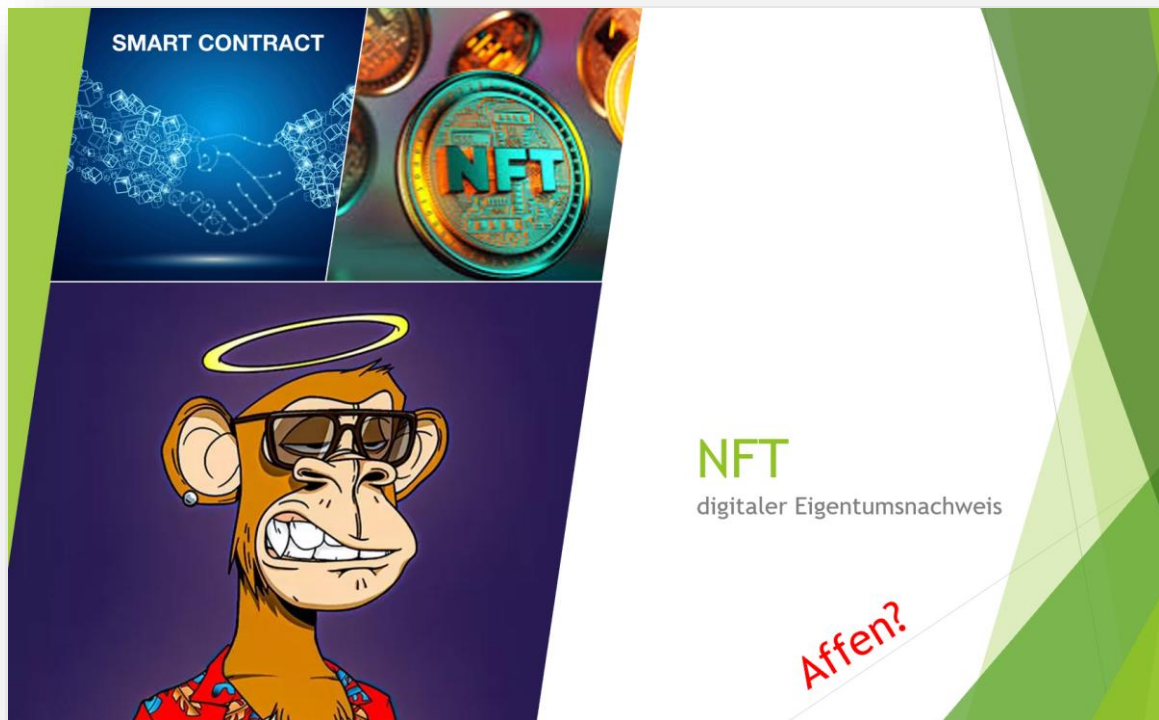


DLT-UseCase

- NFT -



Inhalt

1	NFT – Non Fungible Token	2
1.1	Was ist ein NFT?	2
1.2	Wie entsteht eine NFT?	3
2	Praxis: Erstellung einer lokalen NFT-dApp.....	4
2.1	Aufgabe	4
2.2	Vorbereitung	4
2.3	Setup	5
2.4	Start.....	6
2.5	NFT minten und transferieren	7
3	Praxis: Veröffentlichen eines NFT auf Sepolia	8
3.1	Smart Contract	8
3.2	Frontend.....	9
3.3	NFT auf OpenSea anzeigen.....	10

1 NFT – Non Fungible Token

1.1 Was ist ein NFT?

Ein **NFT** (Non-Fungible Token) ist ein digitaler Vermögenswert, der die Eigentümerschaft und Echtheit eines einzigartigen Objekts oder einer Datei auf einer Blockchain dokumentiert. Der Begriff "non-fungible" bedeutet, dass ein NFT nicht austauschbar oder gleichwertig mit einem anderen NFT ist, im Gegensatz zu Kryptowährungen wie Bitcoin oder Ethereum, die fungibel sind (austauschbar und von gleichem Wert).

