



Bloque 4 · La DApp del MMDV Wine Token



De la cadena al usuario: cómo crear una DApp que conecta tu contrato con el mundo real.

☒ 1. Introducción: de la blockchain al usuario

Hasta ahora hemos recorrido un camino increíble. En el Bloque 1 entendimos qué es tokenizar un activo y por qué convertir un lote de vino en un token ERC-20 es una forma poderosa de darle trazabilidad y valor digital. En el Bloque 2 construimos el contrato inteligente: sus funciones, su cap, sus eventos y toda la lógica interna que define al **MMDV Wine Token**.

Y en el Bloque 3 dimos el salto grande: **lo desplegamos en la blockchain**, lo verificamos en Etherscan, lo añadimos a nuestra wallet y mintteamos el primer lote real del proyecto (Lote #0). Ese fue el momento en el que la idea dejó de ser teoría y se convirtió en un activo vivo dentro de Ethereum.

Pues bien... si el Bloque 3 fue el momento de “poner el contrato en la cadena”, **este Bloque 4 es el momento de acercarlo a las personas**.

🌐 ¿Qué significa este bloque?

En este módulo vamos a construir una **mini-DApp**, una pequeña aplicación web que se conecta directamente con el contrato del MMDV Wine Token y permite interactuar con él sin necesidad de usar Remix o Etherscan.

Una DApp es, en esencia:

- una web normal,
- que se conecta a una wallet real,
- que lee datos en la blockchain,
- y que ejecuta funciones del contrato con una sola firma.



Es decir, lo que en los bloques anteriores hacíamos desde herramientas técnicas, ahora lo podrá hacer cualquier usuario desde una interfaz simple, visual y accesible.

⌚ ¿Por qué es importante este paso?



Porque la tecnología blockchain no se queda en el código.

Un proyecto real necesita que alguien pueda interactuar con él:

- consultar su balance,
- ver información del token,
- ejecutar una función,
- verificar un lote,
- consumir un activo (redeem)...

y hacerlo desde un lugar cómodo, sin tocar código.

Este bloque es la transición natural desde la lógica técnica del contrato hacia la experiencia de uso. Aquí es donde la blockchain deja de ser “para programadores” y se convierte en algo que cualquier persona puede utilizar.

🚀 ¿Qué vamos a conseguir al final del bloque?

Cuando terminemos, habrás construido:

- Una WebApp con un botón de **“Conectar wallet”**.
- Un panel que muestra los datos del token en tiempo real (cap, supply, decimals, owner...).
- Un panel para que el usuario vea **su propio balance**.
- La primera funcionalidad interactiva real: **la ejecución de la función redeem() desde una interfaz visual**.
- Lectura de eventos y trazabilidad del lote redimido.
- Y, lo más importante... una comprensión profunda del ciclo completo: **Contrato → Blockchain → Wallet → DApp → Usuario**.

Al terminar este bloque podrás decir que has construido no solo un token, sino **la primera versión pública de una aplicación Web3 completa**, lista para evolucionar hacia proyectos reales o incluso hacia la versión en XRPL.

💭 Un bloque más práctico, más visual y más cercano

Si el Bloque 3 fue el bloque “técnico”, este es el bloque “interactivo”. Nos moveremos entre HTML, JavaScript, Metamask y Ethers.js, pero siempre con



explicaciones claras, ejemplos sencillos y una estructura visual que cualquiera podrá seguir.

Y como siempre... documentaremos todo lo que nos fue pasando: las dudas, los errores de conexión, los ABI, las pruebas, los mensajes de Metamask, y esos “¿por qué no conecta ahora?” que forman parte del aprendizaje real.

2. Qué vamos a construir exactamente

En este bloque vamos a dar forma a algo que, hasta ahora, solo existía en herramientas técnicas: **una aplicación web real que se conecta con nuestro contrato inteligente**. Una pequeña DApp, sencilla pero totalmente funcional, que convierte el MMDV Wine Token en una experiencia accesible para cualquier usuario.

La idea es crear una interfaz clara, visual y directa donde una persona pueda:

- **Conectar su wallet** (Metamask) con un solo clic.
- **Ver información del token** (cap, supply, decimales, propietario).
- **Consultar su balance personal** de MWT.
- **Redimir** (redeem) un lote de tokens directamente desde la web.

El objetivo es enseñar cómo se construye, desde cero, el puente entre la blockchain y el usuario final. Nada de especulación: este módulo muestra cómo una WebApp puede leer datos reales de la cadena y ejecutar funciones del contrato en tiempo real.



🎯 ¿Por qué esta DApp es importante?

Porque un contrato inteligente, por sí solo, no es suficiente. Puedes desplegarlo, verificarlo y verlo en Etherscan... pero si quieres que alguien lo use, necesitas darle una puerta de entrada intuitiva.

Esta mini-DApp es precisamente eso: **una demostración real de cómo se transforma un contrato en un producto utilizable**.

Además, te prepara para lo que viene después:

- Integración en proyectos reales.



- Versiones ampliadas en XRPL.
- Paneles administrativos o interfaces de usuario más avanzadas.

💡 ¿Qué incluye exactamente la DApp?

Aquí sí merece la pena usar bullets, porque clarifica lo que se entregará:

- **index.html**
La estructura de la interfaz: botones, paneles de información, área de interacción.
- **script.js**
El corazón de la DApp: conexión con Metamask, carga del contrato, lectura de datos y ejecución de funciones.
- **style.css**
Diseño sencillo, limpio y visual, coherente con el estilo MMDV que vienes construyendo.
- **ABI + dirección del contrato**
Lo que permite que la WebApp realmente sepa “cómo hablar” con nuestro token en Sepolia.

Este conjunto de archivos forma la base mínima y profesional de cualquier DApp real.

🌐 Flujo general de funcionamiento



- El usuario abre la WebApp.
- Hace clic en “**Conectar Wallet**”.
- Metamask solicita permiso.
- Una vez conectada, la DApp lee:
 - dirección del usuario,
 - balance,
 - datos del contrato.
- El usuario puede ejecutar **redeem()** desde un botón visual.
- Metamask pide confirmación.

- La transacción se registra en la blockchain.
- La interfaz actualiza el estado.

Este flujo completo permite ver cómo una DApp actúa como puente entre navegador, wallet y blockchain.

