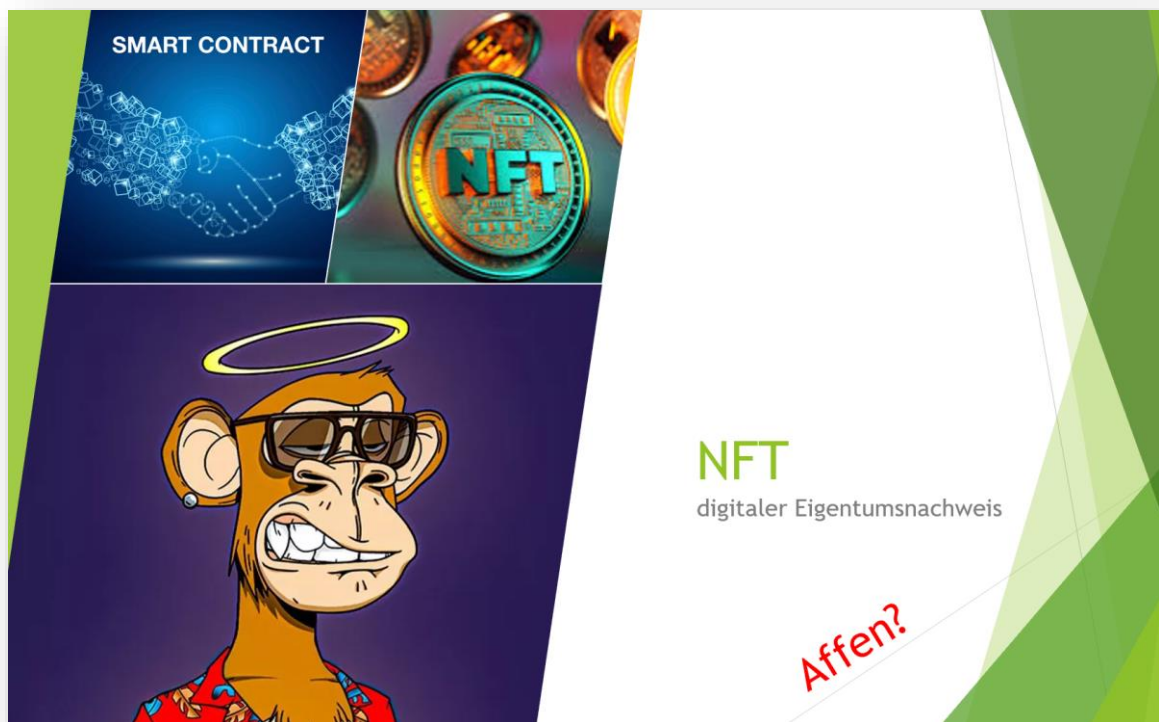


Blockchain

- 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	Setup und Start.....	5
2.3	NFT minten und transferieren	6
2.4	Minten des eigenen NFTs	7
2.5	SmartContract untersuchen	9
2.7	Technologiekonzept	10
3	Praxis: Veröffentlichen der NFT-dApp.....	11
3.1	Smart Contract => Sepolia	11
3.2	Frontend => Internet	12
4	NFT auf OpenSea anzeigen.....	16
5	Speed Run Ethereum.....	17

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:

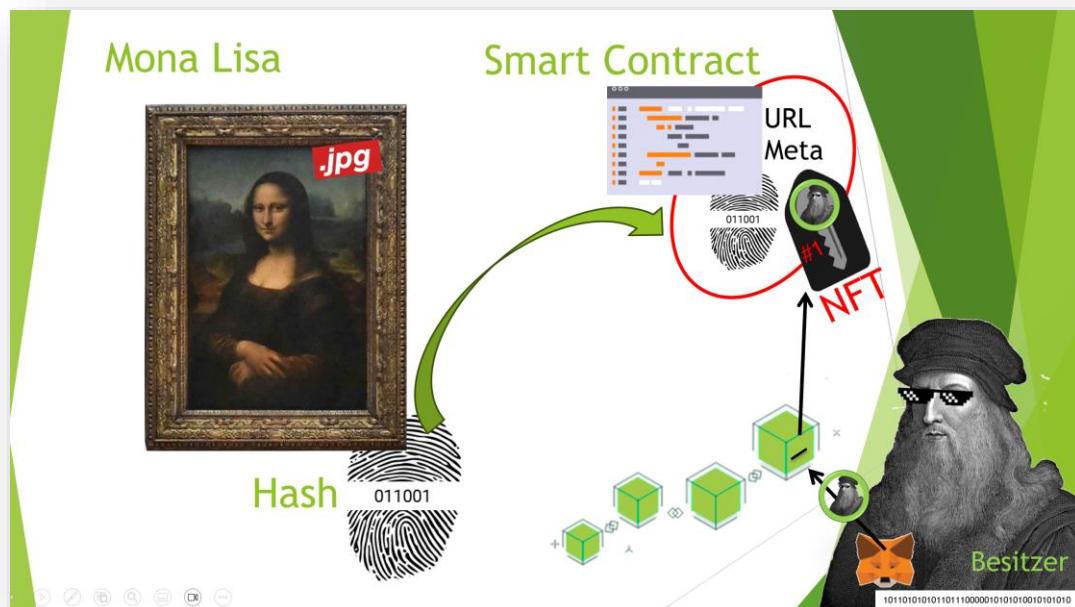
- Ein Künstler erstellt ein Kunstwerk und lädt es auf eine Plattform hoch, die NFTs generiert.
- Die Plattform erstellt den NFT, speichert die Metadaten und weist ihm eine eindeutige Kennung (Hash) zu.
- Der NFT wird auf einem Marktplatz angeboten, und ein Käufer kann ihn erwerben.
- Nach dem Kauf wird 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.

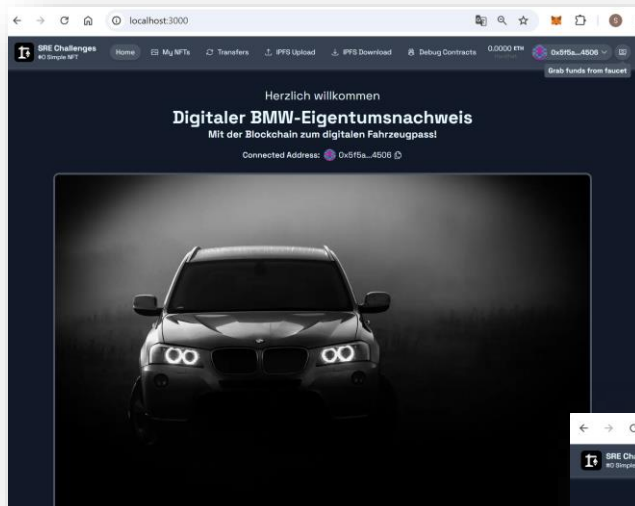
3. Speicherung in der Blockchain

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

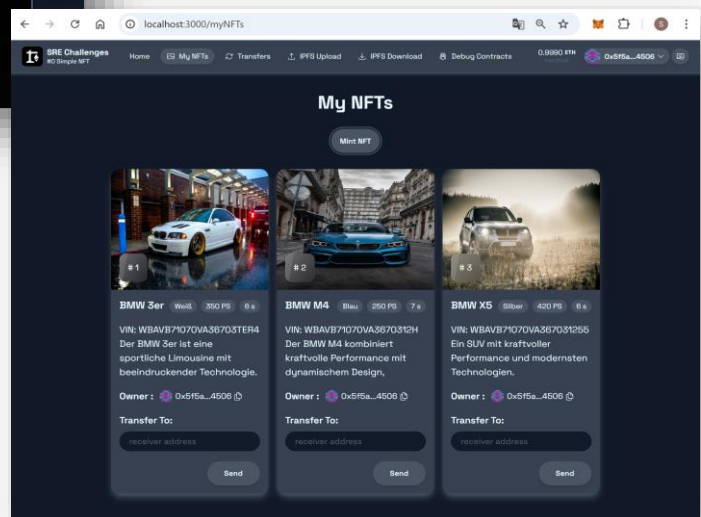
2.1 Aufgabe

Erstellen Sie einen **blockchainbasierten digitalen Eigentumsnachweis** für Fahrzeuge der Marke BMW!



