

# Anleitung zur Nutzung des Prototyps

Folgend die Anleitung zur Nutzung des Prototyps, der im Zuge der Masterthesis von Timur Burkholz entwickelt wurde. Das folgende Dokument ist wie folgt gegliedert.

## Inhaltsverzeichnis

<b>1) Aufbau Projekt .....</b>	<b>3</b>
<b>2) Nutzung über Browser (Einrichtung) .....</b>	<b>5</b>
2.1) <i>Installation MetaMask</i> .....	5
2.2) <i>Nutzung Kovan Testnet</i> .....	8
2.2.1) Erlauben von Verbindung mit Testnetzwerken .....	8
2.2.2) Verbinden mit Kovan Netzwerk .....	9
2.2.3) Prototyp aufrufen .....	10
2.2.4) Test-Ether für das Kovan-Netzwerk erhalten .....	11
<b>3) Installation lokal mit Entwicklungsumgebung .....</b>	<b>14</b>
3.1) <i>Installation von Ganache</i> .....	14
3.2) <i>Installation von Node</i> .....	15
3.3) <i>Installation des Prototyps</i> .....	15
3.4) <i>Verbinden lokaler Blockchain mit MetaMask</i> .....	16
<b>4) Nutzungsanleitung für Funktionalitäten des Prototyps .....</b>	<b>24</b>
4.1) <i>Thesis Token kaufen</i> .....	24
4.2) <i>Thesis Token verkaufen</i> .....	25
4.3) <i>Ether staken</i> .....	27
4.4) <i>Ether unstaken</i> .....	29
4.5) <i>Tokenauszahlung starten</i> .....	31

Der Prototyp kann durch zwei Varianten genutzt werden:

- 1. Nutzung der öffentlichen Webanwendung:** Das Frontend der dezentralen Applikation wurde auf einer öffentlichen Seite veröffentlicht und die Smart Contracts auf dem Kovan Test-Netzwerk veröffentlicht. Hierbei wird keine lokale Entwicklungsumgebung benötigt. Die Anleitung der Nutzung dieser Variante findet sich in Kapitel 2.
- 2. Lokale Nutzung mit Entwicklungsumgebung:** Der Prototyp wurde lokal implementiert und durch eine lokale Entwicklungsumgebung genutzt und getestet. Die Installations- und Nutzungsanleitung finden sich in Kapitel 3.

Folgende Technologien werden benötigt:

**Variante 1** (Nutzung öffentliche Webanwendung und Kovan Test-Netzwerk):

- Browser (Firefox oder Chrome); Firefox ab Version 90.0, Chrome ab Version 80.0
- MetaMask-Browsererweiterung (für Firefox oder Chrome); ab Version 10

**Variante 2** (Lokale Nutzung mit Entwicklungsumgebung):

- Ganache; Version v7.0.3
- Node; Version 16.14.2
- Truffle; Version v5.5.6
- Browser (Firefox oder Chrome); Firefox ab Version 90.0, Chrome ab Version 80.0
- MetaMask-Browsererweiterung (für Firefox oder Chrome); ab Version 10

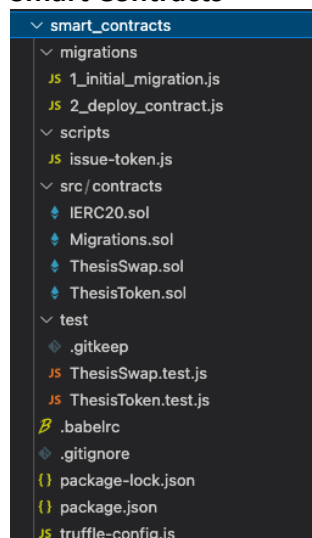
Die Installation der Technologien wird in Kapitel 2 (für Variante 1) bzw. Kapitel 3 (für Variante 2) vorgestellt. Eine Nutzungsanleitung der Hauptfunktionalitäten des Prototyps findet sich in Kapitel 4. In Kapitel 1 findet sich eine Beschreibung zum Aufbau des Projektes.

## 1) Aufbau Projekt

Folgend eine kurze Erläuterung zum Aufbau des Projekts. Das Projekt ist durch zwei Ordner unterteilt:

- **Smart\_contracts:** Hier finden sich die Smart Contracts der dezentralen Applikation. Diese wurden mit Solidity umgesetzt.
- **Defi\_frontend:** Hier findet sich die Frontend Applikation. Diese wurde mit React umgesetzt.

### Smart Contracts

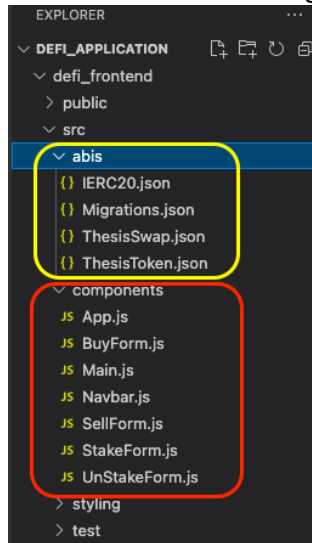


Der Smart\_Contracts-Ordner ist wie folgt aufgebaut:

- **Migrations:** Im Migrations-Unterordner finden sich zwei Dateien, die zur Migration der Smart Contracts auf eine lokale oder remote Blockchain benötigt werden. Dies ermöglicht die Migration mit Truffle.
- **Src/Contracts:** Im Contracts-Unterordner finden sich vier Smart Contracts, die in Solidity implementiert wurden.
  - **IERC20:** Hierdurch wird das ERC-20-Interface definiert, dieses wird durch den Thesis Token implementiert.
  - **Migrations:** Wird verwendet zur Migration des ThesisSwap und ThesisToken Smart Contract.
  - **ThesisSwap:** Implementiert den Smart Contract der Tauschbörse und ermöglicht so den Kauf / Verkauf von Thesis Token und das Staking / Unstaking von Ether.
  - **ThesisToken:** Implementiert das IERC-20 Interface und somit einen eigenen Token.
- **Test:** Hier finden sich die Unit-Tests der Smart Contracts, unterteilt in Tests für den ThesisSwap Smart Contract und dem Thesis Token Smart Contract.

## Defi\_Frontend

Das Frontend wurde mit React umgesetzt. Im *Src*-Order finden sich der implementierte Code. Dieser ist wie folgt gegliedert:



Im *Abis*-Unterordner finden sich die Smart Contracts, die im ABIS-Format bereitgestellt werden. Über dieses Format können die Funktionalitäten der Smart Contracts verwendet werden. Beim Migrieren der Smart Contracts auf die Blockchain, werden die Dateien automatisch aktualisiert, wodurch das Frontend die neuste Version der Contracts verwendet.

Im *Components*-Unterordner findet sich die Komponenten der React-Anwendung. Die App-Klasse definiert dabei die Hauptklasse der Anwendungen, hier werden die Funktionalitäten des Frontend bereitgestellt.

## 2) Nutzung über Browser (Einrichtung)

Diese Kapitel beschreibt die Nutzung des Prototyps über den Browser, ohne eine lokale Entwicklungsumgebung zu benötigen. Das Frontend der dezentralen Applikation wurde auf [https://timucini.github.io/DeFi\\_Application/](https://timucini.github.io/DeFi_Application/) veröffentlicht.

Zur Nutzung im Browser wird die MetaMask-Browsererweiterung benötigt, siehe hierfür Kapitel 2.1)

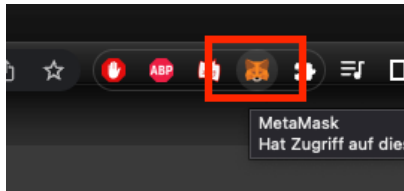
### 2.1) Installation MetaMask

Die MetaMask-Erweiterung kann auf gängigen Browsern installiert werden.