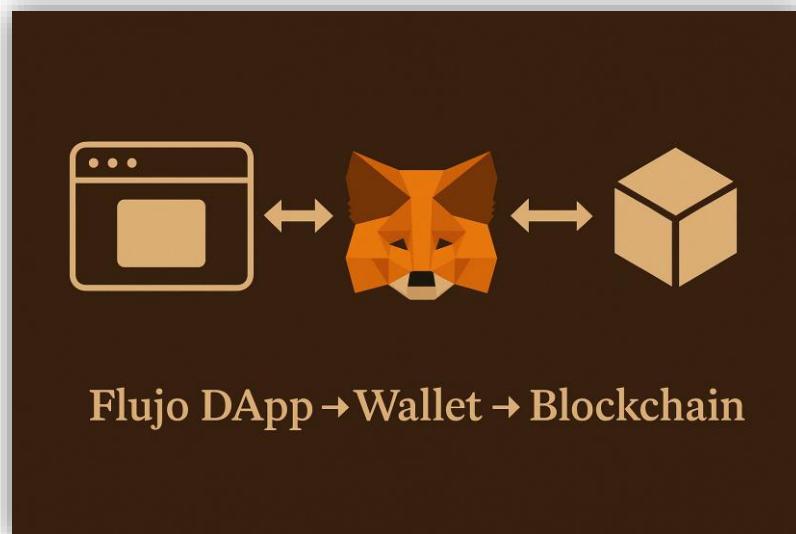


Imagen 1 — Flujo básico de una DApp Web3

Ilustración conceptual del flujo completo entre la aplicación web (DApp), la wallet del usuario (Metamask) y la blockchain (Sepolia/Ethereum). La imagen muestra cómo la DApp se comunica con la wallet para firmar transacciones y, a su vez, cómo la wallet interactúa directamente con la red blockchain para ejecutar operaciones como el redeem o la lectura de datos.

3. Preparación del entorno para la DApp

Para construir una DApp necesitamos tres piezas fundamentales: una **página web**, un **mecanismo de conexión con la wallet** y una forma de **hablar con la blockchain**. La buena noticia es que no hace falta nada complejo: con unos pocos archivos y una librería ligera, podemos levantar una aplicación Web3 completa.

En este punto preparamos el terreno. Es decir, dejaremos listo todo lo necesario para que la WebApp pueda detectar Metamask, leer datos del contrato y ejecutar funciones reales en Sepolia.

3.1. Archivos base del proyecto

Nuestra DApp se apoya en tres archivos muy sencillos:

- **index.html**

La estructura visible: botones, paneles, áreas donde aparecerá la información del token. Es la “cara” de la aplicación.

- **style.css**

Los estilos: tipografías suaves, colores inspirados en el universo MMDV y un diseño limpio que ayude a entender qué está pasando en la interfaz.

- **script.js**

El corazón de todo. Aquí vive el código que conecta la web con la wallet y con la blockchain. Se carga el ABI, se leen datos del contrato y se ejecutan las funciones.

Con estos tres archivos ya podemos construir cualquier DApp sencilla y perfectamente funcional.

3.2. Cómo conectaremos la DApp con Metamask

Metamask actúa como el puente entre el navegador y la blockchain. La WebApp **no puede** firmar transacciones por sí sola; necesita que Metamask se las firme.

El mecanismo es simple:

1. La DApp pide permiso al usuario para conectarse.
2. Metamask muestra una ventana emergente.
3. El usuario acepta, y a partir de ahí la web conoce la dirección de su wallet.
4. Cada vez que ejecutamos una función del contrato, Metamask vuelve a aparecer para firmar.

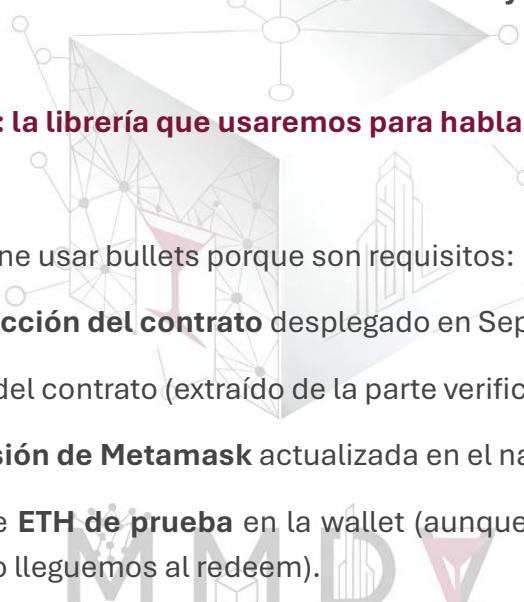
Es el equivalente a “entrar” en una aplicación... pero en Web3.

3.3. Ethers.js: la librería que usaremos para hablar con el contrato

Para que la web pueda leer y escribir en la blockchain, necesitamos una librería. Nosotros usaremos **Ethers.js**, porque es ligera, moderna y perfecta para DApps pequeñas. Ethers nos permite:

- detectar la wallet,
- crear un proveedor conectado a Metamask,
- cargar la dirección del contrato,
- usar el ABI,
- leer datos como totalSupply o balance,
- y ejecutar funciones como redeem con una simple llamada.

La idea clave: **Ethers es el traductor entre la web y la blockchain.**



3.4. Ethers.js: la librería que usaremos para hablar con el contrato

Aquí sí conviene usar bulletts porque son requisitos:

- La **dirección del contrato** desplegado en Sepolia.
- El **ABI** del contrato (extraído de la parte verificada en Etherscan).
- La **versión de Metamask** actualizada en el navegador.
- Algo de **ETH de prueba** en la wallet (aunque solo lo necesitaremos cuando lleguemos al redeem).

Con estas cuatro cosas ya podemos trabajar sin bloqueos.

3.5. ¿Hace falta un servidor?

Con estas cuatro cosas ya podemos trabajar sin bloqueos. No para este módulo.

La DApp puede abrirse directamente desde un archivo local o un hosting muy simple (GitHub Pages, por ejemplo). Más adelante, si quieres una versión más avanzada, sí podríamos montar un backend, pero para este bloque:



HTML + CSS + JS es más que suficiente.



Imagen 2 — Componentes básicos del entorno de una DApp Web3

Ilustración conceptual del entorno mínimo necesario para construir una aplicación Web3: los archivos base (HTML, CSS, JS), la conexión con el navegador y Metamask, y la comunicación final con la blockchain. Un esquema visual sofisticado y moderno que refleja el flujo técnico real del proyecto.

💡 4. Los condicionantes del contrato (muy importante en este bloque)

Antes de construir la interfaz, necesitamos entender que nuestro contrato inteligente no es un “token básico”. El **MMDV Wine Token** incorpora una serie de condicionantes pensados para proteger el proyecto, asegurar la trazabilidad y limitar las operaciones sensibles. Son reglas internas del contrato que la DApp debe respetar y que determinan qué funciones puede ejecutar un usuario y cuáles están restringidas.

Estos condicionantes forman la base de una tokenización seria: evitan abusos, refuerzan la seguridad y garantizan que cada acción del usuario está alineada con el propósito del token.

4.1. Propiedad: solo el owner puede ejecutar ciertas acciones



Dentro del contrato existe un propietario, un owner, establecido de forma permanente en el despliegue. Ese owner es el único con permisos para ejecutar algunas funciones críticas, como **redeem**.

La razón es simple: en un proyecto real, destruir tokens o marcar un lote como “consumido” es una operación sensible. No puede ejecutarla cualquiera.

