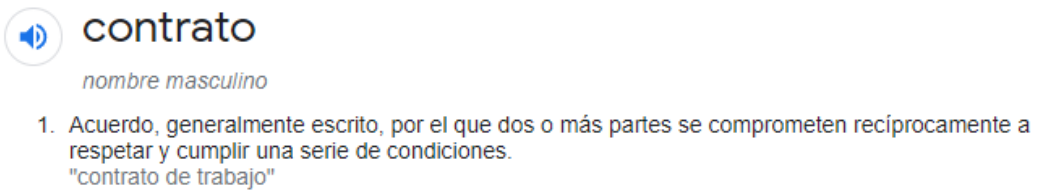
La forma en que funciona la blockchain, asegurando la integridad de los datos, y por lo tanto de sus transacciones hace que puedas confiar en acuerdos escritos a través de un contrato inteligente.

Un contrato inteligente es la combinación de la automatización de los programas de computadora tradicionales y la confianza que da la blockchain.

$$Automatización\ (programa)\ +\ Confianza\ (blockhain)\ =\ Contrato\ Inteligente$$

¿Un contrato inteligente es realmente igual a un contrato?

En la siguiente imagen se muestra la definición de contrato:



Un contrato inteligente está más apegado a la segunda parte de la definición. Es una serie de condiciones que se cumplirán.

A diferencia de un contrato en el sentido clásico, con un contrato inteligente las condiciones se cumplirán sin importar donde se encuentren las partes (diferentes lenguajes, leyes o jurisdicciones).

Un contrato podría ser entre dos individuos o entre un individuo y la blockchain.

¿Qué tan inteligente es un contrato inteligente?

Es mucho más parecido a un expendedor de chicles que a skynet.



Una definición formal de Contrato Inteligente:

Con lo antes discutido ya podemos enfrentarnos a una definición formal de Contrato Inteligente:

> “Un contrato inteligente es un protocolo de transacción computarizado que ejecuta los términos de un contrato. Los objetivos generales del diseño de contratos inteligentes son satisfacer condiciones contractuales comunes (como condiciones de pago, gravámenes, confidencialidad e incluso cumplimiento), minimizar las excepciones tanto maliciosas como accidentales, y minimizar la necesidad de intermediarios confiables. Los objetivos económicos relacionados incluyen la reducción de las pérdidas por fraude, los costos de arbitraje y ejecución, y otros costos de transacción". (Szabo, 1994).

Los puntos resaltantes de la anterior definición son:

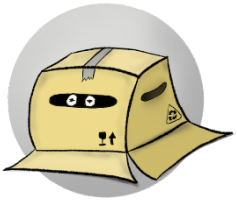
- El cumplimiento de condiciones pre pactadas.
- Minimizar excepciones a esas condiciones (se cumplirán siempre).
- Que se elimina la necesidad de intermediarios.
- Que se reducen costos y pérdidas.

¿Entonces cualquier acuerdo puede ser un contrato inteligente?

Nada que sea subjetivo o abierto a interpretación puede ser un contrato inteligente. A diferencia de un contrato en el sentido clásico, que es interpretado por jueces y jurados para la aplicación de justicia, un contrato inteligente será ejecutado por una computadora por lo que no puede ser subjetivo.

Esto no quiere decir que el proceso de tu DApp no pueda incluir juicios de valor, de calidad, o toma de decisiones complejas. Lo que quiere decir es que estas decisiones no las tomará el contrato. Tendrás que decírselas a partir de una votación o como input de un método.

Una blockchain y un contrato inteligente están aislados del mundo exterior



Otro punto importante es que los contratos inteligentes no tienen la capacidad de extraer datos de recursos fuera de la blockchain (off chain); esa información debe ser "empujada" (pushed) al contrato inteligente.

El tiempo en los contratos inteligentes

Esto incluye el tiempo. La forma en que medimos el tiempo normalmente implica la sincronización de nuestros dispositivos con un servidor de confianza. Esto no es posible en la blockchain. El tiempo solo puede conocerse de manera aproximada a partir de los registros de tiempo hechos por los validadores.

Un contrato inteligente percibe el tiempo solo a partir de la ejecución actual.

En conclusión:

A más información subjetiva o generada fuera de la blockchain use tu contrato inteligente más complejo será su implementación.

Datos de un contrato inteligente que podrían ayudar a entender mejor cómo funcionan:

- Usualmente son simples: si la lógica se vuelve compleja la separas en más de uno (como microservicios).

- Los contratos inteligentes no se ejecutan automáticamente de la manera que crees que lo hacen: solo responden cuando les preguntas y si no preguntas no hacen nada (ej. Un contrato inteligente como despertador no es posible).
- Los contratos inteligentes no pueden automatizar muchas cosas (como pagos y liquidaciones de facturas) como crees que lo hacen.