BMW übergibt jedem Kunden beim Neukauf einen **digitalen Eigentumsnachweis**, in Form eines fahrzeugbezogenen NFTs. Mit dem NFT ist nicht nur ein **digitales Serviceheft** verbunden, sondern können zukünftig auch **Park- oder Tankvorgänge** abgerechnet werden. Weiterhin kann der Besitzer eine digitale Kopie seines Fahrzeugs im „Metaverse“ benutzen.

Für jedes produzierte Fahrzeug wird ein **NFT gemintet**, in dem die Fahrzeugidentifikationsnummer (VIN) und die wichtigsten Attribute des Fahrzeugs sowie die URL eines Fotos aufgeführt sind.



2.2 Setup und Start

(1) Kopieren und installieren Sie die Projektdaten:

Starten Sie eine Eingabeaufforderung und geben Sie die nachfolgend erläuterten Befehle ein!

```
C:\Users\Benutzername> cd blockchain  
C:\Users\Benutzername\blockchain> npx create-eth@0.1.0 -e challenge-0-simple-nft challenge-0-simple-nft  
C:\Users\Benutzername\blockchain> blockchainerkurs\03-NFT\challenge-0-simple-nft-update.bat  
C:\Users\Benutzername\blockchain> cd challenge-0-simple-nft
```

Der Befehl „npx“ führt ein npm-Paket aus, ohne es zuerst dauerhaft zu installieren.

Das Paket „create-eth@0.1.0“ wird temporär heruntergeladen und aus diesem Paket wird das Projekt „challenge-0-simple-nft“ in das gleichnamige Verzeichnis installiert.

Die Batchdatei „challenge-0-simple-nft-update.bat“ kopiert einige Dateien aus den Kursunterlagen in das Projekt, um das Layout auf das BMW-Beispiel anzupassen.

(2) Starten Sie eine lokale Blockchain:

Geben Sie folgenden Befehl in Ihrem geöffneten Terminalfenster ein:

```
C:\Users\Benutzername\blockchain\challenge-0-simple-nft > 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.

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

Geben Sie folgende Befehle in einem zweiten Terminalfenster ein:

```
C:\Users\Benutzername\blockchain\challenge-0-simple-nft > yarn deploy  
C:\Users\Benutzername\blockchain\challenge-0-simple-nft > yarn start
```

Der Befehl „yarn deploy“ 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.

Der Befehl „yarn start“ startet die Frontend-Anwendung. Diese App stellt eine grafische Benutzeroberfläche bereit, um Smart Contracts zu nutzen. 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 unter <http://localhost:3000> im Browser zugänglich.

Der erste Aufruf einen Moment dauern, das die Seiten zur Laufzeit kompiliert werden.

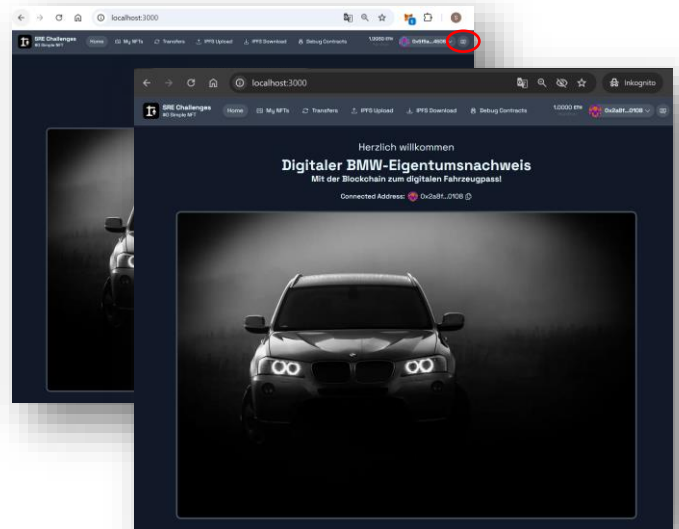
2.3 NFT minten und transferieren

In dieser Übung „minten“ Sie ein NFT und transferieren es an eine weitere Wallet.

Wenn Sie die App starten, wird automatisch eine sogenannte „Burner-Wallet“ erzeugt und mit der Webseite verbunden. Die Wallet ist anfänglich leer und kann über das „Faucet“ oben rechts gefüllt werden.

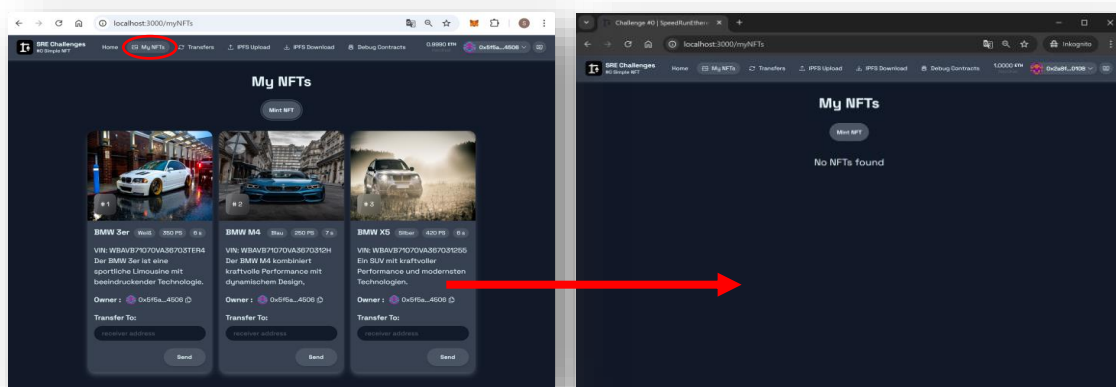
Starten Sie ein weiteres Browserfenster im „Inkognito-Modus“. In diesem Fenster wird eine neue unterschiedliche „Burner-Wallet“ verbunden.

Mit der Oberfläche können Sie Ether zwischen den Wallets transferieren.



Im Menüpunkt „My NFTs“ können NFTs gemintet werden.

Dieser Vorgang würde in der echten BMW-Anwendung automatisch erfolgen, sobald ein neues Auto vom Band rollt.



Minten Sie mindestens drei NFTs!

Wie viel Ether kostet ein Mintvorgang? _____

Der Smart Contract berechnet BMW keine Kaufgebühr für das NFT.

Warum wird die Wallet beim Minten trotzdem mit einem Betrag belastet?

Übertragen Sie ein NFT in das Inkognito-Fenster!

Wie viel Ether kostet die Transaktion? _____

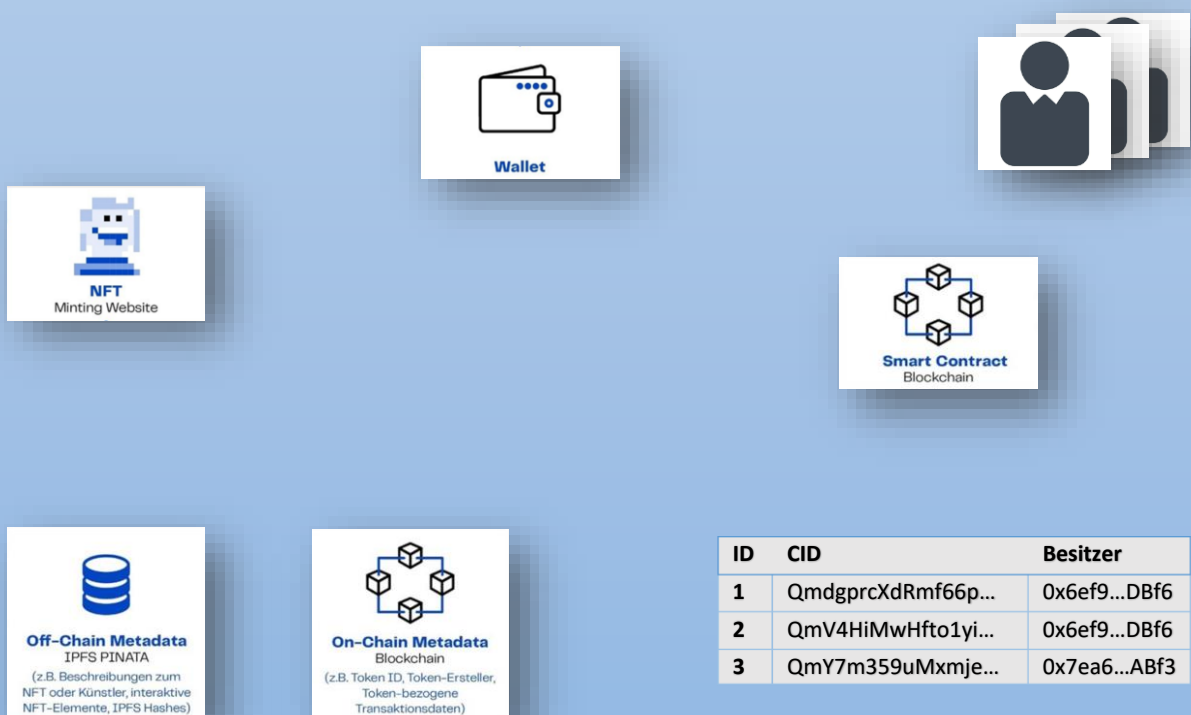
2.4 Minten des eigenen NFTs

Beim Minten eines NFTs werden mehrere technische Prozesse ausgelöst, die sowohl den Smart Contract als auch die Blockchain einbeziehen. Dabei werden folgende Schritte ausgeführt:

- Als Benutzerschnittstelle zum Minten dient häufig eine **Webseite**.
- Damit die Mint-Transaktion signiert und das NFT einer öffentlichen Adresse zugeordnet werden kann, muss eine **Wallet** verbunden mit der Webseite verbunden werden.
- Durch das Drücken des Mint-Buttons werden die Metadaten des NFTs (z.B. Bild-URL) auf dem dezentralen Speichernetzwerk **IPFS** abgelegt. Als Datenformat wird in der Regel JSON verwendet. Das Netzwerk gibt den Hashwert des JSON-Strings zurück, den sogenannten „**Content Identifier**“ (**CID**), der als eindeutiger Zugriffsschlüssel für den hochgeladenen Inhalt gilt.