En la DApp veremos esto reflejado de forma natural: si la wallet conectada no es el owner, la interfaz limitará ciertas opciones o mostrará mensajes de uso restringido.

4.2. El cap total: un límite que no se puede sobrepasar

Nuestro contrato tiene un **cap**, un máximo absoluto de tokens que pueden existir.

Es una protección esencial en proyectos de tokenización: el número total no puede crecer indefinidamente, porque representa un activo físico finito (como el vino de una cosecha concreta).

La DApp debe conocer este límite para mostrar información precisa: el supply existente, el cap y la diferencia entre ambos. Es una forma sencilla de visualizar cuánto “espacio” queda para futuros lotes.

4.3. El batchId: cada lote debe tener identidad propia



En el Bloque 3 aprendimos que cada mint va asociado a un **batchId** único. Esto no es un adorno: es la clave de la trazabilidad del proyecto.

El batchId convierte cada lote de tokens en algo identificable, consultable y registrable dentro de los eventos del contrato.

La DApp debe mostrarlo con claridad: qué lote se creó, quién lo recibió y en qué momento. Cada lote es un fragmento de historia registrado en la cadena.

4.4. Los eventos: cómo la blockchain cuenta lo que ocurre



Nuestro contrato emite eventos cada vez que sucede algo relevante, como el mint o el redeem. Esos eventos son el registro público y verificable de cada acción. La DApp no solo puede leerlos... **debe** leerlos.

Los eventos permiten reconstruir la historia completa de un token: cuándo se creó, cuántos se asignaron, a quién fueron, y en el caso del redeem, cuándo se destruyeron.

Entender esta narrativa on-chain es fundamental en un proyecto que busca trazabilidad.

4.5. Los decimales: el puente entre el mundo humano y el mundo técnico

El contrato trabaja con unidades matemáticas, no con números “humanos”. Nuestro token usa 18 decimales, así que la DApp debe convertir valores automáticamente para que el usuario vea cantidades normales, no cifras imposibles de interpretar.

Este es uno de esos detalles que no se ven... pero sin ellos, nada funciona bien.



Imagen 3 — Condicionantes internos del MMDV Wine Token

Representación visual de las reglas fundamentales del contrato inteligente: la propiedad (owner), el límite máximo del token (cap), la identidad única de cada lote (batchId) y los eventos que registran cada acción en la blockchain. Estos elementos definen cómo funciona el token y marcan los límites operativos que la DApp debe respetar.

💻 5. Diseño de la DApp (UX muy amigable)

El diseño de la DApp nace con una idea muy concreta: reducir la complejidad de la blockchain a una interfaz que cualquiera pueda entender. No buscamos impresionar con efectos, sino transmitir claridad. La página está organizada en secciones que guían al usuario sin que tenga que pensar demasiado. Todo está donde debe estar.

- *Home: el primer contacto*

Al entrar en la DApp, el usuario solo ve lo esencial. Un mensaje corto explicando para qué sirve la aplicación y un gran botón en el centro: **Conectar Wallet**.

La idea es que en cinco segundos entienda cuál es el siguiente paso. Sin menús, sin distracciones. El diseño le dice: “*Esto empieza aquí*”.

- *La información del token*

Una vez conectada la wallet, aparece un panel con datos clave del MMDV Wine Token.

No mostramos nada técnico que pueda generar dudas; solo lo imprescindible para entender el comportamiento del token:

- el cap total,
 - el supply actual,
 - los decimales,
 - la dirección del contrato
 - y quién es el owner.
- 

Todo se presenta en bloques visuales simples, con iconos y texto suave, de manera que el usuario tenga una visión directa de lo que está consultando en la blockchain.

- *La información del usuario*

Justo debajo aparece un panel con la dirección pública conectada y el balance personal del token. Este momento es importante: convierte la experiencia en algo propio.



El usuario ve su wallet, su saldo y su huella en el sistema. La interfaz hace la conversión de decimales para mostrar cantidades “humanas”, no cifras técnicas.

- *La acción principal: Redeem*

El corazón del bloque. El diseño presenta *redeem()* como una acción clara, ordenada y bien explicada. El usuario selecciona el lote, pulsa el botón y Metamask se encarga de la firma.

Si no es el owner, la interfaz lo indica con un mensaje amable y transparente. El objetivo es que nadie se pierda y que la aplicación transmita seguridad en todo momento.

- *Los eventos: la historia del token*

En la parte inferior de la página, la DApp muestra los últimos eventos del contrato: mints, redeems y cualquier acción relevante.

Es una forma visual de enseñar cómo todo queda registrado y cómo la blockchain actúa como un libro abierto.

No hace falta ir a Etherscan: la propia DApp cuenta la historia..



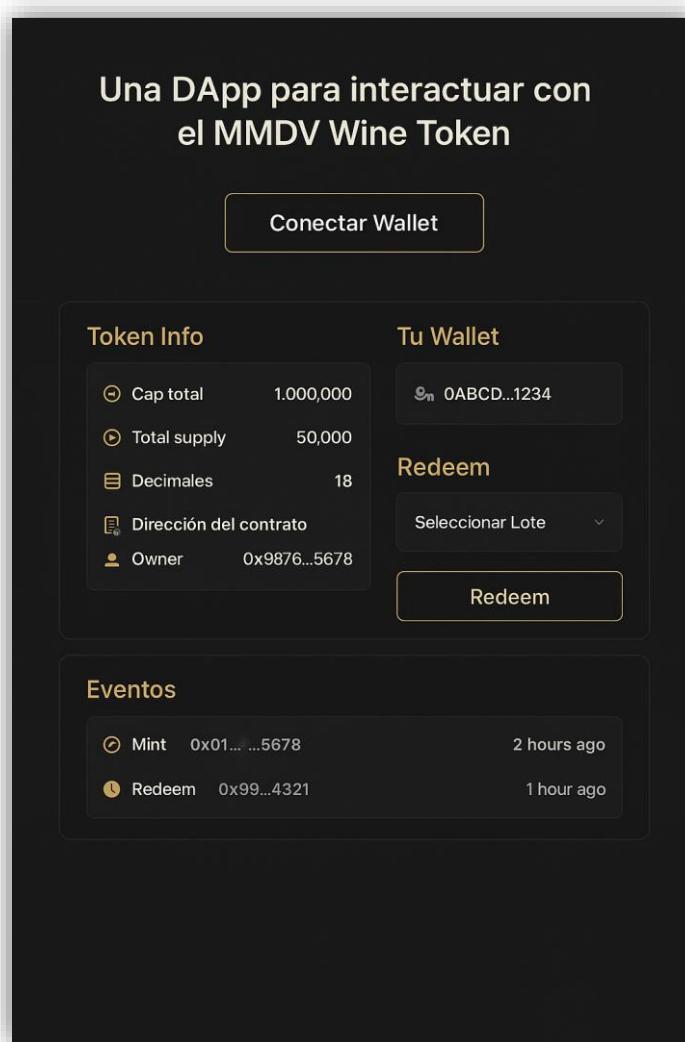


Imagen 4 — Interfaz completa de la DApp del MMDV Wine Token

Captura conceptual de la aplicación Web3 desarrollada en este bloque. La interfaz muestra el botón de conexión con Metamask, los paneles de información del token y del usuario, la acción de Redeem y el listado de eventos recientes. Una vista clara y moderna del flujo real entre usuario, wallet y blockchain.