Anwendungsbeispiele

1. **Digitale Kunst:** Künstler können ihre Werke als NFTs verkaufen, um Eigentum nachzuweisen und neue Einnahmequellen zu schaffen.
2. **Sammlerstücke:** Virtuelle Sammelkarten, Avatare oder Musikstücke können als NFTs gehandelt werden.
3. **Virtuelle Welten und Spiele:** Spieler können digitale Gegenstände wie Grundstücke, Charaktere oder Ausrüstungen als NFTs kaufen und besitzen.
4. **Ereignistickets:** Ein NFT kann als fälschungssicheres Ticket für Veranstaltungen dienen.

Vorgehensweise:

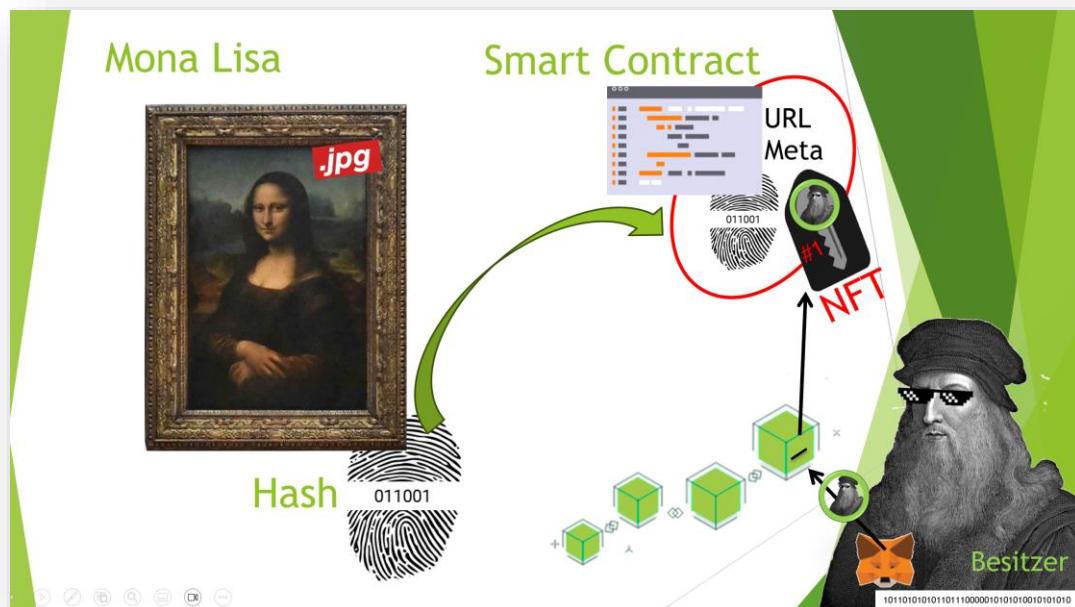
- Ein Künstler erstellt ein Kunstwerk und lädt es auf eine Plattform hoch, die NFTs generiert.
- Die Plattform erstellt den NFT, speichert die Metadaten und weist ihm eine eindeutige Kennung (Hash) zu.
- Der NFT wird auf einem Marktplatz angeboten, und ein Käufer kann ihn erwerben.
- Nach dem Kauf wird die Blockchain aktualisiert, um den neuen Besitzer zu registrieren.
- Der Käufer kann den NFT in seiner Wallet speichern oder ihn weiterverkaufen.

Eigenschaften von NFTs

1. **Einzigartigkeit:** Jedes NFT ist ein Unikat oder Teil einer begrenzten Serie. Es wird durch einen einzigartigen Identifikator repräsentiert, der sicherstellt, dass es sich um ein einmaliges Objekt handelt.
2. **Unveränderlichkeit:** Informationen über ein NFT, einschließlich des Besitzers und der Historie, werden in einer sicheren, dezentralen Datenbank gespeichert, die nicht manipuliert werden kann.
3. **Nachverfolgbarkeit:** Die gesamte Transaktionshistorie eines NFTs ist öffentlich einsehbar und nachvollziehbar.

1.2 Wie entsteht eine NFT?

Angenommen, Leonardo da Vinci möchte sein berühmtes Gemälde **Mona Lisa** als digitales Kunstwerk in der modernen Welt präsentieren und dafür ein **NFT** erstellen. Der Erstellungsprozess läuft folgendermaßen ab:



1. Die digitale Version der Mona Lisa wird erstellt

Leonardo erstellt eine hochauflösende digitale Kopie der Mona Lisa, zum Beispiel als Bilddatei im JPG-Format. Diese Datei ist die Grundlage des NFTs und repräsentiert das digitale Kunstwerk.