- Anschließend wird eine Transaktion ausgelöst, bei der die Funktion **mintItem(CID, Ethereum-Adresse)** des Smart Contracts aufgerufen wird. Als Parameter werden die CID und die Adresse des neuen NFT-Besitzers übergeben.
- Der Smart Contract erzeugt daraufhin eine **Token-ID** (z.B. 1, 2, 3, 4, ...) und speichert hierzu den übergebenen CID und die Ethereum-Adresse ab. So entsteht im Smart Contract eine Tabelle, die alle bereits erzeugten Token-IDs mit den zugehörigen CIDs und den Besitzer-Adressen enthält.
- Da die Transaktion unveränderlich auf der **Blockchain gespeichert** wird, ist dieses NFT nun fest mit dem CID, also mit den hochgeladenen IPFS-Daten verknüpft. Die hinterlegten Daten können ab jetzt nicht mehr verändert oder manipuliert werden.
- Alle Mint-Transaktionen sind einsehbar. Der Smart Contract gibt (über einen https-Call) kostenlos Auskunft über die geminteten NFTs, deren CIDs und Besitzer

Aufgabe:

Verbinden Sie die Elemente mit beschrifteten Pfeilen gemäß der obigen Beschreibung!



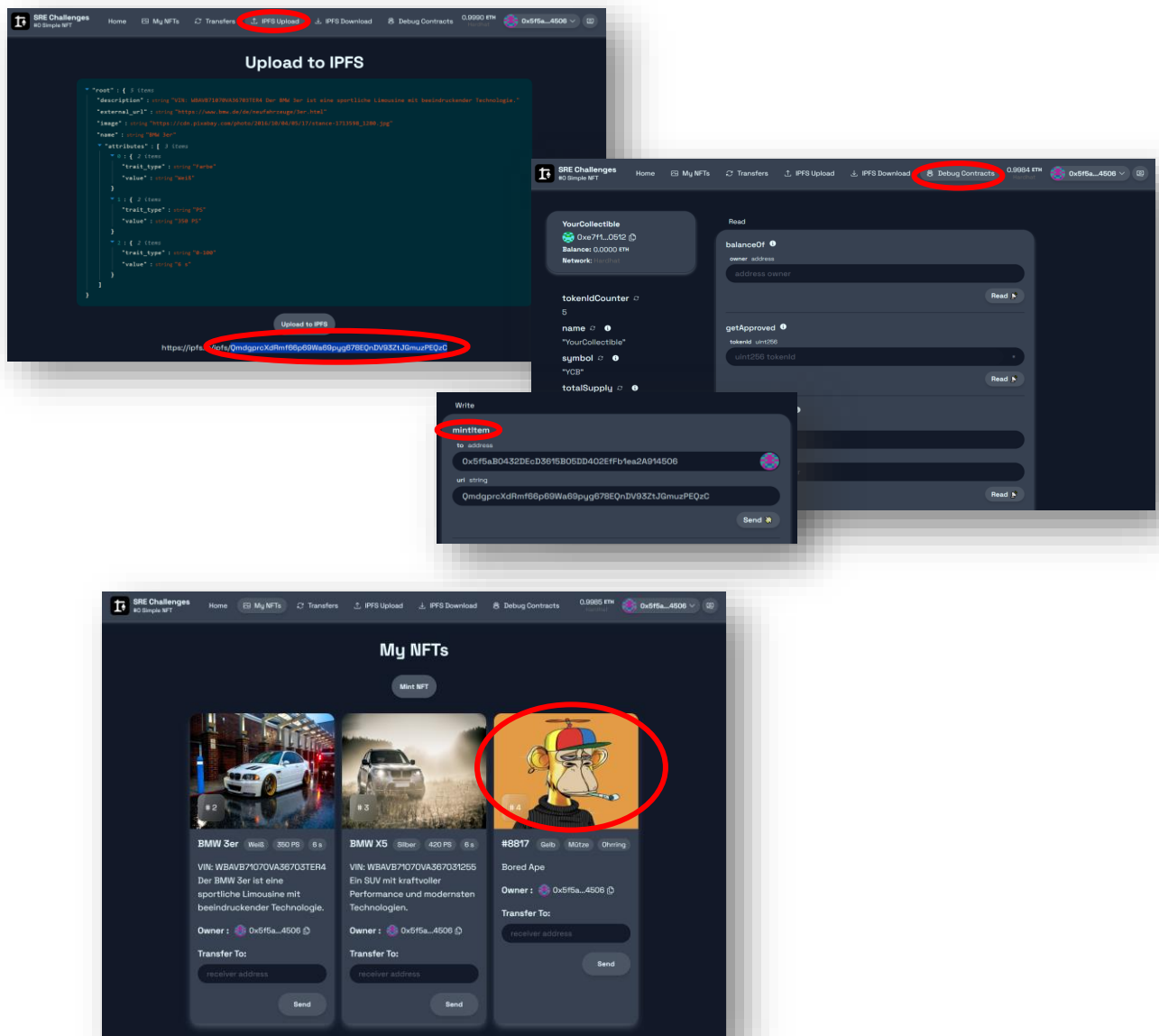
Das Minten eines NFT kann laut ERC-721 nur vom Besitzer des Smart Contracts direkt ausgeführt (oder delegiert) werden. In dem hier gezeigten Beispiel ist der Vertrag jedoch so angelegt, dass jeder beliebige Nutzer minten kann.

Aufgabe:

Minten Sie Ihr persönliches NFT!

Führen Sie hierzu folgende Schritte aus:

1. Ändern Sie zunächst im Menüpunkt „**IPFS Upload**“ die vorgegebenen NFT-Daten. Sie können z.B. die URL eines beliebigen Bildes angeben. Senden Sie die Daten ab und kopieren Sie die angezeigte **CID**-Ziffernfolge.
2. Rufen Sie unter dem Menüpunkt „**Debug Contracts**“ die Funktion **mintItem** auf. Tragen Sie die kopierte CID in das Eingabefeld **URI** (Uniform Resource Identifier) ein. Außerdem die Ethereum-Adresse aus der Burner-Wallet (oben rechts) in das Eingabefeld **to**. Senden Sie die Transaktion ab.
3. Im Menüpunkt „**My NFTs**“ sollte nun Ihr eigenes NFT erscheinen.

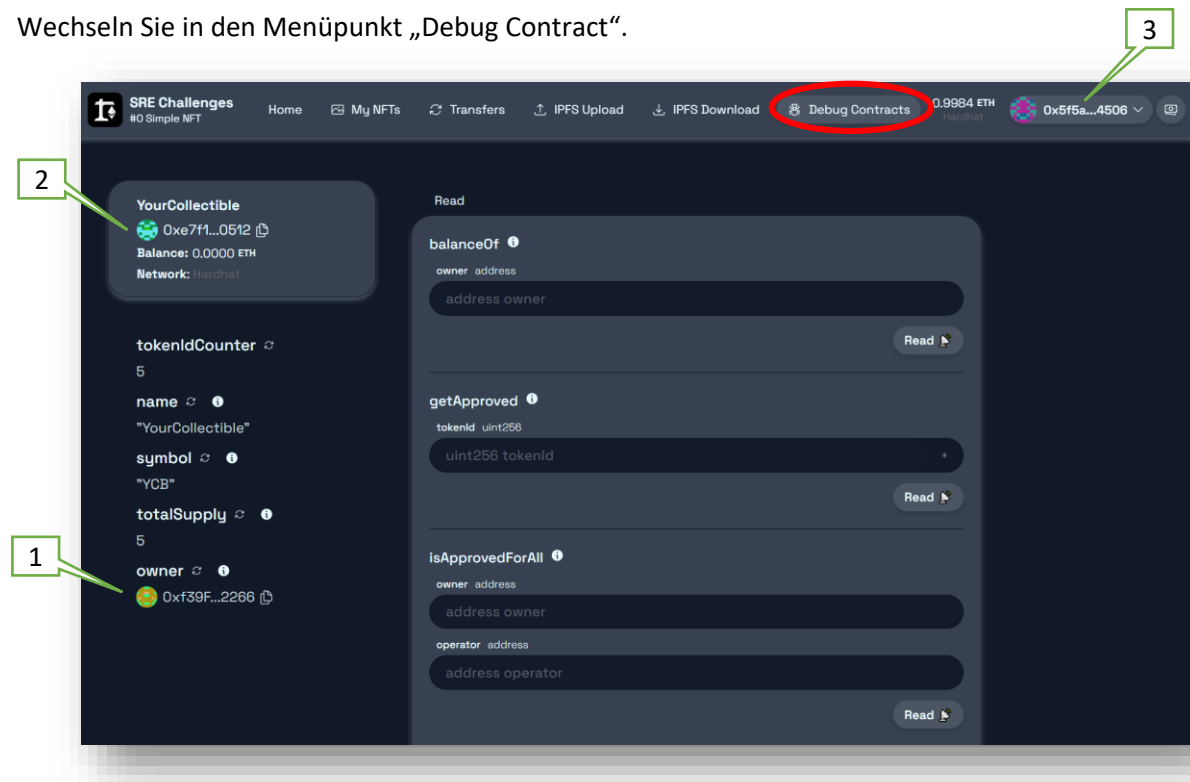


The screenshots illustrate the process of minting an NFT. The first image shows the 'Upload to IPFS' screen where metadata is defined and an IPFS URL is generated. The second image shows the 'Debug Contracts' screen where the 'mintItem' function is used to mint an NFT by providing a recipient address and the IPFS URI. The third image shows the 'My NFTs' screen where the newly minted NFT, a 'Bored Ape', is displayed alongside other NFTs.

2.5 SmartContract untersuchen