💡 6. Conexión con Metamask

La conexión con Metamask es el primer acto “real” de interacción entre el usuario y la blockchain. Es el momento en el que la DApp deja de ser una página web y pasa a convertirse en una aplicación Web3. Todo comienza con un clic.

Cuando el usuario pulsa “**Conectar Wallet**”, la DApp envía una solicitud al navegador pidiéndole acceso a la extensión de Metamask. Metamask responde inmediatamente con una ventana emergente —siempre del lado del usuario— donde muestra la dirección disponible y le pide permiso. Este detalle es clave: la



DApp nunca ve ni controla la clave privada; solo recibe la dirección pública y la capacidad de pedir firmas.

Una vez el usuario acepta, la DApp obtiene dos cosas:

- la dirección de la wallet,
- y la confirmación de que podrá utilizar Metamask para firmar cualquier operación posterior, como consultar datos o ejecutar el redeem.

Desde ese momento, la interfaz se actualiza automáticamente. Los bloques que estaban ocultos se despliegan, el panel de información del token se rellena en tiempo real y el panel del usuario muestra su dirección y su balance. La experiencia transmite una sensación clara: “Ahora sí, estás conectado a la blockchain.”

Si el usuario cambia de cuenta o de red dentro de Metamask, la DApp reacciona en tiempo real. Se actualizan los datos, se refresca la pantalla y la aplicación garantiza que siempre estemos trabajando en **Sepolia**, la red correcta del proyecto.

Este pequeño proceso —solo unos segundos— es uno de los más importantes del módulo: convierte la web en un puente real hacia el contrato inteligente. A partir de aquí, cada acción se firma, se envía a la cadena y se verifica. El alumno experimenta de forma directa lo que significa trabajar en un entorno blockchain.

7. Llamadas reales al contrato (Ethers.js + ABI)

Una vez la wallet está conectada, la DApp necesita dos cosas para interactuar con el contrato: la **dirección del contrato** y su **ABI**. Esas dos piezas permiten que la web entienda qué funciones existen, qué datos puede leer y qué operaciones puede ejecutar. Es, literalmente, darle a la DApp el “diccionario” de tu token.

Con esa información, la aplicación crea una instancia del contrato a través de Ethers.js. Ese objeto, que vive en el script.js, es el intermediario que convierte lo que ocurre en la web en instrucciones comprensibles para la blockchain. A partir de ese instante, cada dato que aparece en la pantalla proviene directamente de Sepolia.

7.1. Lecturas (read): conocer el estado real del token

Las primeras llamadas que ejecuta la DApp son de lectura. No requieren firma, no consumen gas y son inmediatas. Cuando el usuario se conecta, la aplicación consulta:

- el cap total,
- el total supply,
- los decimales,
- el owner,
- y el balance del usuario.

Esas lecturas refrescan la interfaz y generan la sensación de que la DApp “cobra vida”. No hay datos inventados ni simulados: todo lo que se muestra está en la blockchain en ese mismo momento. Esta transparencia es lo que distingue una DApp de una web tradicional.

7.2. Escritura (write): ejecutar acciones reales en la blockchain

La parte más interesante ocurre cuando el usuario ejecuta algo que **modifica** el estado del contrato. En este módulo, esa acción es el **redeem()**.

Al pulsar el botón, la DApp prepara la llamada con los parámetros adecuados (dirección, lote, cantidad si la hubiera). Antes de enviar nada, Metamask aparece y pide al usuario que firme la operación. Aquí la blockchain impone sus normas: sin firma del usuario, **no hay transacción**.

Una vez firmada, la transacción se publica en la red y queda a la espera de confirmación. Durante esos segundos, la DApp muestra un pequeño estado intermedio. Cuando llega la confirmación, el panel se actualiza y el evento aparece tanto en Etherscan como en la sección de eventos de la propia aplicación.

Este proceso —clic, firma, confirmación, actualización— es el mejor ejemplo práctico de cómo funciona una transacción blockchain desde el punto de vista del usuario.

7.3. La importancia de los errores y su manejo



En este punto también aparece la parte “humana” del desarrollo: pequeñas situaciones que la DApp debe interpretar correctamente. Si el usuario no está en Sepolia, la interfaz avisa. Si intenta ejecutar redeem sin ser owner, la DApp muestra un mensaje claro. Si el contrato devuelve un execution reverted, la aplicación lo traduce a un texto comprensible.

El objetivo no es solo que funcione, sino que el alumno entienda **por qué** algo no se puede ejecutar.

Este apartado cierra el ciclo técnico: la DApp ya no es un diseño bonito, sino una aplicación que **lee, interpreta, transmite, firma y actualiza información real en la cadena**.

8. La función redeem explicada con calma

La función **redeem()** es el momento en el que el MMDV Wine Token demuestra que no es un simple token fungible más, sino una representación fiel de un activo que tiene un ciclo de vida. Si el mint simboliza la creación del lote, el redeem simboliza su consumo, cierre o retirada. Es la otra mitad del proceso.

En este bloque vemos esa mecánica de principio a fin: desde lo que significa conceptualmente hasta cómo lo vive un usuario en la DApp.

8.1. Qué representa redeem() en un proyecto real

El redeem no es una transferencia ni una operación administrativa: es una **destrucción controlada de tokens**, una acción irreversible que marca un hito en la historia del lote.

Puede representar:

- que el vino ha sido entregado,
- que el lote físico se ha agotado,
- que se cierra un ciclo logístico,
- o que un activo deja de estar disponible.



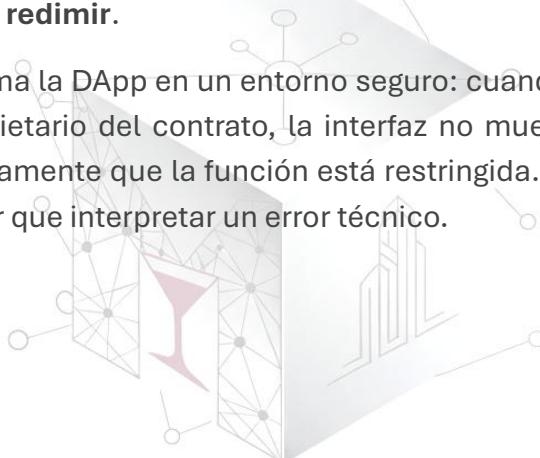
Cada redención queda grabada en la blockchain como un evento público, creando un registro permanente del momento en que ese lote dejó de existir digitalmente.