1. Installieren der MetaMask-Browsererweiterung für den genutzten Browser:
  - a. Für Chrome:  
<https://chrome.google.com/webstore/detail/metamask/nkbihfbeogaeaoehlefnkodbefgpgknn?hl=de>
  - b. Für Firefox:  
<https://addons.mozilla.org/de/firefox/addon/ether-metamask/>

2. Wenn installiert, MetaMask einrichten:

- a. Im Browser die MetaMask-Erweiterung öffnen, z.B. durch Klick auf das Icon oben rechts im Browser.



- b. Daraufhin öffnet sich ein MetaMask-Fenster im Browser. Dann mit der Einrichtung von MetaMask beginnen (Erste Schritte).



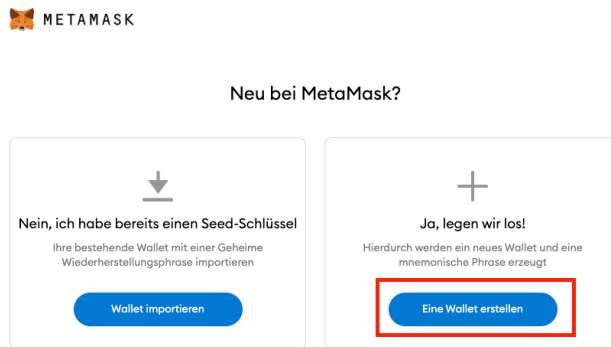
Willkommen zu MetaMask

MetaMask ist ein sicherer Identitätssafe für Ethereum.

Wir freuen uns, Sie zu sehen.

Erste Schritte

- c. In der geöffneten Seite dann eine Wallet erstellen.



- d. Daraufhin muss ein frei wählbares Passwort für die Wallet erstellt werden (Passwort wird für die Verwendung benötigt, daher bitte merken).

METAMASK  
< Zurück

## Passwort erstellen

Neues Passwort (min. 8 Zeichen)

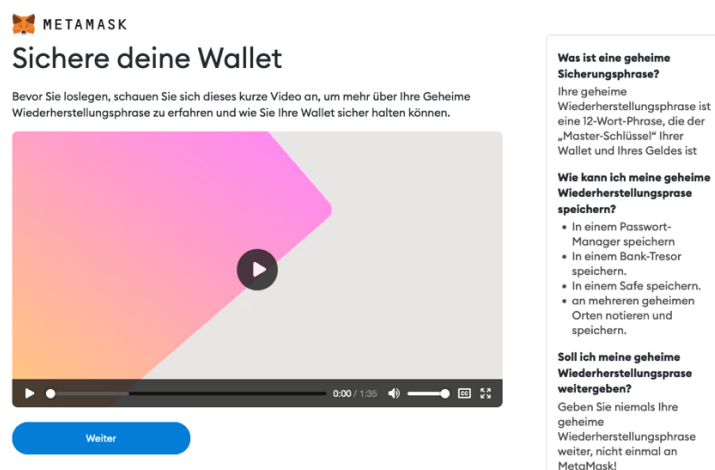
Passwort bestätigen

☐ Ich habe die [Nutzungsbedingungen](#) gelesen und stimme ihnen zu

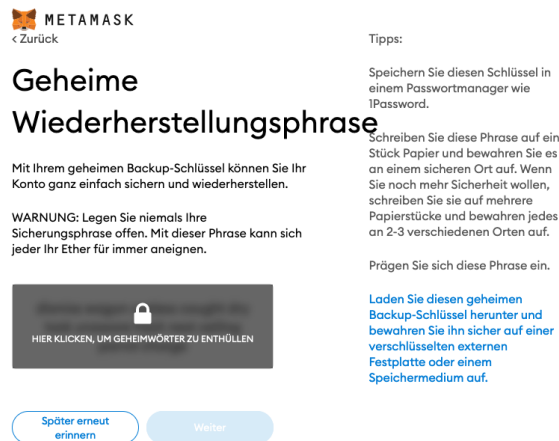
Erstellen

Passwort wählen und Account erstellen.

- e. Dann öffnet sich das folgende Fenster. Klicken auf „Weiter“.



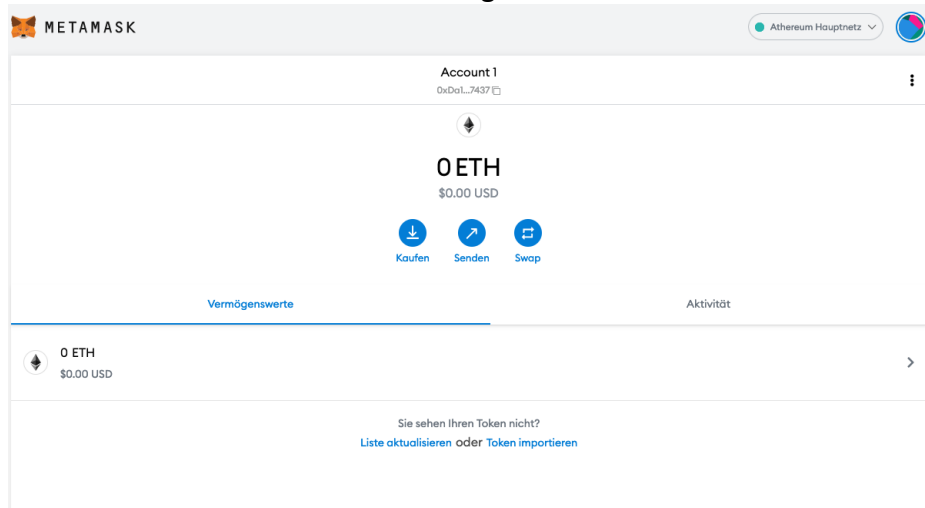
f. Daraufhin öffnet sich das folgende Fenster.



Hier können die Geheimwörter enthüllt werden, mit denen man die Wallet bei Passwortverlust freischalten kann.

Die Geheimwörter können gespeichert werden oder später erneut abgerufen werden. Die Geheimwörter werden normalerweise nicht benötigt.

g. Nach der Auswahl öffnet sich das folgende Fenster:



Die Einrichtung von MetaMask ist damit abgeschlossen.

Wird der Prototyp nur im Browser verwendet, muss als nächster Schritt die Verbindung mit dem Kovan-Testnetzwerk eingerichtet werden, hierfür siehe Kapitel 2.2.

Wird der Prototyp lokal installiert, muss sich mit der lokalen Ganache Blockchain verbunden werden, siehe hierfür Kapitel 3.1.1.

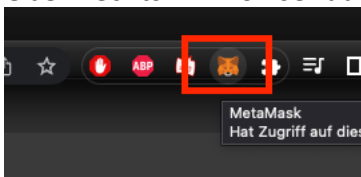
## 2.2) Nutzung Kovan Testnet

Die Webanwendung unter [https://timucini.github.io/DeFi\\_Application/](https://timucini.github.io/DeFi_Application/) nutzt die Smart Contracts, die auf dem Kovan-Testnetzwerk veröffentlicht wurden. In diesem Kapitel wird beschrieben, wie sich mit diesem Netzwerk verbunden werden kann. Dafür mit Kapitel 2.2.1 beginnen.