2. Erstellung des NFTs auf der Blockchain

Leonardo nutzt eine NFT-Plattform (z. B. OpenSea), die mit einer **Blockchain** wie Ethereum arbeitet. Die NFT-Plattform führt folgende Schritte durch:

a) Upload der Datei

Die Datei wird auf einem Webserver oder noch besser auf einem dezentralen Speicher wie IPFS hochgeladen und damit sicher zugänglich gemacht.

b) Smart Contract

Ein Smart Contract wird erstellt. Dieser automatisierte Vertrag beinhaltet:

- Einen **URL-Link** zur digitalen Datei.
- **Metadaten**, wie den Titel, Künstler, Beschreibung und das Erstellungsjahr
- Einen **Hash** des Bildes (und ggf. der Metadaten) als kryptografischer Fingerabdruck, der sicherstellt, dass der NFT einzigartig ist und niemand ihn fälschen kann.
- Optionale **Lizenzgebühren**: Wenn der NFT später weiterverkauft wird, erhält Leonardo z. B. 10 % des Verkaufspreises.

c) Eigentum:

Leonardo da Vinci wird mit Hilfe seiner Wallet-Adresse als erster Besitzer des NFTs eingetragen.

3. Speicherung in der Blockchain

Der Smart Contract mit allen Daten, einschließlich des Hashes, der Metadaten und des Eigentümers, wird in die Blockchain geschrieben. Diese Speicherung macht den NFT unveränderlich und öffentlich nachvollziehbar.

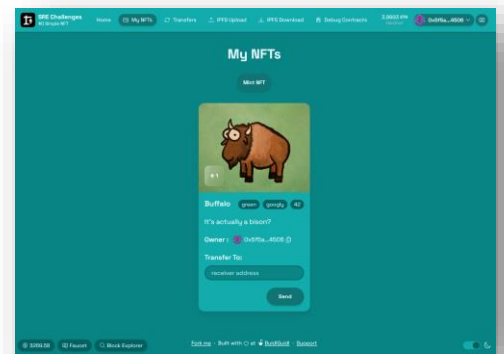
2 Praxis: Erstellung einer lokalen NFT-dApp

2.1 Aufgabe

Sie kompilieren und implementieren Ihren ersten Smart Contract.

Schließlich implementieren Sie ein NFT in einem öffentlichen Netzwerk, um es mit Freunden zu teilen!

Das Endprodukt ist eine App, mit der Benutzer NFTs kaufen und übertragen können. Stellen Sie Ihre Verträge in einem Testnetz bereit, erstellen Sie dann Ihre App und laden Sie sie auf einen öffentlichen Webserver hoch.



2.2 Vorbereitung

Bitte installieren Sie die folgenden Softwarepakete (Windows oder Linux)

Node (>= v18.18):

<https://nodejs.org/en/download/>

Node.js® ist eine kostenlose, plattformübergreifende Open-Source-JavaScript-Laufzeitumgebung, mit der Entwickler Server, Webanwendungen, Befehlszeilentools und Skripte erstellen können.

Git:

<https://git-scm.com/downloads>

Git ist ein kostenloses und quelloffenes verteiltes Versionskontrollsystem, das für die schnelle und effiziente Abwicklung kleiner bis sehr großer Projekte konzipiert ist.

2.4 Start

(1) Starten Sie eine lokale Blockchain:

Geben Sie folgenden Befehl in ihrem geöffneten Terminalfenster ein:

```
yarn chain
```

Der Befehl startet das lokale Ethereum-Entwicklungsnetzwerk „Hardhat“. Es wird verwendet, um Smart Contracts zu testen, Transaktionen zu simulieren und das Zusammenspiel zwischen Frontend und Blockchain zu entwickeln, bevor die Anwendung auf ein öffentliches Netzwerk übertragen wird. Da die Blockchain lokal läuft, können Aktionen wie das Debugging, das Überprüfen von Transaktionen und das Testen von Smart Contracts schnell und ohne Kosten durchgeführt werden. Das Netzwerk stellt mehrere Konten mit Test-Ether zur Verfügung, die du für das Testen verwenden kannst. Das Skript chain, das durch den Befehl yarn chain aufgerufen wird, befindet sich in der Datei package.json.