8.2. Por qué solo el owner puede ejecutar el redeem

La destrucción de tokens es una acción crítica. Permitir que cualquier usuario la ejecute sería un riesgo directo para el proyecto, porque impacta en la trazabilidad y en la correspondencia entre el activo físico y su versión tokenizada.

El contrato protege esta operación con un condicionante estricto: **solo el owner puede redimir**.

Esto transforma la DApp en un entorno seguro: cuando la wallet conectada no es el propietario del contrato, la interfaz no muestra el botón activo o explica directamente que la función está restringida. El usuario entiende la regla sin tener que interpretar un error técnico.



8.3. Qué ocurre exactamente cuando se ejecuta `redeem()`

Aquí es donde vemos la magia. Cuando el owner pulsa el botón en la DApp:

1. La aplicación prepara la llamada con los datos necesarios (generalmente el ID del lote o la cantidad a redimir).
2. Metamask abre una ventana pidiendo confirmación, dejando claro que se trata de una acción irreversible.
3. Cuando el usuario firma, la transacción viaja a la red Sepolia.
4. La blockchain valida la operación.
5. El contrato destruye los tokens correspondientes.
6. Se emite un **evento redeem**, que queda publicado en Etherscan, con su lote, su dirección y su timestamp.



7. La DApp recibe esa confirmación y actualiza la interfaz: el supply cambia, el lote aparece marcado como redimido y el evento se muestra en la lista inferior.

La sensación para el alumno es muy potente: lo que acaba de hacer no es un “clic”, sino una operación registrada públicamente en la cadena para siempre.

8.4. Cómo lo mostramos en la interfaz

Aunque la lógica interna es densa, la experiencia en la DApp es extremadamente sencilla. El usuario solo ve:

- un menú para seleccionar lote,
- un botón Redeem,
- una firma en Metamask,
- y una confirmación final.

Todo lo demás —la validación del contrato, la emisión del evento, la actualización del supply— ocurre de forma transparente. La interfaz oculta las partes complejas y muestra solo lo que aporta claridad.

8.5. El valor pedagógico de este punto

El alumno, al ejecutar redeem por primera vez, comprende de manera intuitiva tres principios fundamentales de Web3:

- que la blockchain es un sistema de verdad pública,
- que cada acción tiene un impacto irreversible,
- y que la relación entre front-end, wallet y contrato es directa, sin intermediarios.

Redeem convierte un concepto abstracto en una experiencia real, visible y verificable.

Es la primera vez en el curso en la que el alumno **modifica** el estado de la cadena desde una interfaz construida por él mismo. Ese es el aprendizaje más valioso del módulo.

▣ 9. Flujo completo desde el punto de vista del usuario

La experiencia del usuario dentro de la DApp sigue un recorrido muy natural. Todo empieza cuando abre la página y se encuentra con un entorno limpio, sin distracciones. Hay un mensaje claro que le explica para qué sirve la aplicación y un único paso obvio para avanzar: pulsar **Conectar Wallet**. En ese instante, la web deja de ser una simple interfaz estática y se convierte en una puerta de entrada al mundo blockchain.

Metamask aparece automáticamente y le pide permiso. El usuario acepta, quizá sin saber aún lo que implica, pero en el momento en que lo hace, la página cobra vida. La interfaz se reorganiza: donde antes había silencio, ahora hay información. Ve su dirección pública, su balance y datos del token que no provienen de un servidor, sino directamente de Ethereum. Es como si la DApp le dijera: “Ahora sí, estamos conectados”.

Con todo preparado, llega el momento de interactuar. El usuario se desplaza hacia la sección central de la página y ve la acción principal del módulo: **redeem**. Si no es el owner, la DApp se lo explica sin dramatismos y sin errores técnicos. Si lo es, la opción aparece habilitada. El usuario elige un lote y pulsa el botón. La acción parece sencilla, pero detrás ocurre todo un proceso: la DApp prepara la transacción, Metamask pide una firma y la blockchain se encarga del resto.

Tras la confirmación, el usuario ve cómo el panel cambia. El supply se actualiza, el lote aparece marcado como redimido y, unos segundos después, el evento queda reflejado en la parte inferior de la página. No hace falta ir a Etherscan: la historia del contrato se actualiza delante de sus ojos. Es un momento de claridad total. Ha ejecutado una acción en la blockchain desde una interfaz sencilla, y lo ha visto reflejado en tiempo real.

Ese es el flujo completo. No hay complicación, no hay rodeos: conectar, ver, actuar y confirmar. Todo está diseñado para que el usuario sienta que entiende lo que está haciendo, incluso si es la primera vez que toca Web3. La DApp no solo le permite interactuar con el contrato: le enseña cómo hacerlo.

☒ 10. Pruebas reales y resultados esperados

Una vez terminada la DApp, llega el momento de comprobar que todo funciona como debe. Y aquí no hablamos de pruebas complejas ni de entornos avanzados: las mejores pruebas son las que reproducen exactamente lo que hará un usuario



real. La idea es simple: abrir la DApp, conectarse, interactuar con ella y observar cómo responde.

El primer paso de la prueba consiste en cargar la aplicación desde el navegador y pulsar **Conectar Wallet**. Si Metamask aparece de inmediato, significa que la comunicación base está funcionando. Si el usuario acepta, la interfaz debe actualizarse con los datos del token: el cap, el total supply, los decimales y el owner. Esta actualización es el primer indicador de éxito. Si los datos aparecen, la DApp está leyendo correctamente de la blockchain.

Después llega el momento de las pruebas de interacción. El owner puede intentar ejecutar un **redeem**. Al pulsar el botón, Metamask debería abrir una ventana pidiendo confirmación y mostrando el resumen de la transacción. Si el usuario firma, la operación pasa a la red y, tras unos segundos, debe aparecer confirmada. La DApp debería reflejar el cambio: el supply se reduce, el lote desaparece o queda marcado como redimido, y el evento aparece en la lista inferior. Si estas tres señales coinciden, la prueba es un éxito total.

En paralelo, el alumno puede abrir Etherscan para ver la transacción en tiempo real. Aunque no es obligatorio, suele ser uno de los momentos más reveladores del módulo: comprobar que lo que ocurre en la web está realmente ocurriendo en la cadena. Esa doble verificación —en la DApp y en Etherscan— da la tranquilidad de que el flujo completo es correcto.

También es importante observar “lo que pasa cuando algo no pasa”. Si el usuario está en la red equivocada, la DApp lo debería indicar. Si se intenta redimir sin ser owner, la interfaz debe bloquear la acción o mostrar un aviso. Si la wallet no tiene ETH de prueba, la transacción no podrá enviarse. Estos comportamientos no son fallos; son parte natural de un entorno blockchain. Lo relevante es que la DApp los gestione de forma comprensible, sin mensajes crípticos.