Ein Smart Contract zum Minten von NFTs (ERC-721) bietet standardmäßig eine ganze Reihe weiterer Funktionen an. Diese sollen im Weiteren untersucht werden.

Wechseln Sie in den Menüpunkt „Debug Contract“.



In der obigen Maske sind **drei Adressen** aufgeführt. Erläutern Sie deren Funktion!

Adresse 1: _____

Adresse 2: _____

Adresse 3: _____

Probieren Sie folgende **Funktionen** des Smart Contracts aus und erläutern Sie kurz deren Aufgabe:

balanceOf: _____

ownerOf: _____

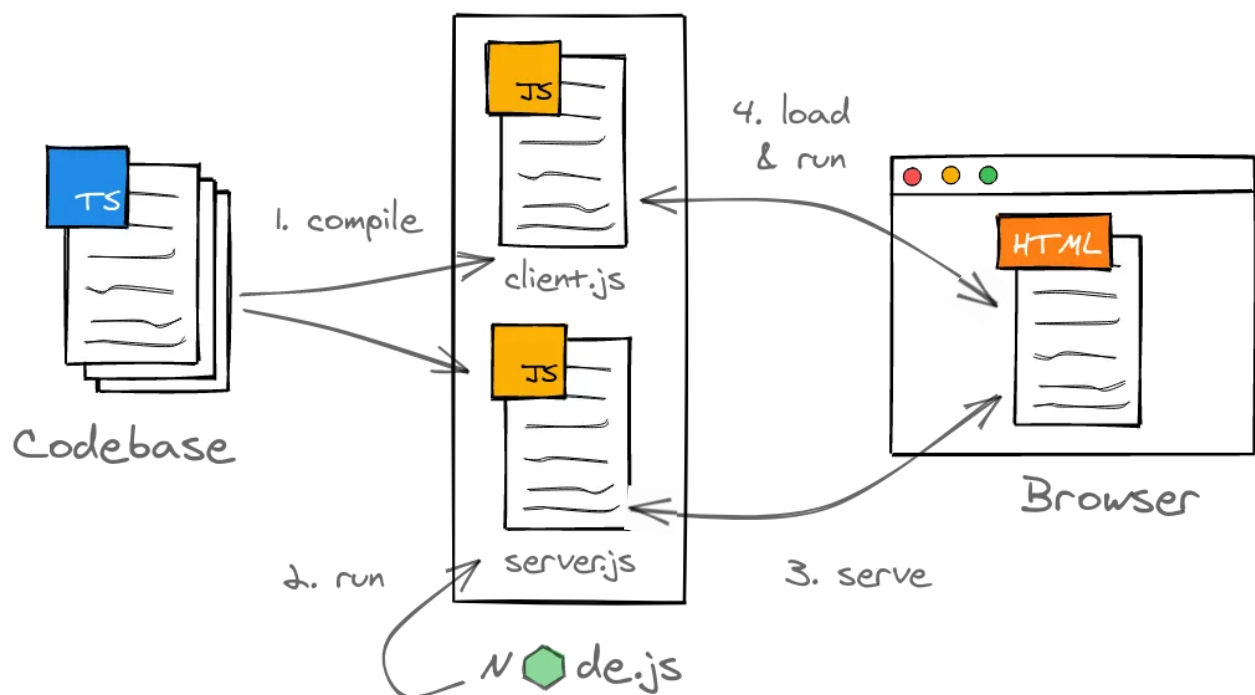
tokenURI: _____

2.7 Technologiekonzept

Die folgende Skizze zeigt, wie am Ende alles zusammenpasst.

Unser Quellcode besteht aus **TypeScript**-Dateien, die wir in zwei separate JavaScript-„Bundles“ kompilieren – im Grunde einzelne Dateien, die eine Anwendung darstellen. TypeScript ist eine Programmiersprache, die auf **JavaScript** basiert und zusätzliche Funktionen wie statische Typisierung und erweiterte Entwicklungswerkzeuge bietet. Es wird als eine „strikte Superset-Sprache von JavaScript“ bezeichnet, was bedeutet, dass jedes gültige JavaScript-Programm auch ein gültiges TypeScript-Programm ist. TypeScript wurde von Microsoft entwickelt und ist Open Source.

Eine Anwendung ist unser Server, der in Node.js läuft und auf HTTP-Anfragen mit serverseitig gerenderten HTML-Dokumenten antwortet. Die andere Anwendung läuft im Browser eines Benutzers und erweitert diese HTML-Dokumente mit Interaktivität über React. **Next.js** ist ein Open-Source **React**-Framework, das auf Node.js basiert. Es bietet eine Reihe von Funktionen, die über das hinausgehen, was React allein bietet, und macht die Entwicklung moderner Webanwendungen einfacher, leistungsfähiger und skalierbarer.



Möglicherweise ist Ihnen aufgefallen, dass React nicht direkt Teil der Konzeptskizze ist. Das liegt daran, dass es nicht integraler Bestandteil des Setups selbst ist, sondern „nur“ eine Bibliothek zum Rendern von HTML.

Quelle: <https://nils-mehlhorn.de/posts/typescript-nodejs-react-ssr/>

Bereiten Sie eine kurze Präsentation vor, in der Sie das Technologiekonzept der App erläutern!

3 Praxis: Veröffentlichen der NFT-dApp

Bisher lief der Smart Contract auf der lokalen Blockchain „Hardhat“.

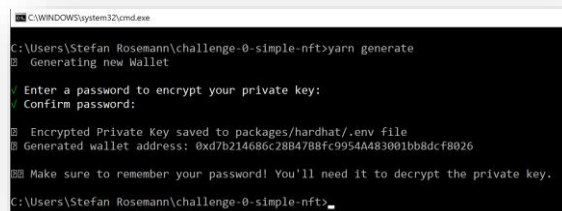
Im nächsten Schritt bringen Sie den SmartContract auf die Blockchain „Sepolia“ und die zugehörige App auf einen öffentlichen Webserver.

3.1 Smart Contract => Sepolia

(1) Eine Deployer-Adresse erzeugen:

Erzeugen Sie mit dem folgenden Befehl ein neues lokales Ethereum-Konto:

```
yarn generate
```



```
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>
```

Um einen Smart Contract zu veröffentlichen, muss die Transaktion mit einem privaten Schlüssel signiert werden. Es ist also ein Ethereum-Konto erforderlich. Während auf einer lokalen Blockchain ein vordefiniertes Standardkonto verwendet werden kann, das bei jeder lokalen Installation wiederverwendet werden kann, muss auf einer öffentlichen Blockchain (wie z.B. 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.

(2) Öffentliche Adresse und Kontostand einsehen:

Zeigen Sie Ihre Kontoinformationen an:

```
yarn account
```

Der Befehl zeigt die öffentliche Adresse und den Kontostand an.



```
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: 0xec774FbaA343f94f03e88651c11c0a019CF346A2
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
```

Wie viele Sepolia-Token hat der Deployvorgang verbraucht? _____

3.2 Frontend => Internet

In diesem Abschnitt verbinden Sie ihr Frontend mit der Sepolia-Blockchain und starten die Veröffentlichung auf der kostenfreien Hosting-Plattform „Vercel“.

(1) Frontend-Konfiguration anpassen:

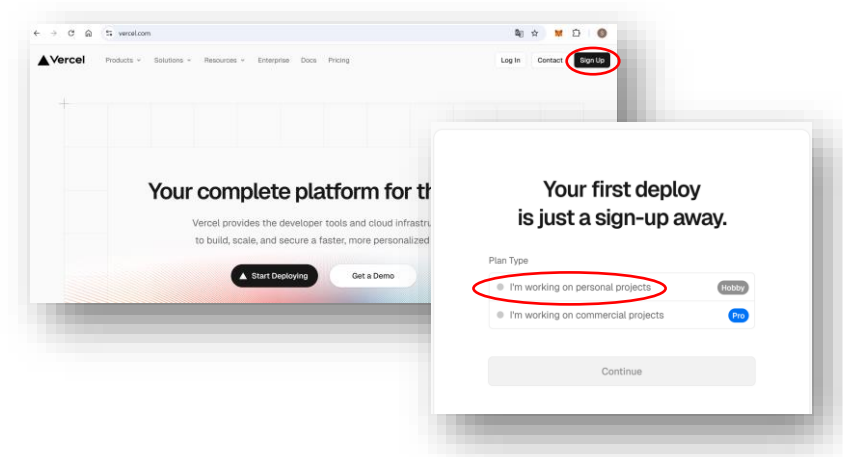
Verbinden Sie das Frontend mit dem Sepolia-Netzwerk, statt mit der lokal Hardhat-Chain. Ändern Sie in der Datei „**packages/nextjs/scaffold.config.ts**“ den Eintrag **targetNetworks** folgendermaßen:

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

(2) Vercel-Account erstellen

<https://vercel.com> 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.

Erstellen Sie zunächst einen (kostenfreien) Vercel-Account (z.B. über Ihre Email-Adresse) und melden Sie sich an.



(3) API-Token erstellen

Da die Veröffentlichung der Webseite scriptgesteuert abläuft, benötigen wir ein „API-Token“.

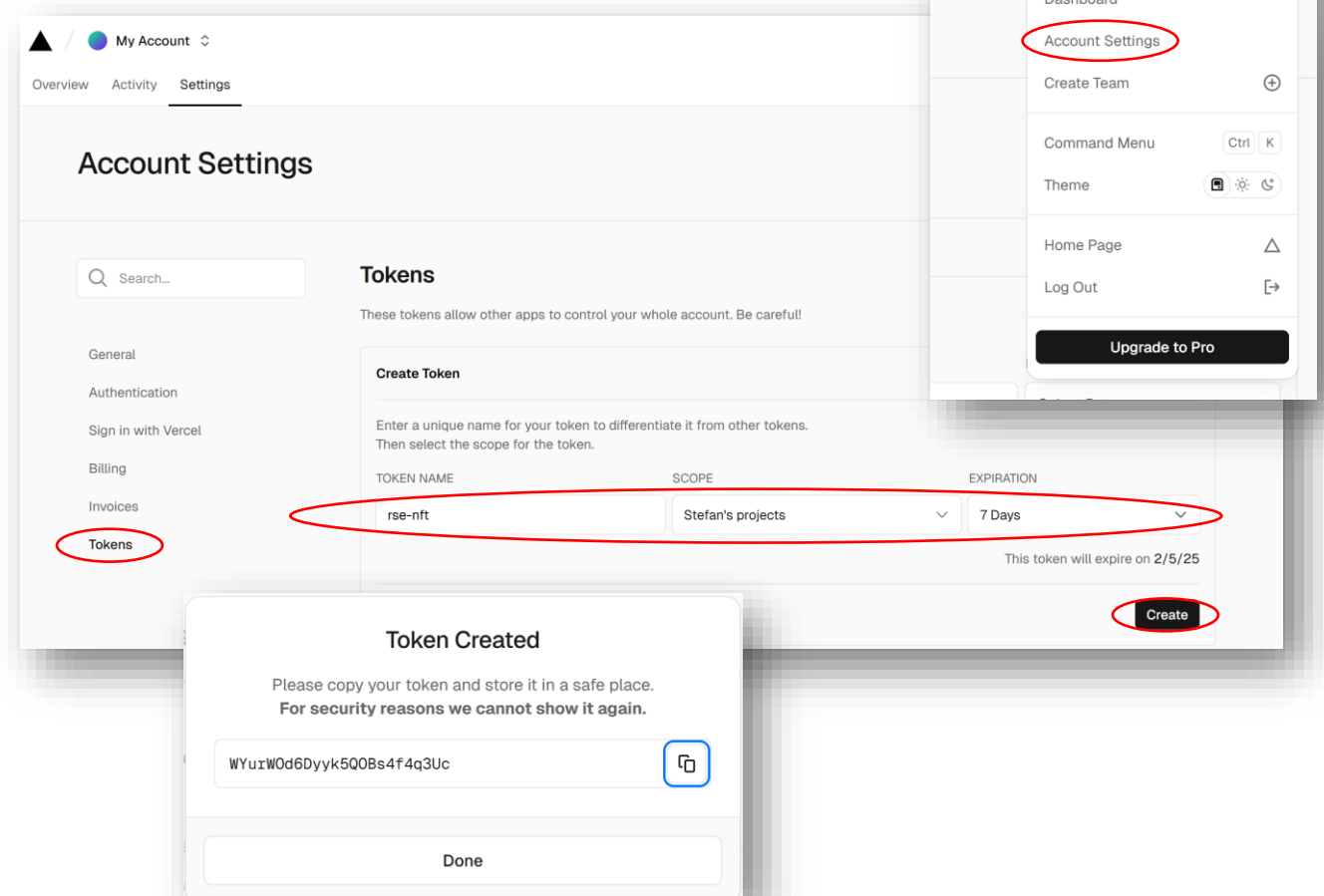
Ein API-Token ist ein spezieller Schlüssel, der es einer Anwendung ermöglicht, sicher mit einer Plattform wie Vercel zu kommunizieren. Er fungiert als eindeutiger Identifikator, der der Plattform mitteilt, dass die Anwendung berechtigt ist, auf bestimmte Dienste oder Daten zuzugreifen.

Stellen Sie sich das API-Token wie einen digitalen Ausweis vor. Wenn eine Anwendung Daten von Vercel anfordert oder Aktionen ausführen möchte, legt sie dieses Token vor, um ihre Identität und Berechtigung nachzuweisen. Ohne ein gültiges Token würde die Plattform die Anfrage ablehnen, ähnlich wie der Zutritt zu einem gesicherten Bereich ohne Ausweis verweigert wird.

Die Verwendung von API-Tokens stellt sicher, dass nur autorisierte Anwendungen auf die Plattform zugreifen können, was die Sicherheit und Integrität der Daten schützt. Entwickler können diese Tokens in ihren Anwendungen verwenden, um automatisierte Aufgaben durchzuführen oder Daten sicher auszutauschen.

Gehen Sie in die „Account-Settings“ (oben rechts) und dann zum Menüpunkt „Tokens“, um einen API-Token für Vercel zu erstellen.

Merken Sie sich den Token in einer Textdatei!



The screenshot displays the Vercel account settings page. On the right sidebar, the 'Account Settings' link is circled in red. The main content area shows the 'Tokens' section, which is also circled in red. The 'Create Token' form is visible, with the following details:

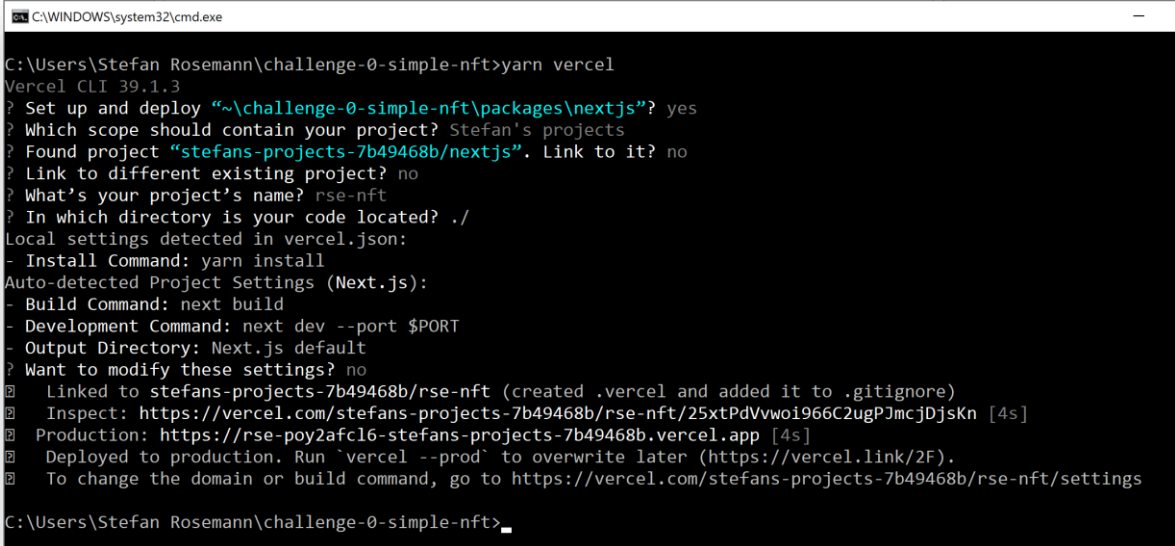
TOKEN NAME	SCOPE	EXPIRATION
rse-nft	Stefan's projects	7 Days

The 'Create' button at the bottom right of the form is circled in red. Below the form, a 'Token Created' modal is shown, indicating that the token has been successfully created and providing the token value: WYuiW0d6Dyyk5Q0Bs4f4q3Uc. A copy icon is present next to the token value.

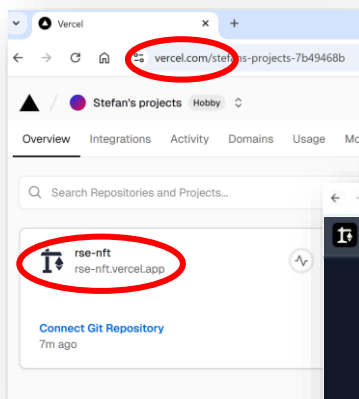
(4) Frontend veröffentlichen:

Veröffentlichen Sie Ihr Frontend auf der Plattform „Vercel“.
Tragen Sie Ihren persönlichen Vercel-API-Token ein.

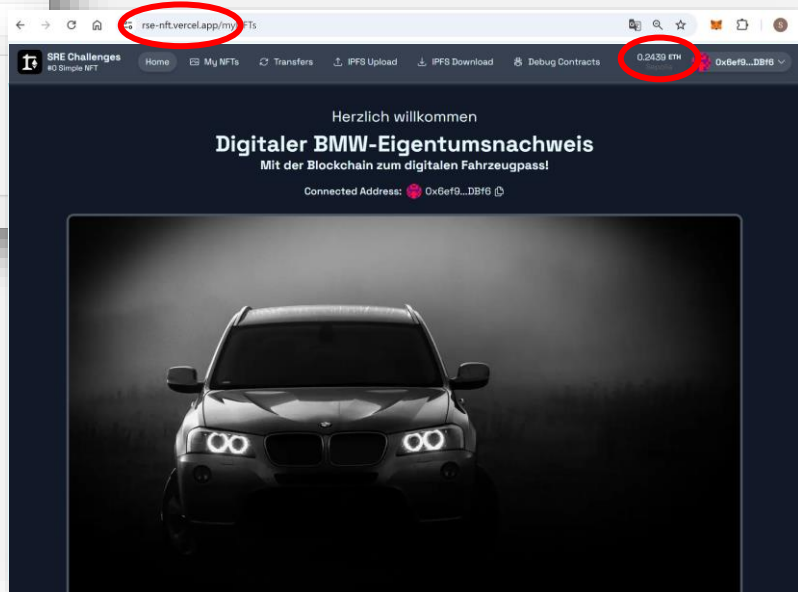
```
yarn vercel -t <API-Token>
```