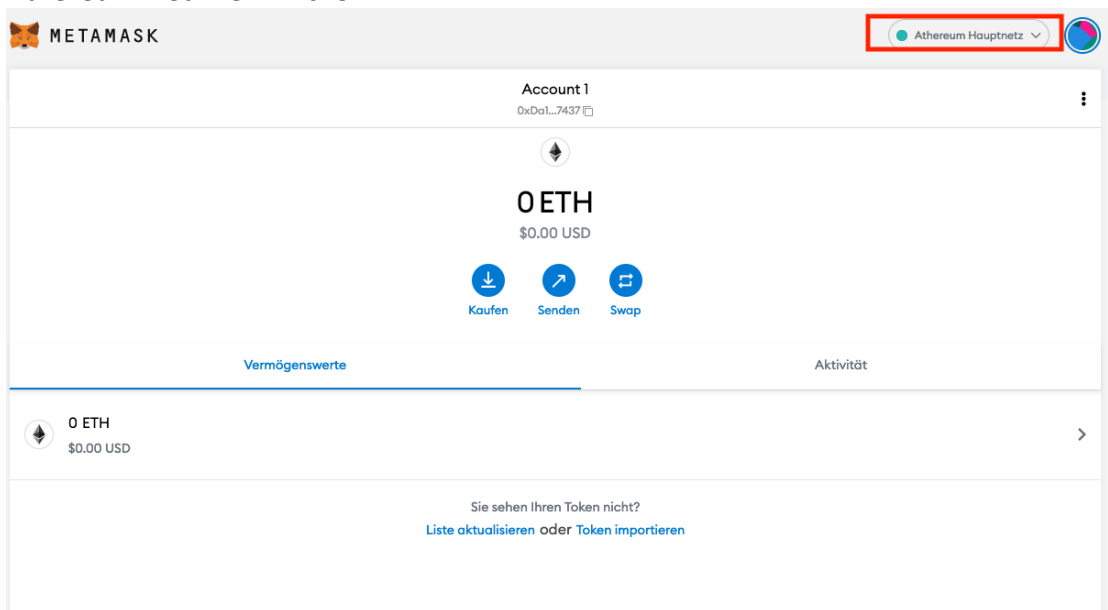
### 2.2.1) Erlauben von Verbindung mit Testnetzwerken

Zuerst muss die Option zur Verbindung mit Testnetzwerken in MetaMask aktiviert werden. Dafür sind folgende Schritte notwendig:

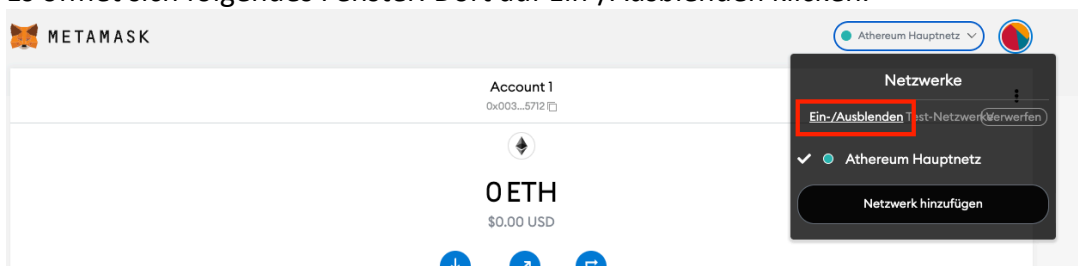
1. Oben rechts im Browser auf das MetaMask-Icon klicken.



2. Daraufhin öffnet sich folgendes Fenster. Dort im oberen rechten Teil des Fensters auf Athereum Netzwerk klicken.



3. Es öffnet sich folgendes Fenster. Dort auf Ein-/Ausblenden klicken.





4. Folgendes Fenster öffnet sich, dort Test-Netzwerk anzeigen aktivieren.

Test-Netzwerke anzeigen

Wählen Sie dies, um Testnetzwerke in der Netzwerkliste anzuzeigen

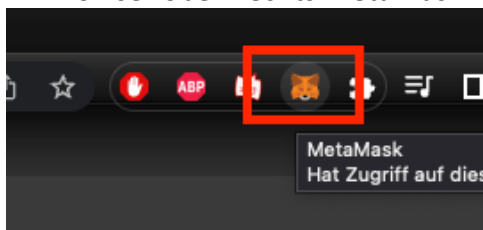


Damit kann sich MetaMask mit Testnetzwerken verbinden. Als nächstes muss sich mit dem Kovan-Netzwerk verbunden werden (Kapitel 2.2.2).

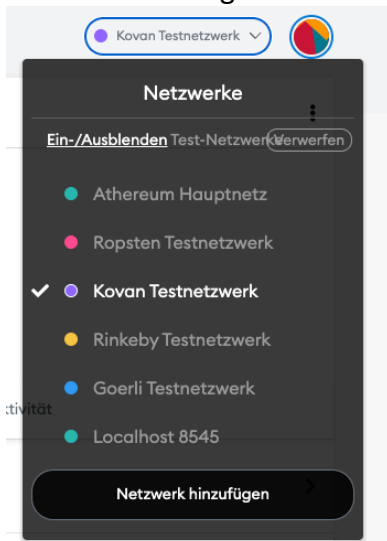
## 2.2.2) Verbinden mit Kovan Netzwerk

Um sich mit dem Kovan Netzwerk zu verbinden:

1. Im Browser oben rechts MetaMask-Icon anklicken.



2. Es öffnet sich folgendes Fenster, dort Kovan Testnetzwerk auswählen.



MetaMask ist jetzt mit dem Kovan Testnetzwerk verbunden. Als nächstes wird der Prototyp aufgerufen (Kapitel 2.2.3).

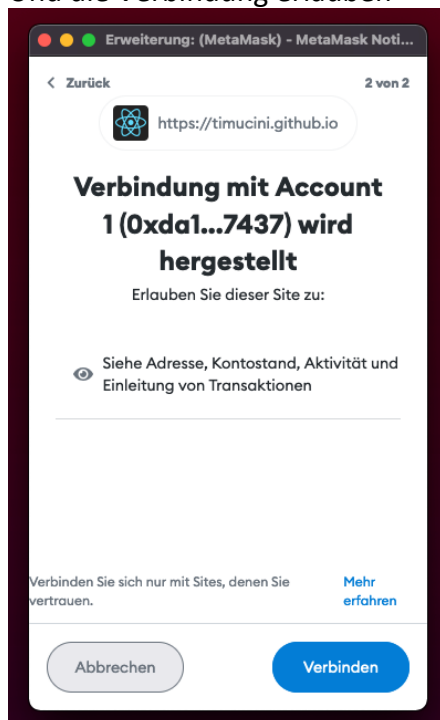
### 2.2.3) Prototyp aufrufen

Der Prototyp kann jetzt im Browser verwendet werden. Hierfür:

1. [https://timucini.github.io/DeFi\\_Application/](https://timucini.github.io/DeFi_Application/) aufrufen
2. Es öffnet sich folgendes Fenster in MetaMask, hier auf Weiter klicken:



3. Und die Verbindung erlauben



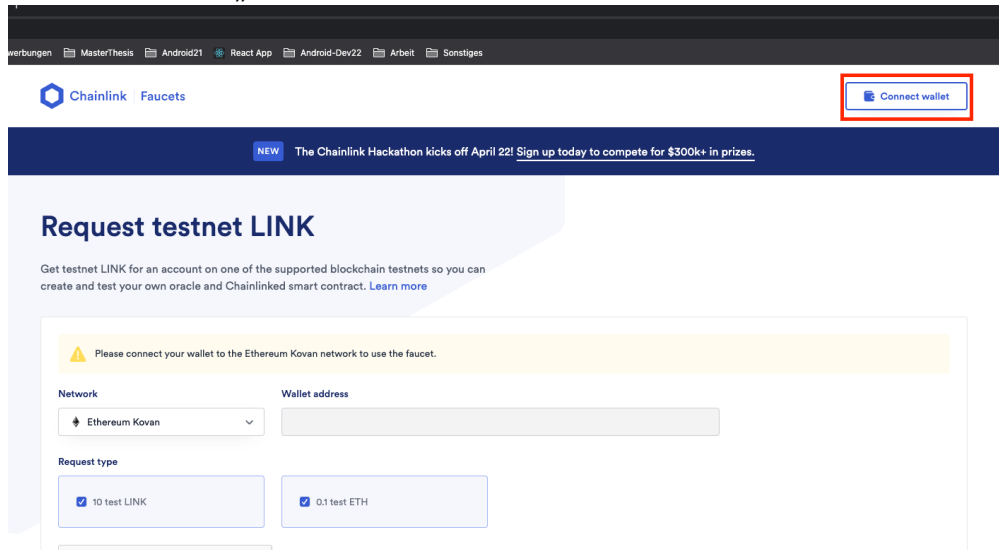
MetaMask ist jetzt mit dem Kovan-Netzwerk und der dezentralen Applikation verbunden und kann genutzt werden. Um Thesis Token zu kaufen, benötigt man Test-Ether. Wie man diese erhält, wird in Kapitel 2.2.4 erläutert.

## 2.2.4) Test-Ether für das Kovan-Netzwerk erhalten

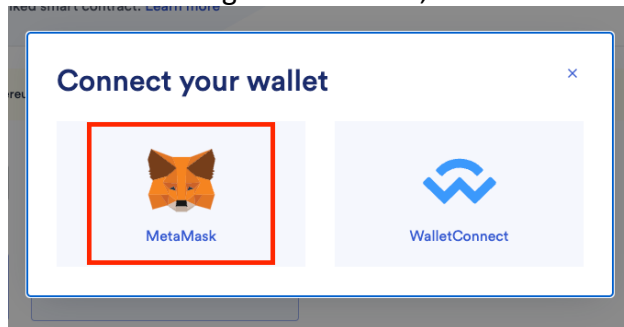
Um Thesis Token kaufen zu können oder Ether zum Staking einzahlen zu können, wird Test-Ether genutzt. Ether müsste hierfür normalerweise käuflich erworben werden. Im Kovan-Testnetzwerk ist es möglich Test-Ether zu erhalten, ohne diese bezahlen zu müssen.

Dafür folgende Schritte tätigen:

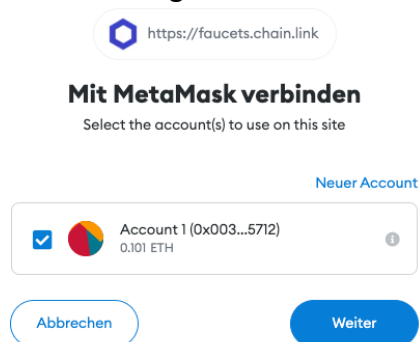
1. Aufrufen der Seite <https://faucets.chain.link/>
2. Dort oben rechts „Connect Wallet“ anklicken.



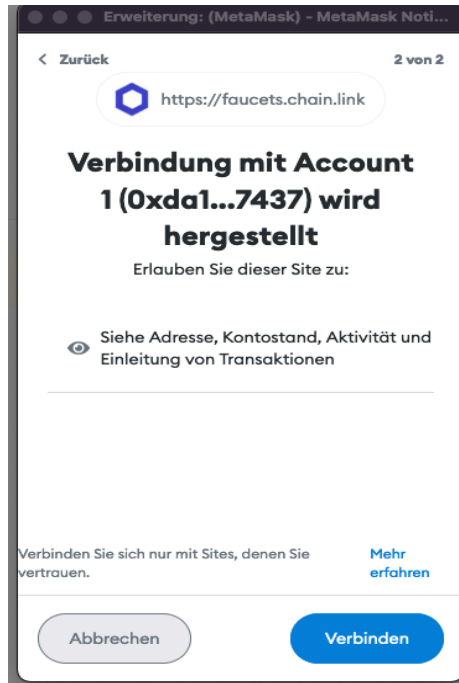
3. Es öffnet sich folgendes Fenster, dort MetaMask auswählen



4. Es öffnet sich folgendes Fenster in MetaMask, dort weiter klicken.



5. Im nächsten Fenster in MetaMask die Verbindung erlauben.



6. Die Wallet ist dann mit der Anwendung verbunden, dort folgende Optionen auswählen und „Send Request“ klicken.

## Request testnet LINK

Get testnet LINK for an account on one of the supported blockchain testnets so you can create and test your own oracle and Chainlinked smart contract. [Learn more](#)

Network

Ethereum Kovan

Wallet address

0x0039391e964ed5d07cff3b121a397c3184b25712

Request type

☐ 10 test LINK

☒ 0.1 test ETH

✓ Ich bin kein Roboter.

reCAPTCHA

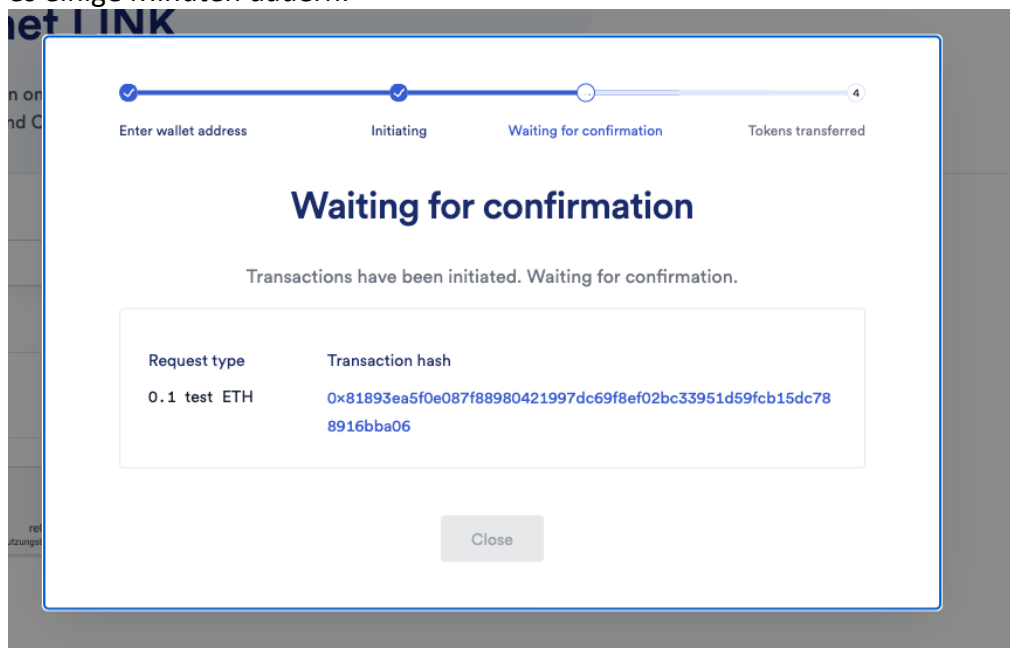
Datenschutzerklärung · Nutzungsbedingungen

Send request

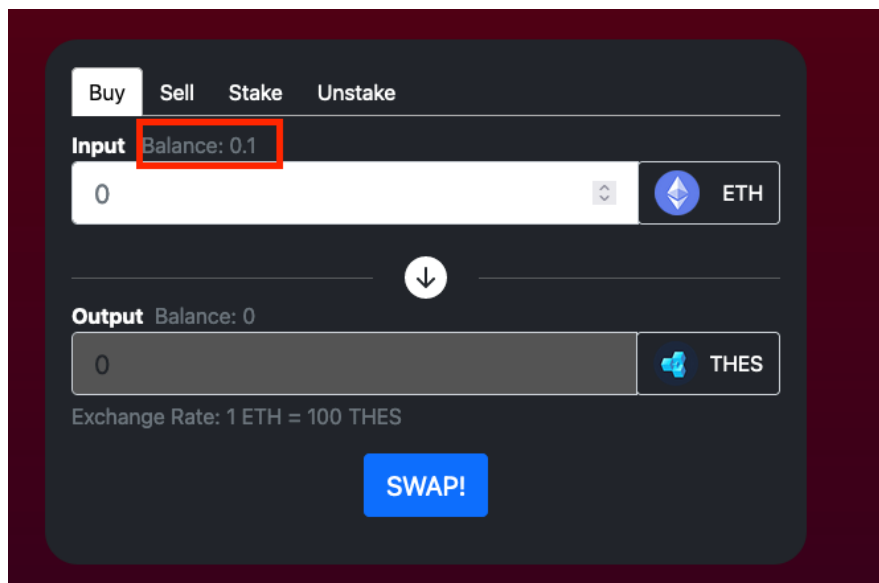
?