El resultado esperado de estas pruebas es simple: que la DApp se comporte siempre de forma coherente, que cada acción tenga un efecto visible y que cada confirmación o bloqueo tenga una explicación lógica. Cuando eso ocurre, el alumno entiende no solo la aplicación, sino también la tecnología que hay detrás.

A continuación se muestran tres vistas reales de la DApp desarrollada en este bloque, que permiten visualizar tanto las reglas educativas como el estado del contrato y las acciones disponibles para el usuario.



MWT5 - PROYECTO EDUCATIVO TOKEN DE VINO EN SEPOLIA

MWT5 – MMDV Wine Token · Proyecto Educativo

Esta Dapp muestra cómo un **smart contract** ERC-20 puede incorporar reglas educativas: precio simbólico de salida, periodo de cancelación, bloqueo mínimo, bonus por antigüedad, prioridad para próximos lotes y opción de recompra por parte del owner. Todo está pensado como laboratorio didáctico, **no como producto financiero real**.

Red: Sepolia Testnet

Dirección del contrato: 0x8e913dEadC1F9c25a2ADc78AD793cEf72CD02dc2

¿Qué representa MWT5?

Un token = una botella de vino simbólica. Aquí sólo medimos unidades y reglas, no euros reales. El objetivo es **explicar conceptos**, no vender vino on-chain.

Reglas educativas clave

- **Cap educativo:** hasta 1.000.000 tokens enteros.
- **Precio simbólico inicial:** 10 unidades por token.
- **Cancelación 24h:** el comprador puede devolver los tokens al owner.
- **Bloqueo 1 año:** no se permiten transfers a terceros antes del año.
- **Precio mínimo de reventa:** +25% por año completo después del primero.
- **Bonus de fidelidad:** +10% de tokens tras 4 años holdeando.
- **Recompra del owner:** el owner puede recomprar a precio simbólico fijo.
- **Prioridad:** quienes participan en este lote tienen preferencia en los siguientes.

Cómo usar esta página

- **Explora los datos del contrato en la zona de "Estado global".**
- **Opcional:** pulsa "Conectar wallet" para que la Dapp lea tus datos como holder.
- **Juega con el simulador de precio mínimo** según tu antigüedad.
- **Si estás dentro del periodo,** prueba la **cancelación 24h** con Sepolia testnet.
- **Si conectas con la dirección del owner,** verás el panel de **buyback**.

- **Reglas:** condiciones de funcionamiento del token.
- **Contrato:** ver código fuente y datos del contrato.
- **Holder:** ver lista de holders y sus balances.
- **Buyback:** ver historial de compra y venta.



Imagen A – Condiciones Educativas y presentación del proyecto MWT5

Vista inicial de la DApp del proyecto MWT5. Aquí se presenta el objetivo educativo del token, acompañado de las reglas simbólicas diseñadas para explicar conceptos de mercado on-chain: precio base, periodo de cancelación, bloqueo mínimo, bonus de antigüedad, prioridad en lotes y recompra del owner. Esta pantalla sirve como introducción teórica al funcionamiento del MMDV Wine Token, en un formato claro y accesible para el alumno.



Estado global del contrato

Información que aplica a todo el lote tokenizado MWT5.

TOKEN MMDV Wine Token V5 (MWT5)	Parámetros simbólicos	Conexión Web3
SUPPLY EMITIDO / CAP EDUCATIVO 8306 / 1000000 MWT5	PRECIO BASE EDUCATIVO 10 unidades simbólicas	Puedes explorar todo en modo lectura sin conectar nada. Si quieras interactuar con el contrato (por ejemplo, cancelación 24h) conecta una wallet de prueba en Sepolia.
OWNER DEL CONTRATO 0xC9d05cd8fE0b2611A0b79364DF6c43d2612a919	PERÍODO DE CANCELACIÓN 24 horas	CUENTA CONECTADA No conectada
	BLOQUEO MÍNIMO DE TRANSFERENCIAS 1.0 años	Conectar wallet
	PRECIO SIMBÓLICO DE RECOMpra (OWNER) 12 unidades por token	Desconectar wallet

Panel del holder

Aquí puedes ver cómo aplican las reglas al titular de una dirección concreta.

Dirección a inspeccionar DIRECCIÓN (OPCIONAL) Ex... Si se deja vacío, usa la HOLDER ACTUAL	Posición educativa BALANCE ACTUAL (ON-CHAIN) — CANTIDAD ASOCIADA AL TRACKING EDUCATIVO — PRIMERA COMPRA REGISTRADA — PRIORIDAD EN FUTUROS LOTES —	Valor, bonus y mínimos MULTIPLICADOR TEÓRICO POR ANTIGÜEDAD — BONUS DE FIDELIDAD POTENCIAL (4 AÑOS) — PRECIO MÍNIMO EDUCATIVO DE REVENTA —
--	--	---

Imagen B — Estado global del contrato y conexión Web3.

Pantalla que muestra la información global del MMDV Wine Token en la red Sepolia: supply emitido, cap educativo, parámetros simbólicos y restricciones del contrato. A la derecha aparece el módulo de conexión con Metamask, lo que permite al alumno interactuar con la blockchain en modo lectura o ejecutar funciones cuando la wallet está conectada. Es una sección clave para comprender qué datos son del contrato, cuáles afectan a todos los holders y cómo se visualizan en una DApp real.



Simulador de precio mínimo

Introduce un precio simbólico por token y deja que el contrato te diga si respeta las reglas educativas de MWT5.

Validar precio propuesto

PRECIO PROUESTO (UNIDADES POR TOKEN)

Ej.: 15

Validar precio

Lectura recomendada

Esta lógica de mínimos educativos está programada dentro del smart contract, no en la interfaz. La Dapp simplemente llama a validateResalePrice() y muestra la respuesta. Es una buena forma de explicar el concepto de "reglas de mercado on-chain".

Acciones educativas (requieren wallet)

Estas funciones envían transacciones reales en la testnet de Sepolia. Asegúrate de usar sólo cuentas y ETH de prueba.

Cancelación dentro de 24h

Permite al comprador devolver tokens al owner durante las primeras 24h desde su primera compra. El reembolso económico se simula fuera de la cadena; aquí sólo se registra el movimiento de tokens.

CANTIDAD A DEVOLVER (TOKENS)

Ejecutar cancelación 24h

Aprobar permiso (vendedor)

Antes de que el owner pueda recomprar tokens, debes darle permiso. Esto no mueve tokens: sólo autoriza al owner para ejecutar el buyback.

DIRECCIÓN DEL OWNER

0xC9d5cdBFEB02611A0b70364DF6c43d2612a919

CANTIDAD A AUTORIZAR (TOKENS)

100

Aprobar permiso

Buyback del owner (demo)

Sólo debería usarlo la cuenta owner del contrato. El pago en dinero es externo; on-chain sólo movemos tokens y dejamos una traza mediante el evento OwnerBuyback.

DIRECCIÓN DEL VENDEDOR

0x...