```
C:\WINDOWS\system32\cmd.exe
C:\Users\Stefan Rosemann\challenge-0-simple-nft>yarn vercel
Vercel CLI 39.1.3
? Set up and deploy "~\challenge-0-simple-nft\packages\nextjs"? yes
? Which scope should contain your project? Stefan's projects
? Found project "stefans-projects-7b49468b/nextjs". Link to it? no
? Link to different existing project? no
? What's your project's name? rse-nft
? In which directory is your code located? ./
Local settings detected in vercel.json:
- Install Command: yarn install
Auto-detected Project Settings (Next.js):
- Build Command: next build
- Development Command: next dev --port $PORT
- Output Directory: Next.js default
? Want to modify these settings? no
[2] Linked to stefans-projects-7b49468b/rse-nft (created .vercel and added it to .gitignore)
[2] Inspect: https://vercel.com/stefans-projects-7b49468b/rse-nft/25xtPdVvwoi966C2ugPJmcjDjsKn [4s]
[2] Production: https://rse-poy2afcl6-stefans-projects-7b49468b.vercel.app [4s]
[2] Deployed to production. Run `vercel --prod` to overwrite later (https://vercel.link/2F).
[2] To change the domain or build command, go to https://vercel.com/stefans-projects-7b49468b/rse-nft/settings
C:\Users\Stefan Rosemann\challenge-0-simple-nft>
```



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

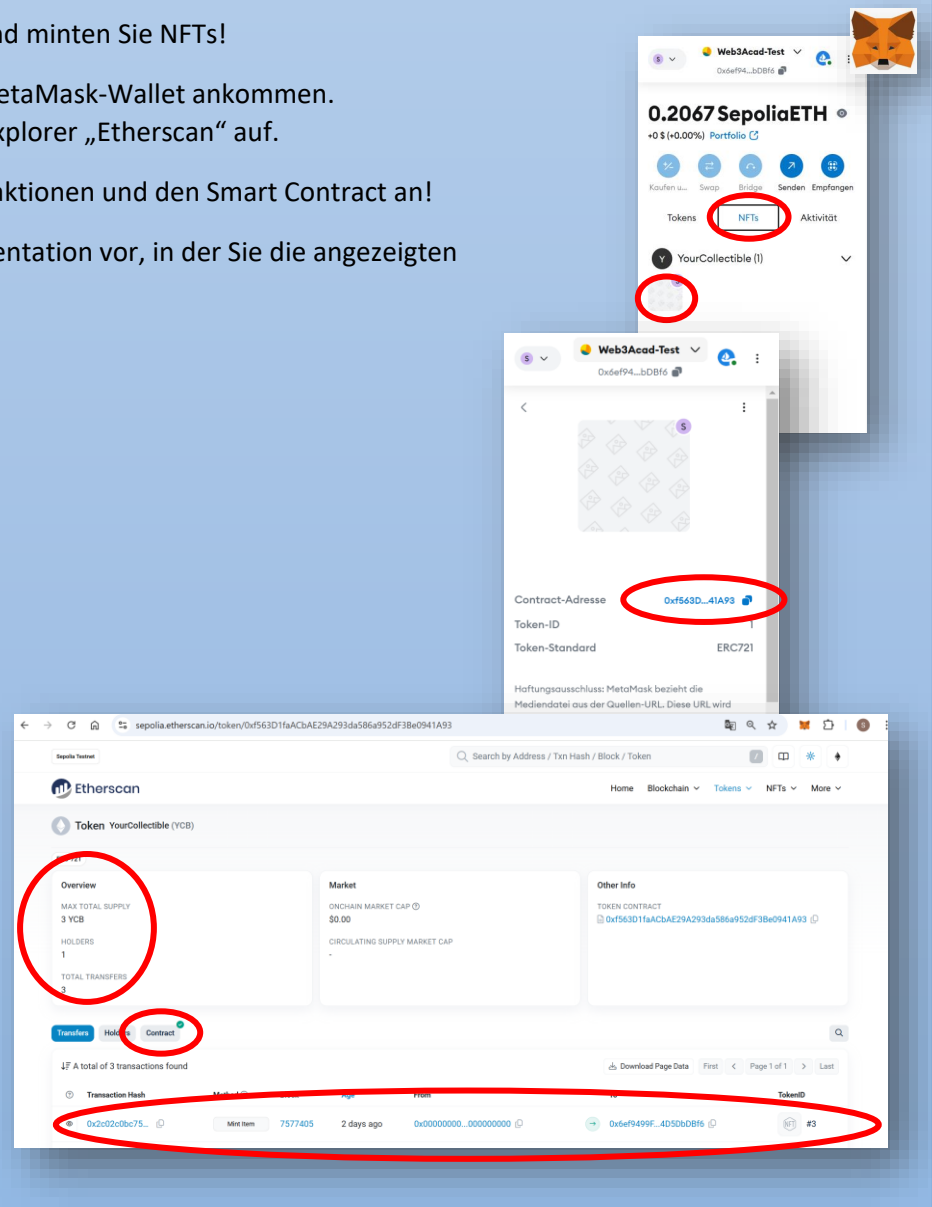


Testen Sie Ihre WebApp und minten Sie NFTs!

Die NFTs sollten in Ihrer MetaMask-Wallet ankommen.
Rufen Sie den BlockchainExplorer „Etherscan“ auf.

Schauen Sie sich die Transaktionen und den Smart Contract an!

Bereiten Sie eine Kurzpräsentation vor, in der Sie die angezeigten
Daten erläutern!



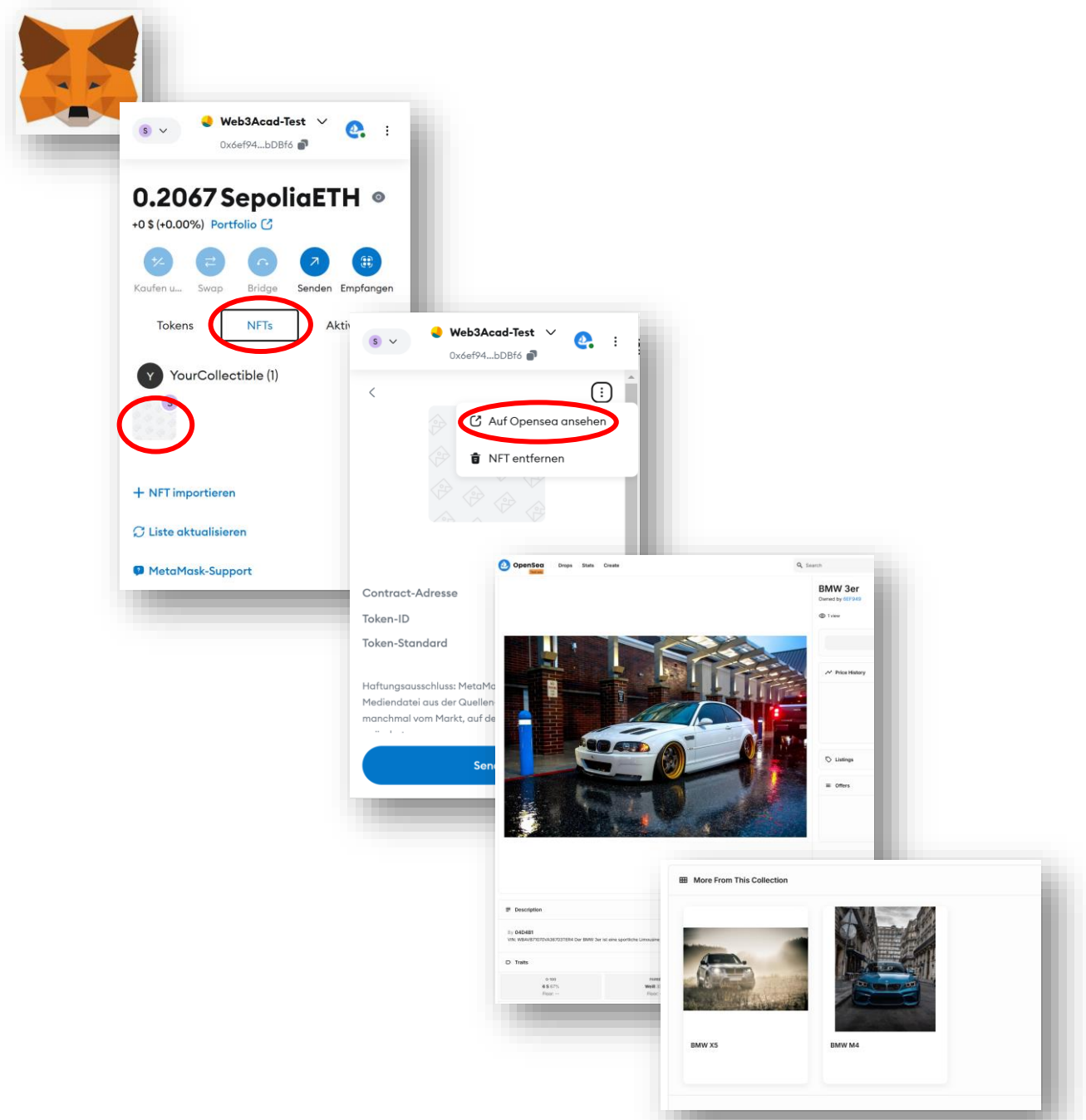
4 NFT auf OpenSea anzeigen

OpenSea ist ein großer Online-Marktplatz für NFTs, auf dem man digitale Dinge als NFTs kaufen, verkaufen oder tauschen kann. NFTs können alles Mögliche sein, z. B. Kunstwerke, Musik, Videos, Sammelkarten, Tickets oder sogar Gegenstände in Videospielen. Es funktioniert ähnlich wie eBay und macht es super einfach, NFTs zu entdecken, zu kaufen oder selbst zu verkaufen.

OpenSea erfährt automatisch von neu geminteten NFT, indem Sie alle gängigen Ethereum-Blockchain mit sogenannten „OpenSea-Indexer-Bots“ überwachen. Diese Bots scannen alle Blöcke und suchen nach Transaktionen, die einen ERC-721-kompatiblen Smart Contract ansprechen.