Fácil, difícil e imposible

Idea	Dificultad	Observación
Marketplace de NFTs	Fácil	Toda la información está dentro de la blockchain.
DApp de herencia de tus activos digitales (tokens)	Difícil	¿Cómo comprobarías que alguien murió? Necesitarás un oracle o mucho ingenio.
DApp de Seguros contra incendios	¿Imposible?	Demasiado subjetivo (ajustes de siniestros, estimación de riesgos).

~~Armemos un contrato~~



¿Con qué piezas cuento?: Partes de un Contrato Inteligente.

Memoria o almacén (Storage):



Puedes pensar en el Storage de un contrato Inteligente como un block de notas, costoso (de alta calidad) y que por su material se te hace de difícil lectura (empaña la vista).

Usas una libreta para anotar datos que quieres recordar a largo plazo. Por ejemplo, si administras una tienda que da productos a crédito, registras quienes te deben dinero. O si tienes un spa de mascotas, anotas información de los dueños, para regresarle el perrito correcto a cada persona.

Pero como es una libreta costosa, solo anotas información importante. Guardar información en el storage de un contrato Inteligente es costoso, por ello se guarda el mínimo necesario. Ej. La imagen de los NFTs no se guarda en los contratos. Lo que se guarda es la dirección de la imagen en un almacenamiento externo.

Continuando con la analogía, como es difícil leer de la libreta, no puedes leer todo su contenido en una consulta, y tomas precauciones al respecto [Otra analogía: se dice que escribir en la blockchain es como escribir en piedra. Si las hojas de la libreta fueran de piedra, pasar a la siguiente sería difícil].

Cada consulta a la información dentro de un contrato inteligente tiene costo, por lo que es una mala idea hacer consultas de mucha información a la vez.

Funciones o métodos (Lo que tu contrato puede hacer):

Una función es una pieza de código que realiza una tarea.

Un método es otro nombre para una función al pensar en un contrato inteligente como un objeto (OOP).

Un objeto en el mundo real se describe por la información que lo define (variables) y por las cosas que puede hacer (se asocian con verbos/acciones). Ej: una lámpara es alta o pequeña (variable tamaño), y puede encenderse o apagarse (métodos).

Ya se había comparado el storage de un contrato inteligente con un block de notas. Para escribir en él necesitarás un método. Si quieres leer de él también necesitarás un método.

Hay dos tipos de métodos para un contrato inteligente:

- Métodos que hacen cambios en el Storage (de escritura):
  Son métodos que escriben o cambian información de la memoria del contrato. Estos consumen gas. Son estos los que sostienen la lógica del contrato. Pueden entre otras cosas:

  - Escribir en el storage.
  - Desplegar un contrato (factory).
  - Transferir tokens.
  - Hacer stake.
  - Llamar a funciones de otros contratos (cross contract call).

- Métodos que consultan información del Storage (de lectura):
  Son métodos que solo consultan información de la memoria del contrato. Por lo pronto no cuestan gas. Aún así tienen límite en cuanta información puedes consultar por llamada.

Es útil recordar que todas las transacciones y llamadas a métodos son firmadas por el usuario, por lo que puedes restringir quien puede llamar a qué métodos. Por ejemplo, puedes hacer que una venta pueda ser cancelada solo por el usuario que la creó.

Llamadas entre contratos



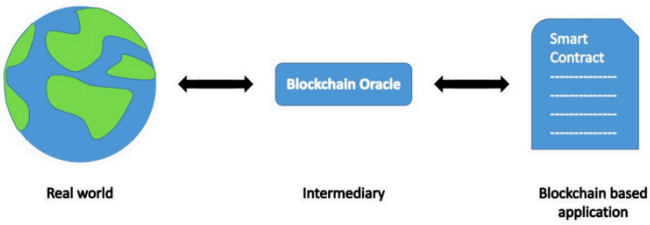
Los contratos inteligentes pueden comunicarse entre sí. A estas llamadas se les llama cross contract calls. Puedes tener en tu DApp varios contratos trabajando juntos. Por ejemplo; una dApp de NFTs podría organizarse como 3 contratos. Un contrato que cree y guarde los NFT (minter), otro a cargo de la lógica de compra y venta (marketplace) y por último uno a cargo del token de utilidad o gobernanza de la dApp.

Piezas/patrones adicionales para una DApp

Si intentas analizar una DApp que conozcas te encontrarás con que además de las tareas de las que se encargue el front, esta podría estar compuesta por más de un contrato inteligente o por cualquiera de las siguientes partes:

Nota que los NFTs, FT y DAOs son también contratos inteligentes.

Oracle

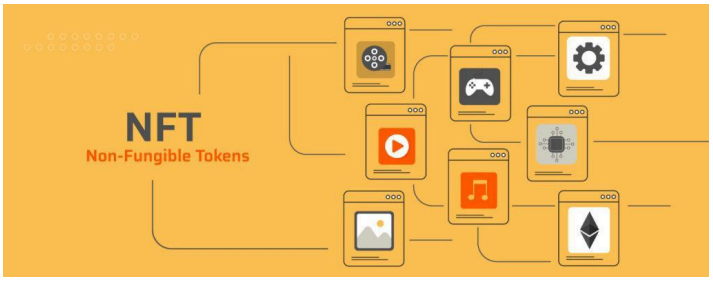


Los Blockchain Oracles son servicios ofrecidos por terceros que suelen estar fuera de la Blockchain y que permiten que los contratos inteligentes tomen decisiones finales.