CANTIDAD A RECOMPRAR (TOKENS)

Ejecutar buyback (owner)

Estado de transacciones

Aquí verás el hash, bloque y posibles errores de las operaciones que ejecutas desde esta Dapp.

Sin transacciones aún.



Imagen C — Acciones educativas y simulación de reglas.

Vista de las funciones prácticas de la DApp, donde el alumno puede experimentar con las reglas educativas: validación de precios mínimos, cancelación dentro de 24 horas, autorizaciones del vendedor y buyback simbólico por parte del owner. Todas estas acciones envían transacciones reales a la testnet de Sepolia, lo que permite ver el flujo completo: interfaz → wallet → blockchain → eventos. Incluye también un panel de “Estado de transacciones” que facilita el seguimiento del resultado de cada operación.

🔗 Enlace a la DApp (versión testnet)

<https://luisromero78.github.io/MMDV-Blockchain/projects/mmdv-wine-token/dapp/>

⚠ 11. Los problemas reales que surgieron y cómo los resolvimos

Trabajar con una DApp que interactúa con un contrato inteligente real significa encontrarse con pequeñas situaciones que, lejos de ser errores, forman parte natural del proceso. De hecho, son los momentos más valiosos del aprendizaje, porque muestran qué sucede cuando la teoría se enfrenta a la práctica. Aquí resumimos los seis escenarios más relevantes que aparecieron durante el desarrollo y las pruebas del proyecto..

11.1. Metamask no conectaba

Uno de los primeros tropiezos fue la sensación de que la DApp no detectaba Metamask. La interfaz no reaccionaba, el botón de conexión seguía igual y la dirección del usuario no aparecía. El problema era muy simple: el navegador estaba abierto en una ventana donde la extensión estaba desactivada. En cuanto se habilitó Metamask y se refrescó la página, todo fluyó con normalidad.

Aprendizaje:

En Web3, si Metamask no responde, **la DApp no existe**. Es el puente básico.



11.2. ABI no coincidía con el contrato verificado

En una de las primeras pruebas, la DApp devolvía errores raros al intentar leer el cap, el total supply o los parámetros simbólicos. La causa: estábamos usando un ABI que no correspondía exactamente a la versión del contrato verificado en Sepolia. Bastó con copiar el ABI directamente desde Etherscan para que todo comenzara a funcionar.

Aprendizaje:

Si el ABI no es exacto, la DApp está hablando un idioma distinto al del contrato.

11.3. Error “execution reverted” al ejecutar redeem ()

Apareció varias veces, incluso aunque el botón parecía estar bien conectado.

El motivo era completamente lógico: la wallet conectada **no era el owner**. El contrato rechazaba la operación porque la función está protegida.

La solución fue ajustar la interfaz para detectar esta situación y mostrar un mensaje amable explicando la restricción.

De esta forma, el alumno entiende la regla sin tener que interpretar un error técnico

11.4. Lecturas que aparecían como undefined

Durante las primeras pruebas, algunos valores como el balance o el primer registro de compra aparecían en blanco o como undefined. Esto no era un fallo del contrato, sino una cuestión temporal: la DApp estaba intentando leer datos **antes** de que Metamask confirmara la conexión o antes de que la instancia del contrato estuviera lista.

Una pequeña espera, o colocar la lectura dentro del callback correcto, resolvió la situación.

Aprendizaje:

En Web3, todo depende del orden en el que suceden las cosas.

11.5. Los retrasos de la red que daban la sensación de “no ha pasado nada”

En varias ocasiones, al ejecutar una acción como cancelar en 24h o aprobar un permiso, la interfaz tardaba unos segundos en actualizarse. El alumno pensaba que algo había fallado... pero no: la transacción estaba esperando confirmación en Sepolia. Etherscan mostraba el movimiento unos segundos después.

Por eso añadimos mensajes de estado y refrescos automáticos.



Aprendizaje:

La blockchain nunca es instantánea, siempre hay un pequeño latido entre acción y confirmación.

11.6. El clásico CORS al abrir el proyecto desde archivos locales

Este fue muy típico: abrir la DApp directamente desde un archivo local (file://) provocaba problemas de CORS en navegadores más estrictos. La DApp necesitaba un entorno mínimamente servido para permitir lecturas del contrato.

La solución fue simple: usar un servidor estático (GitHub Pages, Vercel, o incluso npx serve).

Desde ese momento, cero problemas.

Aprendizaje:

Las DApps no se ejecutan como un archivo suelto, sino como una pequeña aplicación servida.

En conjunto, estos seis puntos son un recordatorio perfecto de cómo se trabaja en Web3: con paciencia, con lógica y con comprensión del flujo wallet → contrato → red. Ningún problema fue grave; todos fueron parte del viaje, y cada uno de ellos convirtió este módulo en algo mucho más realista y formativo.

12. Conclusiones del Bloque 4



Este bloque marca un punto de madurez dentro del proyecto MMDV Wine Token. Hasta ahora habíamos visto cómo nace un token, cómo se despliega, cómo se verifica y cómo se mintea un lote. Pero aquí dimos un paso mucho más importante: convertir ese contrato en una **aplicación real**, accesible, comprensible y utilizable por cualquier persona.

En este módulo hemos construido la primera versión funcional de una DApp completa. Una aplicación capaz de conectarse a una wallet, leer datos de la blockchain en tiempo real y ejecutar acciones sobre el contrato de forma segura. El alumno ha visto cómo una web se transforma en una interfaz Web3 viva, donde cada clic tiene un impacto visible y verificable en la red.



También aprendimos que una DApp no es solo diseño visual; es experiencia. Supimos cómo estructurar la información, cómo guiar al usuario, cómo simplificar conceptos complejos y cómo presentar acciones críticas de manera clara. La función `redeem()` fue un ejemplo perfecto: un proceso técnico profundo presentado como una acción intuitiva y pedagógica.

En paralelo, este bloque mostró algo igual de valioso: la realidad del desarrollo Web3. Los retrasos, los errores, las firmas, los problemas de ABI o de red no fueron obstáculos, sino escenarios reales que cualquier desarrollador encontrará. Verlos, entenderlos y solucionarlos forma parte del aprendizaje que queríamos transmitir con este proyecto.

Al completar este bloque, el alumno ha recorrido un ciclo completo:

- Un contrato inteligente vivo en la blockchain.
- Una DApp que lo interpreta y lo hace accesible.
- Acciones reales que modifican el estado de la cadena.
- Eventos visibles que cuentan la historia de cada lote.

Ese es el auténtico poder de Web3: unir lo técnico, lo visual y lo educativo en un mismo flujo.

Con esta base, el proyecto queda listo para evolucionar. Puede crecer hacia nuevas funcionalidades, integrarse en una experiencia XRPL, incorporar dashboards más avanzados o incluso convertirse en una plataforma educativa completa sobre tokenización. Lo importante es que ahora existe un puente sólido entre el contrato y el usuario, y ese puente lo construimos aquí.