(2) Veröffentlichen Sie einen Smart Contract auf der Blockchain:

Geben Sie folgenden Befehl in einem zweiten Terminalfenster ein:

```
cd challenge-0-simple-nft  
yarn deploy
```

Der Befehl startet ein Skript, das die Bereitstellung von Smart Contracts auf einer Blockchain automatisiert. Es kompiliert die Contracts, verbindet sich mit dem gewünschten Netzwerk und lädt die Contracts hoch. Dieses Skript ist essenziell für die Ethereum-Entwicklung, da es den Deployment-Prozess standardisiert und vereinfacht.

(3) Starten Sie ihr Web-Frontend:

Geben Sie folgenden Befehl in einem dritten Terminalfenster ein:

```
cd challenge-0-simple-nft  
yarn start
```

Der Befehl startet die Frontend-Anwendung. Diese DApp (dezentrale Anwendung) läuft auf einem Entwicklungsserver und stellt eine grafische Benutzeroberfläche bereit, mit der Nutzer Smart Contracts nutzen können. Dabei wird zunächst der React-Entwicklungsserver gestartet, der die Webanwendung lokal ausführt. Die Anwendung verbindet sich mit der lokalen Blockchain und ermöglicht die Interaktion mit den bereitgestellten NFTs.

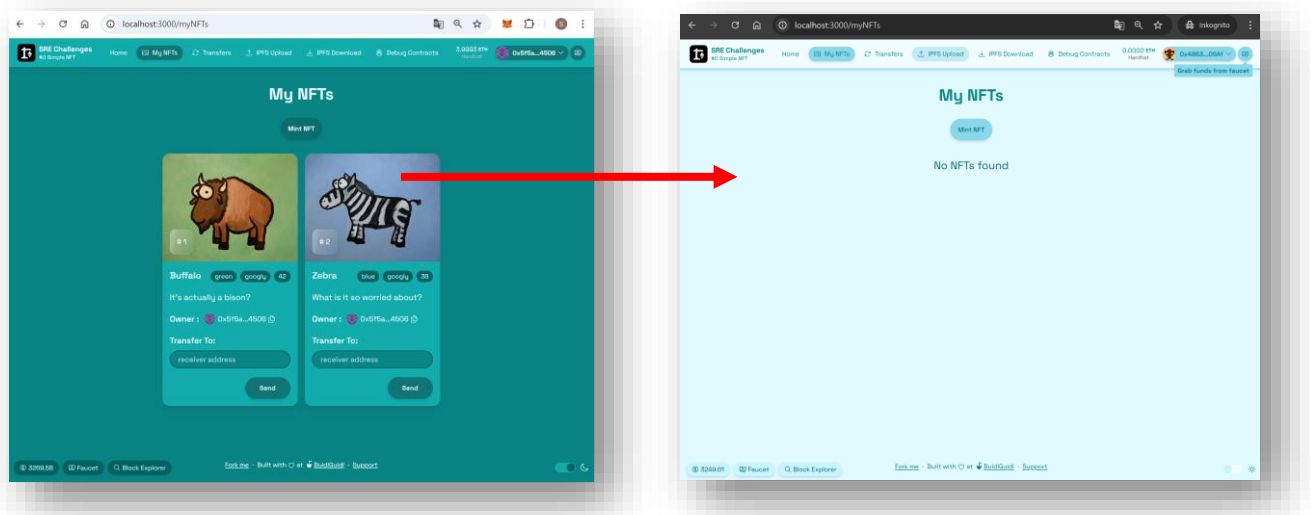
(4) Browser aufrufen:

Die Anwendung ist dann üblicherweise unter <http://localhost:3000> im Browser zugänglich.

2.5 NFT minten und transferieren

Minten Sie zwei NFTs.

Öffnen Sie ein zweites Browserfenster im Inkognito-Modus und übertragen ein NFT an die neue Wallet.