Email us at [faucets@chain.link](mailto:faucets@chain.link) if there is an issue here, thanks!

7. Es öffnet sich folgendes Fenster und Test-Ether wird übertragen. Bis zum Erhalt kann es einige Minuten dauern.



Nach Beenden der Transaktion erhält das verbundene Konto Test-Ether, mit denen die Thesis Tokens im Prototyp gekauft werden können.



In der Anwendung sieht man dann die Balance über dem Input-Feld. Der Prototyp kann jetzt vollständig über den Browser und dem Kovan Testnet verwendet werden.

### 3) Installation lokal mit Entwicklungsumgebung

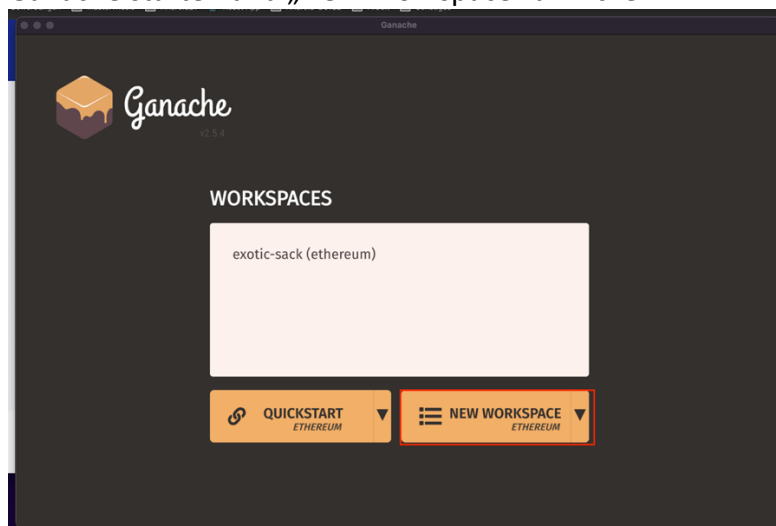
Um den Prototypen vollständig auf der lokalen Maschine laufen zu lassen, müssen folgende Schritte getätigt werden:

#### 3.1) Installation von Ganache

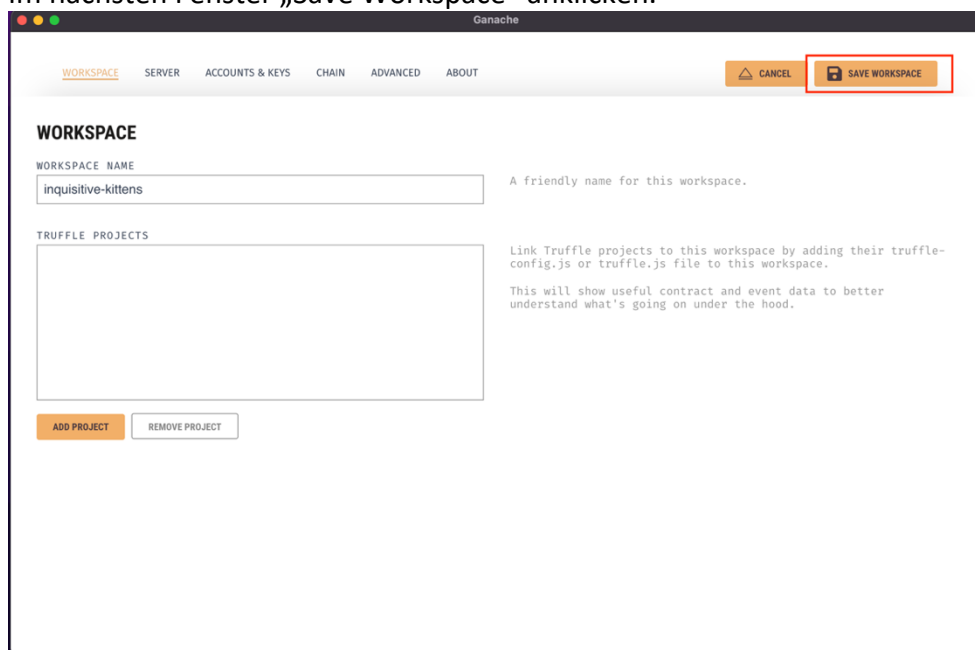
Zur lokalen Nutzung des Prototyps wird eine lokale Blockchain benötigt, die über Ganache realisiert wird. Hierzu muss unter <https://trufflesuite.com/ganache/index.html> die entsprechende Software gedownloadet und installiert werden.

Nach erfolgreicher Installation wird die lokale Blockchain wie folgt erstellt:

1. Ganache starten und „New Workspace“ anklicken.



2. Im nächsten Fenster „Save Workspace“ anklicken.



Damit ist die lokale Blockchain mit Ganache eingerichtet. Weiter mit Kapitel 3.2.

### 3.2) Installation von Node

Zur Installation der Packages im Projekt wird Node verwendet. Hierfür die passende Node-Version unter <https://nodejs.org/en/download/> downloaden und installieren.

Zur Überprüfung der Installation kann im Terminal über die Eingabe „node -v“ die Node-Version überprüft werden. Node muss im Terminal verfügbar sein. Danach weiter mit Kapitel 3.3.

### 3.3) Installation des Prototyps

Um den Prototyp lokal zu installieren folgende Anweisungen folgen:

#### Installation der Smart Contracts

1. Herunterladen des Source-Codes.  
Dieser kann unter [https://github.com/timucini/DeFi\\_Application](https://github.com/timucini/DeFi_Application) abrufen werden oder wurde über die HTW-Cloud bereitgestellt.
2. Im Terminal die Projekt-Directory öffnen.
3. Den Smart Contract-Folder im Terminal öffnen durch „*cd smart\_contracts/*“
4. Installation der Node-Module für die Smart Contracts. Dafür folgenden Befehl im Terminal aufrufen:  
„*npm install*“
5. Danach folgenden Befehl im Terminal ausführen:  
„*npm install -g truffle*“  
  
Dadurch wird Truffle installiert, was für das Deployment der Smart Contracts benötigt wird.
6. Nachdem Truffle installiert wurde folgenden Befehl im Terminal ausführen (Hinweis hierfür muss Ganache laufen, siehe Kapitel 2.1.1):  
„*truffle migrate --reset*“  
  
Dadurch werden die Smart Contracts auf die lokale Blockchain in Ganache migriert.
7. Im nächsten Schritt folgenden Befehl im Terminal ausführen:  
„*truffle exec scripts/issue-token.js*“  
  
Wird benötigt zur richtigen Zuweisung der lokalen Accounts.
8. Um die Einrichtung zu Überprüfen folgenden Befehl zum Start der Unit Tests starten:  
„*truffle test*“

Die Unit Tests müssten alle erfolgreich durchlaufen.