Finden Sie eine öffentliche Mint-Transaktion ruft OpenSea die URI (IPFS CID) ab und erstellt mit den Metadaten und der Bild-URL einen neuen OpenSea-Eintrag.

Dies ist auch für unsere NFTs geschehen. Entdecken Sie Ihr NFT auf OpenSea!



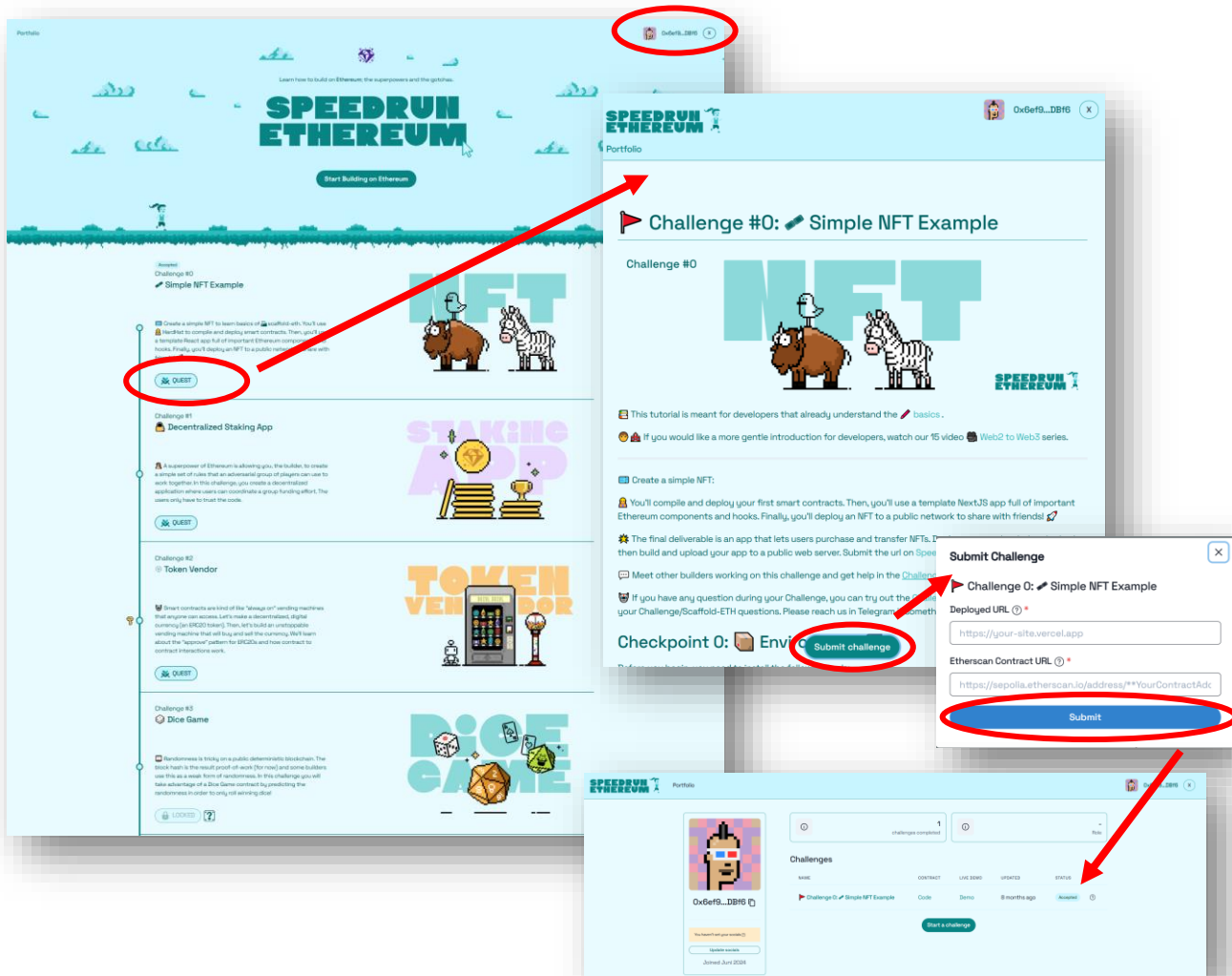
5 Speed Run Ethereum

Die von Ihnen entwickelte BMW-NFT-App basiert auf einer Übung aus „Speed Run Ethereum“.

"Speed Run Ethereum" ist ein interaktiver Kurs, der darauf abzielt, Entwicklern die Erstellung von Anwendungen auf der Ethereum-Blockchain beizubringen. Durch eine Reihe von praktischen Herausforderungen lernen Teilnehmer, sogenannte "Smart Contracts" zu schreiben und dezentrale Anwendungen (dApps) zu entwickeln. Diese Herausforderungen sind so gestaltet, dass sie Anfänger Schritt für Schritt durch den Entwicklungsprozess führen und ihnen helfen, ein tiefes Verständnis für die Ethereum-Technologie zu erlangen. (speedrunethereum.com)

Die offizielle Ethereum-Website empfiehlt "Speed Run Ethereum" als Lernressource für Entwickler, die ihre Fähigkeiten im Umgang mit Solidity und der Entwicklung von dApps verbessern möchten. (ethereum.org)

Ein besonderes Feature von "Speed Run Ethereum" ist, dass Teilnehmer ihre **Lösungen einreichen** und automatisch **überprüfen** lassen können. Dafür müssen sie die Adresse ihres **Smart Contracts** und die URL ihrer **WebApp** angeben. Sobald die Lösung korrekt ist, wird die Challenge als bestanden gewertet und neue Herausforderungen freigeschaltet!



The image shows a collage of screenshots from the "Speed Run Ethereum" website. Red circles and arrows highlight the steps to submit a challenge:

- Challenge List:** A red circle highlights the "QUEST" button next to "Challenge #0: Simple NFT Example".
- Challenge Details:** A red circle highlights the "Submit challenge" button in the "Checkpoint 0" section.
- Submission Modal:** A red circle highlights the "Submit" button in the "Submit Challenge" modal.
- Challenge Details (Again):** A red circle highlights the "Submit challenge" button in the "Checkpoint 0" section.
- Submission Modal (Again):** A red circle highlights the "Submit" button in the "Submit Challenge" modal.

Lassen Sie Ihre „Challenge #0“ überprüfen!