3 Praxis: Veröffentlichen eines NFT auf Sepolia

3.1 Smart Contract

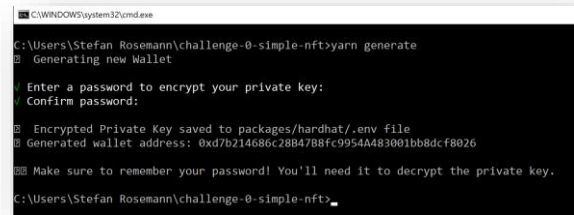
(1) Eine Deployer-Adresse erzeugen:

Erzeugen Sie ein neues Ethereum-Konto:

```
yarn generate
```

Um einen Smart Contract zu veröffentlichen, muss die Transaktion mit einem privaten Schlüssel signiert werden. Das bedeutet, dass ein Ethereum-Konto erforderlich ist. Während auf einer lokalen Blockchain ein vorgegebenes Standardkonto verwendet werden kann, das bei jeder Installation verfügbar ist, muss auf der öffentlichen Blockchain Sepolia ein einzigartiges Konto genutzt werden.

Da der Deploy-Vorgang über Skripte gesteuert wird, wäre es denkbar, den privaten Schlüssel aus MetaMask in eine Datei einzutragen. Dies birgt jedoch erhebliche Sicherheitsrisiken und sollte unbedingt vermieden werden. Stattdessen erstellen wir mit dem oben genannten Befehl ein neues Ethereum-Konto, wobei der private Schlüssel passwortgeschützt gespeichert und automatisch an der richtigen Stelle abgelegt wird.



```
C:\WINDOWS\system32\cmd.exe
C:\Users\Stefan Rosemann\challenge-0-simple-nft>yarn generate
Generating new Wallet
Enter a password to encrypt your private key:
Confirm password:
Encrypted Private Key saved to packages/hardhat/.env file
Generated wallet address: 0xd7b214686c2884788fc9954A483001bb8dcf8026
Make sure to remember your password! You'll need it to decrypt the private key.
C:\Users\Stefan Rosemann\challenge-0-simple-nft>
```

(2) Öffentliche Adresse und Kontostand einsehen:

Zeigen Sie Ihre Kontoinformationen an:

```
yarn account
```

Der Befehl die öffentliche Adresse und den Kontostand im Terminal aus.



```
C:\WINDOWS\system32\cmd.exe
C:\Users\Stefan Rosemann\challenge-0-simple-nft>yarn account
Enter your password to decrypt the private key:
Public address: 0xec774FbaA343F94FD3e8B651c11c0a019CF346A2
Can't connect to network localhost
-- mainnet --
balance: 0
nonce: 0
-- sepolia --
balance: 0.01461275998483586
nonce: 1
-- arbitrum --
```

(3) Konto mit 0,01 Sepolia aufladen:

Kopieren Sie die angezeigte öffentliche Adresse und überweisen Sie (mit MetaMask) mind. **0,01 Sepolia** auf das Konto. Überprüfen Sie den Kontostand erneut mit „yarn account“.

(4) Smart Contract veröffentlichen:

Veröffentlichen Sie den SmartContract auf dem Sepolia-Testnetzwerk:

```
yarn deploy --network sepolia
```

3.2 Frontend

(1) Frontend-Konfiguration anpassen:

Verbinden Sie das Frontend mit dem Sepolia-Netzwerk, statt mit der lokal Hardhat-Chain. Ändern Sie in der Datei „**packages/nextjs/scaffold.config.ts**“ den Eintrag **targetNetworks** folgendermaßen:

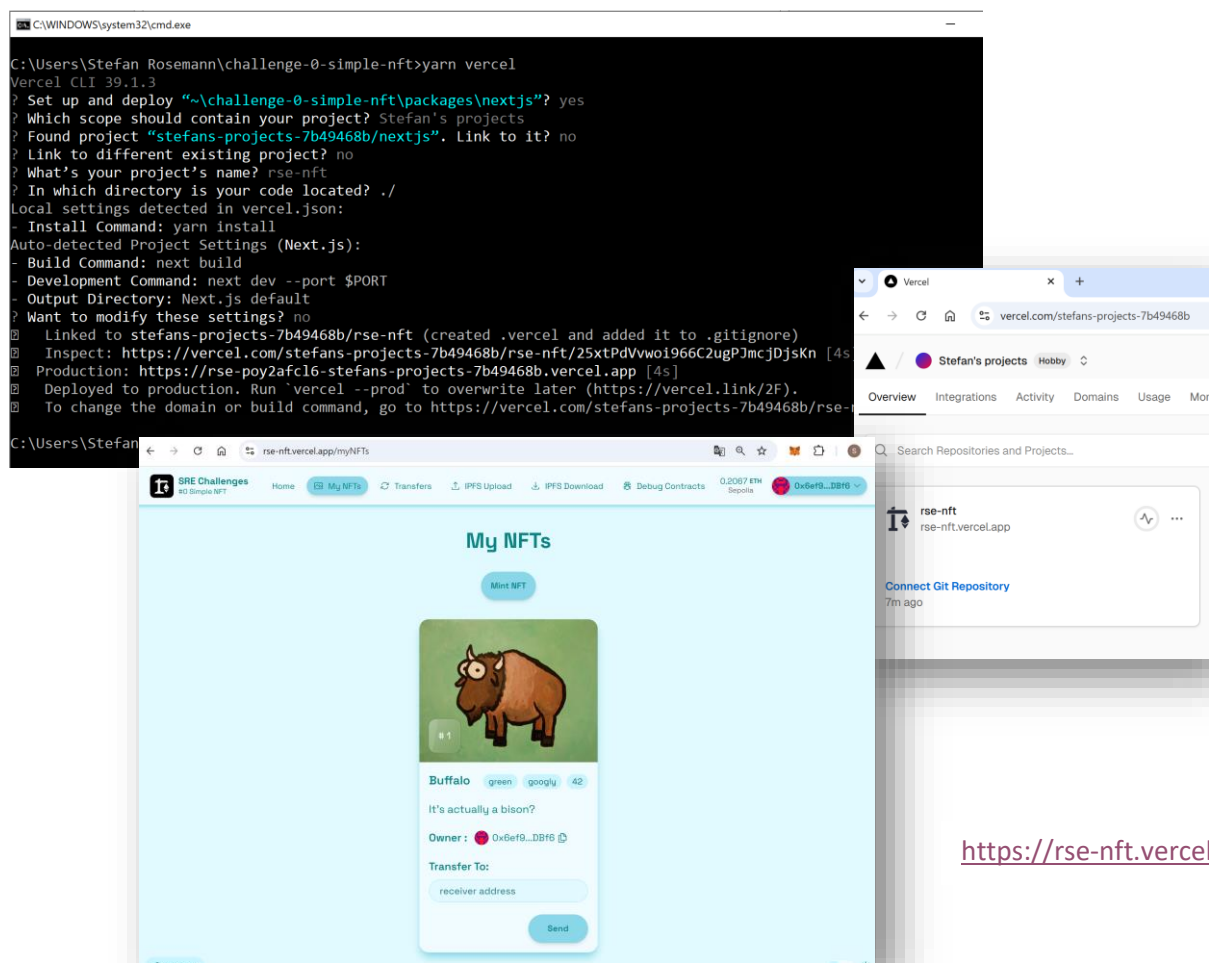
```
const scaffoldConfig = {  
  // The networks on which your DApp is live  
  targetNetworks: [chains.sepolia],  
}
```

(2) Smart Contract veröffentlichen:

Veröffentlichen Sie ihr Frontend auf der Plattform „Vercel“. Für die Erstanmeldung bietet sich die Authentifizierung per Email an.

```
yarn vercel
```

Vercel ist eine cloudbasierte Plattform für die Entwicklung, Bereitstellung und Skalierung von Webanwendungen. Sie ist besonders auf Frontend-Projekte spezialisiert und wird häufig für Frameworks wie Next.js, React, Vue.js, und ähnliche verwendet. Vercel bietet eine einfache Möglichkeit, moderne Webanwendungen zu hosten und nahtlos mit Entwicklertools zu integrieren.



<https://rse-nft.vercel.app>

3.3 NFT auf OpenSea anzeigen