Die Smart Contracts wurden jetzt erfolgreich auf der lokalen Blockchain mit Ganache migriert. Im nächsten Schritt muss das Frontend vorbereitet werden.

## Installation Frontend

Folgende Anweisungen zur Installation des Frontends ausführen:

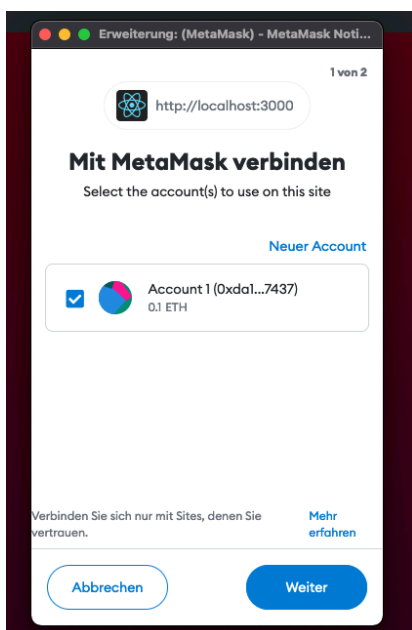
1. Im Terminal die Projekt-Directory öffnen
2. Frontend im Terminal öffnen durch:  
„cd defi\_frontend/“
3. Node Module installieren durch Befehl im Terminal:  
„npm install“
4. Nachdem Node Module installiert wurden im Terminal:  
„npm start“

Dadurch wird das Frontend gestartet. Die Anwendung sollte sich im Browser unter [http://localhost:3000/DeFi\\_Application](http://localhost:3000/DeFi_Application) öffnen.

Im Browser öffnet sich zudem ein MetaMask-Fenster (falls MetaMask nicht installiert und eingerichtet bitte Schritte aus 2.1 folgen). Hierbei muss die Verbindung mit der lokalen Blockchain ermöglicht werden. Dies wird im folgenden Kapitel erläutert.

### 3.4) Verbinden lokaler Blockchain mit MetaMask

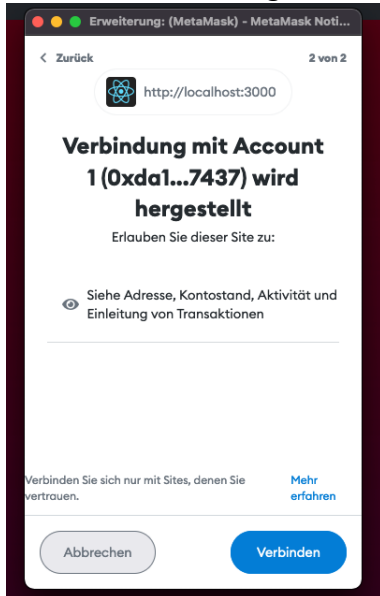
Nach Starten der Frontend Anwendung öffnet sich folgendes MetaMask-Fenster im Browser.



Nach Klicken auf „Weiter“ öffnet sich ein weiteres Fenster.

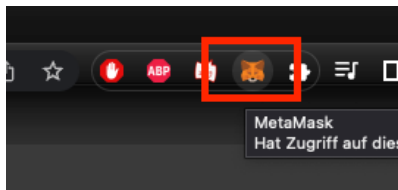


Dann die Verbindung mit der Anwendung erlauben.

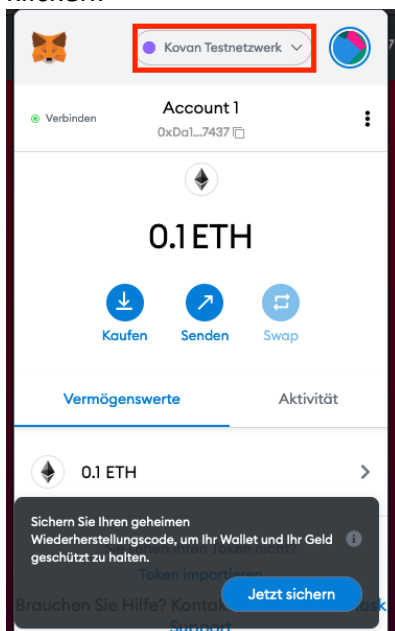


Nun muss die lokale Ganache Blockchain in MetaMask hinzugefügt und verbunden werden. Dafür folgende Anweisungen tätigen:

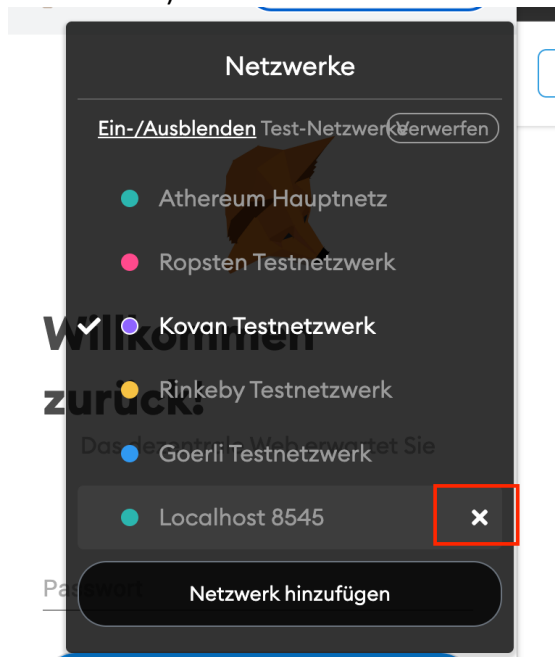
1. MetaMask im Browser öffnen durch Klicken des MetaMask-Icons oben rechts:



2. Im geöffneten MetaMask-Fenster die Schaltfläche zum Auswählen des Netzwerkes klicken:

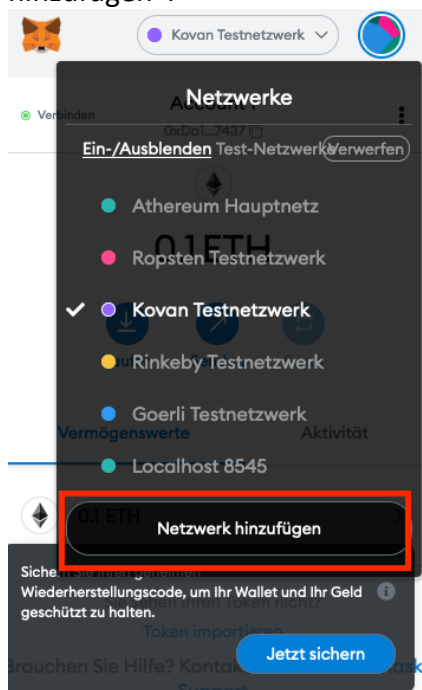


3. Dann falls Localhost 8545 als Netzwerk vorhanden ist, dieses Netzwerk löschen, da dieser Port verwendet werden soll. (Falls Localhost 8545 nicht vorhanden, mit Schritt 4 fortfahren):



Durch Klick auf das „X“ und bestätigen wird das Netzwerk gelöscht.