Este bloque demuestra que la blockchain no es un concepto distante, sino una herramienta que puede hacerse comprensible y cercana cuando se diseña bien. Y lo más importante: deja claro que el alumno ya no está observando desde fuera; está participando directamente en el sistema.

13. Apéndice: Código comentado

Este apéndice reúne los fragmentos más relevantes del código utilizado en la DApp, explicados de forma clara para que el alumno comprenda no solo *qué hace* cada parte, sino *por qué es necesaria*. No mostramos el archivo completo aquí —para mantener el documento limpio—, pero sí las piezas fundamentales que permiten



entender la arquitectura del proyecto. El código íntegro estará disponible en el repositorio de GitHub.

A continuación, los fragmentos más importantes:

- *Detección de Metamask y conexión inicial*

```
javascript
```

```
if (window.ethereum) {  
    provider = new ethers.providers.Web3Provider(window.ethereum);  
} else {  
    alert("Metamask no está disponible en este navegador.");  
}
```

Este bloque detecta si Metamask está instalada. Sin este puente, la DApp no puede interactuar con la blockchain, así que es el primer paso obligatorio.

- *Solicitud de conexión y obtención de la cuenta del usuario*

```
javascript
```

```
await provider.send("eth_requestAccounts", []);  
signer = provider.getSigner();  
const userAddress = await signer.getAddress();
```

Aquí se solicita al usuario permiso para conectar su wallet. A partir de este momento, la DApp puede firmar transacciones en su nombre (solo cuando él lo autorice). Es la base de todo el flujo Web3.

- *Creación de la instancia del contrato*

```
javascript
```

```
const contract = new ethers.Contract(CONTRACT_ADDRESS, ABI, signer);
```

Esta línea es el corazón de la DApp. Toma la dirección del contrato y el ABI, y crea una versión del contrato que la DApp puede “entender”. Desde aquí se pueden leer datos y ejecutar funciones.

- *Lectura del estado del token*

```
javascript
```

```
const supply = await contract.totalSupply();
const cap = await contract.cap();
const decimals = await contract.decimals();
```

Esta es la llamada clave del módulo. Aquí el owner confirma la operación en Metamask, la blockchain procesa la transacción, y la DApp espera la confirmación antes de actualizar la interfaz.

¶ 14. Apéndice 2 — Arquitectura final del proyecto

Para cerrar el módulo, es importante entender cómo se relacionan todas las piezas: la DApp, la wallet del usuario, el contrato inteligente y la blockchain. La arquitectura completa puede resumirse en tres capas que trabajan juntas y de forma sincronizada.

Capa 1 — Interfaz (Front-end)

Es la parte que ve el usuario:

- index.html, donde están los elementos visuales.
- style.css, que define el estilo.
- script.js, que contiene la lógica principal.

La interfaz no sabe nada de blockchain por sí sola. Necesita a Ethers.js para traducir sus acciones.

Capa 2 — Wallet del usuario (Metamask)

Metamask es el puente.



- Firma las transacciones.
- Gestiona la red seleccionada (Sepolia).
- Proporciona la dirección pública del usuario.
- Protege las claves privadas.

Sin Metamask, la DApp no puede modificar la blockchain.

Capa 3 — Blockchain y contrato inteligente

Aquí vive el contrato desplegado en Sepolia.

- Guarda el cap, el supply, los eventos y los batchId.
- Valida quién es el owner.
- Ejecuta operaciones como redeem() cuando se cumplen los permisos.



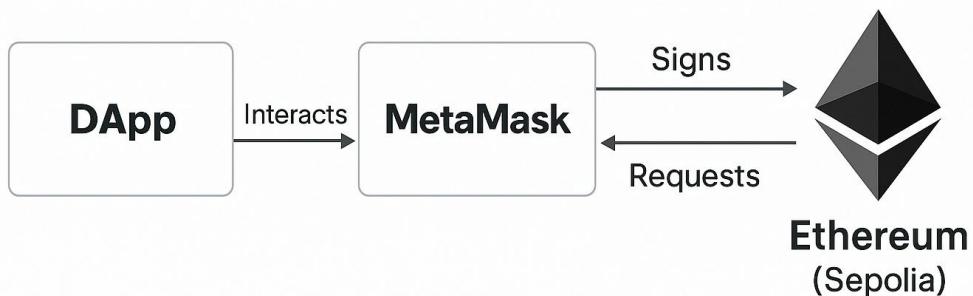
Cada acción deja un registro público visible en Etherscan.

Flujo completo

1. El flujo general del sistema es este:
2. El usuario abre la DApp.
3. La interfaz pide permiso a Metamask.
4. Metamask devuelve la dirección del usuario.
5. La DApp crea la instancia del contrato usando Ethers.js.
6. La interfaz lee datos (supply, cap, owner...).
7. El usuario ejecuta una acción (p. ej., redeem).
8. Metamask solicita la firma.
9. La transacción viaja a Sepolia.
10. El contrato la procesa y emite un evento.
11. La DApp refresca los datos y muestra el resultado.

Este flujo convierte una web convencional en un sistema vivo conectado a una blockchain real.

Diagrama recomendado





💡 Qué veremos en el Bloque 5

Después de construir y entender la DApp en profundidad, el siguiente paso natural es elevar el proyecto a un nivel más avanzado. En el **Bloque 5** daremos un salto desde una DApp funcional a un **ecosistema completo**, donde veremos cómo ampliar, profesionalizar y escalar nuestro token educativo.

En este bloque exploraremos:

- ✓ **Nuevos roles y permisos en contratos inteligentes.** Cómo diseñar contratos con múltiples niveles de acceso, permisos flexibles y operaciones condicionadas para simular sistemas reales (vendedores, administradores, holders con privilegios...).
- ✓ **Lógica avanzada en la blockchain.** Implementaremos reglas más ricas, como períodos dinámicos, límites variables, cálculos internos, bonus progresivos, penalizaciones y validaciones más complejas.
- ✓ **Integración de nuevas funciones on-chain.** Simularemos mecanismos como recompras (buybacks), cancelaciones reales, autorizaciones de terceros, simuladores de precio interno y herramientas para visualizar decisiones en la cadena.
- ✓ **Nuevos paneles dentro de la DApp.** Mejoraremos la interfaz para incluir módulos avanzados:
 - estado del holder,
 - simuladores de precio,
 - visualización de bonus,
 - acciones que requieren varias firmas.
- ✓ **Trazabilidad y análisis de datos desde los eventos.** Aprenderemos a transformar los eventos emitidos por el contrato en información útil para el usuario, abriendo la puerta a dashboards educativos y herramientas de análisis.
- ✓ **Preparación para el gran salto: XRPL.** Cerramos el bloque explorando cómo migrar conceptos de Ethereum a XRP Ledger, preparando el terreno para el curso final de tokenización avanzada.

“En este bloque no solo construimos una DApp: construimos el puente entre la idea y la realidad. A partir de aquí, el contrato ya no vive solo en la cadena... vive en las manos del usuario.”