El Oracle proporciona datos necesarios para la acción, pero no es la fuente de información. Es un ente que consulta la información y la autentifica para transmitirla.

Necesitas de un Oracle en tu DApp cuando no puedes evitar el tener que usar información generada fuera de la blockchain. Este es un recurso que debería preferirse evitarse porque implica riesgos de seguridad o de posibilidad de corrupción en tu contrato.

NFTs



Puedes incluir NFTs en el proceso de tu DApp cuando necesites que algo sea único, y que pueda demostrarse la pertenencia a un grupo. Por ejemplo, si quisieras entregar beneficios a un grupo selecto de usuarios, les entregas un NFT, el cuál puede ser transferido o transado entre ellos, pasando así los derechos al portador.

FT



Al incluir un FT en tu dApp podrías montar un sistema de incentivos al interior de tu aplicación. Puedes pensar en este elemento como las fichas de juegos dentro de un centro de entretenimiento. Las fichas solo tienen valor dentro del establecimiento, pero pueden llegar a tener mucho valor para los usuarios si los juegos son populares. En el caso de los FT puedes hacer que estos puedan intercambiarse por otros en un dex.

Podrías también usar FT para entregar la gobernanza de la aplicación a los usuarios.

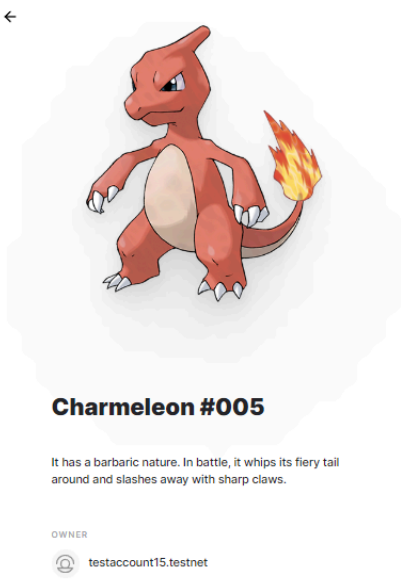
DAO

Puedes incluir una DAO en tu dApp a modo de dispositivo de gobernanza o solo incluir la idea de una decisión conjunta para resolver alguna decisión al interior de tu contrato. Por ejemplo, NearP2P utiliza uno de estos mecanismos para resolver disputas. NearP2P es una DApp de intercambio del token near por dólares. Si uno de los usuarios luego del intercambio abre una disputa por considerar que la otra parte no cumplió con el trato se abre un proceso de mediación. El proceso de mediación resuelve quien tiene la razón, el comprador o el vendedor. Se revuelve a través de voto mayoritario entre 3 partes; comprador, vendedor y el servicio P2P.

Herramientas de escalamiento:

- Indexadores: Necesitas un indexador para llevar la información de tu dApp fuera de la blockchain y poder tenerla como un servicio de consulta. Ej. Si visitan el Marketplace de paras.id podrán ver que la página muestra mucha información de manera rápida. Esa información no proviene de la blockchain, sería muy costoso mostrar toda esa data a través de llamadas a contratos. Esta proviene de un indexador.

- Servidores IPFS: Es utilizada para guardar media en un servicio también descentralizado. Ej. Las imágenes de los NFTs no se guardan en la blockchain o el estado de los contratos. Esta se guarda en un servicio de IPFS. En el contrato solo se guarda la dirección.



Así luce un NFT en la wallet de NEAR.


```
{
  token_id: '22915',
  owner_id: 'testaccount15.testnet',
  metadata: {
    title: 'Charmeleon #005',
    description: 'It has a barbaric nature. In battle, it whips its fiery tail around and slashes away with sharp claws.',
    media: 'bafybeifisztbqrhw62ax6ficbrc2fruin4nrljwhzsg6zex6cqi4hqz3wu',
    media_hash: null,
    copies: null,
    issued_at: null,
    expires_at: null,
    starts_at: null,
    updated_at: null,
    extra: null,
    reference: null,
    reference_hash: null
  },
  creator_id: 'testaccount15.testnet',
  approved_account_ids: {},
  royalty: { 'mcarbajalc2.testnet': 1000 }
}
```


Así luce un NFT dentro del contrato inteligente en el que vive.

Nota que en el contrato no se guarda la imagen. Solo se guarda la dirección de la imagen en un servicio externo. La información para encontrarla es la del campo “media”. Puedes ver la imagen en:

https://cloudflare-ipfs.com/ipfs/bafybeifisztbqrhw62ax6ficbrc2fruin4nrljwhzsg6zex6cqi4hqz3wu

Recursos adicionales

https://www.directivosyempresas.com/internet/tecnologia/4-cosas-que-debes-saber-sobre-el-blockchain-oracles/