4. Dann das lokale Ganache Netzwerk hinzufügen, dafür im gleichen Fenster „Netzwerk hinzufügen“:



- Es öffnet sich folgendes Fenster, dort folgende Daten eingeben und Netzwerk speichern:

Netzwerke > Ein neues Netzwerk hinzufügen

**Ein betrügerischer Netzwerkanbieter kann bezüglich des Status der Blockchain täuschen und Ihre Netzwerkaktivitäten aufzeichnen. Fügen Sie nur vertrauenswürdige Netzwerke hinzu.**

<b>Netzwerkname</b>	<b>Neue RPC-URL</b>
<input type="text" value="Ganache"/>	<input type="text" value="HTTP://127.0.0.1:7545"/>
<b>Ketten-ID</b> ⓘ	<b>Währungssymbol</b>
<input type="text" value="1337"/>	<input type="text" value="ETH"/>
	<small>Ticker symbol verification data is currently unavailable, make sure that the symbol you have entered is correct. It will impact the conversion rates that you see for this network</small>
<b>Block-Explorer</b> (Optional)	
<input type="text"/>	
<input type="button" value="Abbrechen"/>	<input type="button" value="Speichern"/>

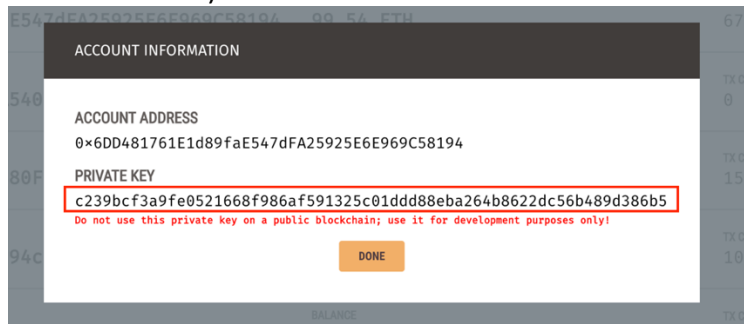
Die lokale Blockchain ist nun verbunden. Im letzten Schritt muss noch ein entsprechender Account verbunden werden. Dafür folgende Schritte befolgen:

- Ganache öffnen (Ganache Projekt muss gestartet sein) und das Schlüssel-Symbol einer der Adressen klicken:

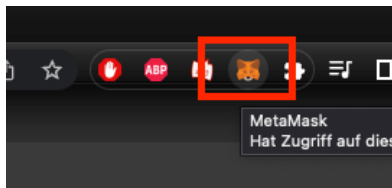
The screenshot shows the Ganache application window. At the top, there are tabs for ACCOUNTS, BLOCKS, TRANSACTIONS, CONTRACTS, EVENTS, and LOGS. Below these, there's a search bar and a status bar showing various metrics like CURRENT BLOCK, GAS PRICE, GAS LIMIT, HARDFORK, NETWORK ID, RPC SERVER, MINING STATUS, and WORKSPACE. The main area displays a list of accounts with columns for ADDRESS, BALANCE, TX COUNT, and INDEX. A red box highlights the key icon next to the first account.

ADDRESS	BALANCE	TX COUNT	INDEX
0x6DD481761E1d89faE547dFA25925E6E969C58194	99.54 ETH	67	0
0x13B8a03250Eb72BA540D2748b105A3D1B704D1cE	100.00 ETH	0	1
0xEec6f6613afB227280F5DcAf31D274B6bb0A1eC3	99.99 ETH	15	2
0x9a06214b4c24ee7E94ccC91fAC01E8a11Ffc0Cc0	99.99 ETH	10	3
0x7Fd1547C3B4F536859d74546f321A0E0a595607a	100.00 ETH	0	4
0x4aeaB2d60DaA6671bE0f0DF45495B70acb7ac125	100.00 ETH	0	5
0x47f485722CaF93a94EBD5C3B8ADBf7AC0543ACf8	100.00 ETH	0	6

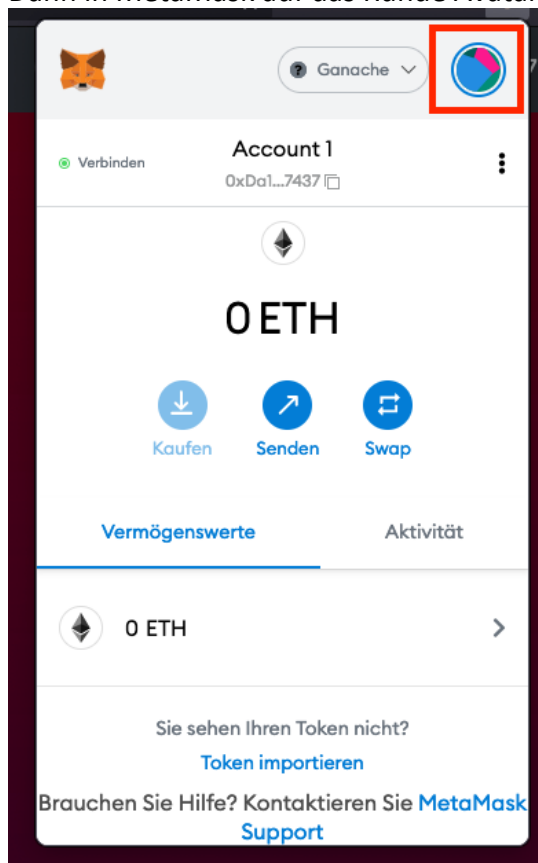
2. Das öffnet folgendes Fenster, dort den Private Key kopieren (dieser ist unterschiedlich):



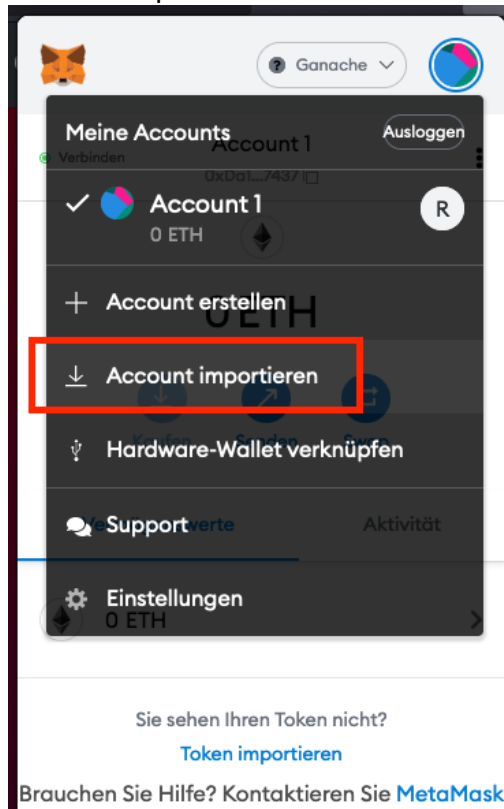
3. Dann zurück zum Browser und MetaMask im Browser öffnen durch Klicken des Icons:



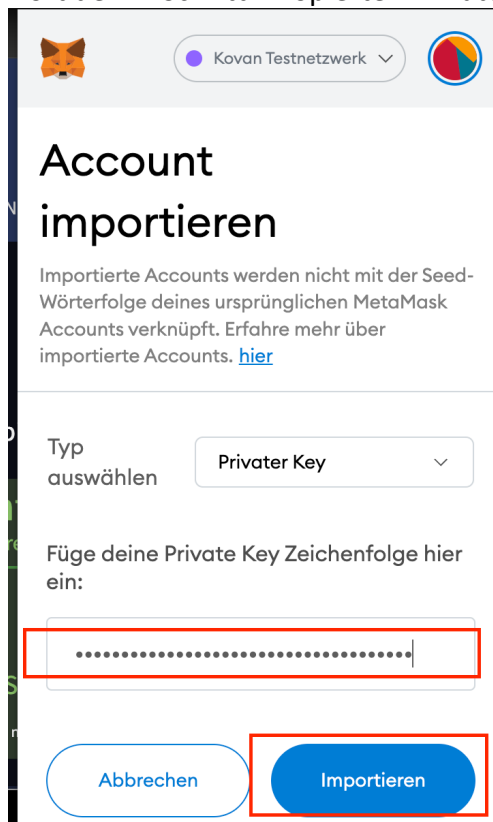
4. Dann in MetaMask auf das Runde Avatar-Bild klicken:



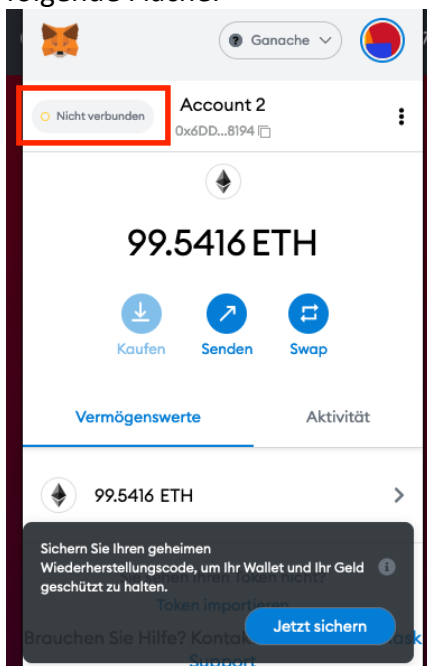
5. Account importieren:



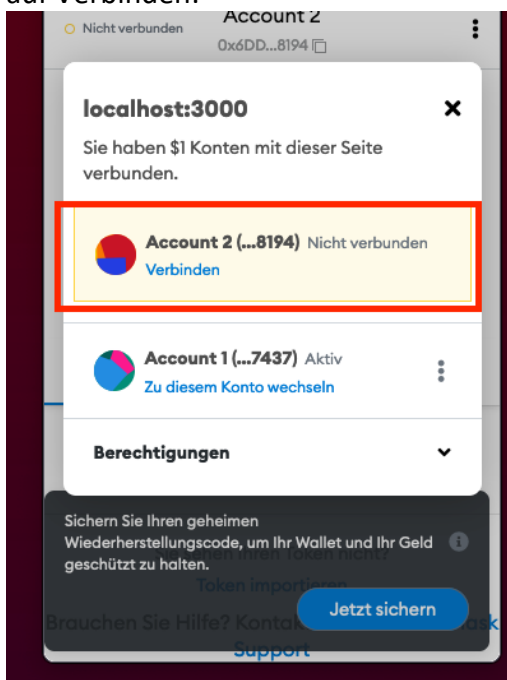
6. Dort den in Schritt 2 kopierten Private Key einfügen und Account importieren.



7. In MetaMask muss dann der Account noch verbunden werden durch Klicken auf folgende Fläche:

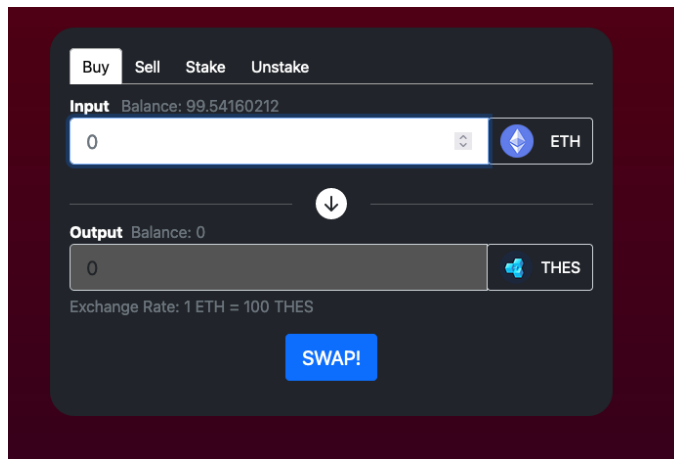


8. Im nächsten Fenster den oberen, nicht verbunden Account verbinden, durch Klicken auf Verbinden:



9. Die gestartete Anwendung unter [http://localhost:3000/DeFi\\_Application](http://localhost:3000/DeFi_Application) im Browser neu laden.

Der Account ist jetzt mit der Anwendung verbunden. Im Balance Feld über dem Input müsste das Guthaben an Ether angezeigt werden, wie im folgenden Bild zu sehen:



Die Installation ist abgeschlossen und die Anwendung kann vollständig genutzt werden.

## 4) Nutzungsanleitung für Funktionalitäten des Prototyps

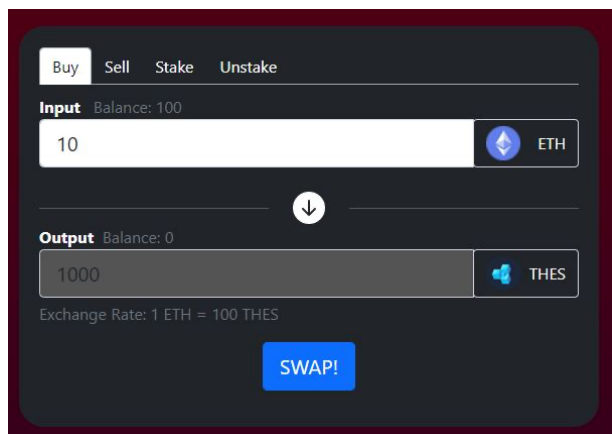
Im folgenden Kapitel wird erläutert, wie die Funktionalitäten des Prototyps verwendet werden können. Wichtig ist hierbei zu beachten, dass der Prototyp durch Variante 1 oder Variante 2 installiert und gestartet werden muss.

Hierbei muss ein Account in MetaMask verbunden sein, der über Test-Ether verfügt, die für die Funktionalitäten benötigt werden.

### 4.1) Thesis Token kaufen

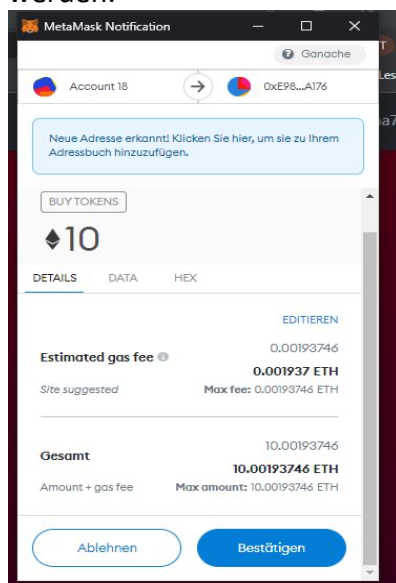
Die einzelnen Formulare können über die Tabs aufgerufen werden. Das Buy-Formular ermöglicht den Kauf von Thesis Token gegen Ether. Im Buy-Formular der Frontend-Anwendung wird über das Input-Feld der Etherbetrag festgelegt, der zum Kauf von Thesis Token verwendet werden soll (größer 0, kleiner als Balance an Ether). Im Output-Feld wird dann der auszahlende Betrag an Thesis Token angezeigt.

Durch Klicken auf „Swap“ wird die Transaktion gestartet.



The screenshot shows a web interface for buying Thesis Tokens. At the top, there are tabs: 'Buy', 'Sell', 'Stake', and 'Unstake'. The 'Buy' tab is selected. Below the tabs, there is an 'Input' section with a balance of 100. A text input field contains the number '10', and next to it is a button with the Ethereum logo and 'ETH'. Below this is a downward arrow icon. The 'Output' section shows a balance of 0 and a text input field containing '1000', with a button next to it showing the Thesis Token logo and 'THES'. Below the input fields, the 'Exchange Rate' is displayed as '1 ETH = 100 THES'. At the bottom, there is a large blue button labeled 'SWAP!'.

Die Transaktion muss dann bestätigt werden, wodurch Thesis Token gegen Ether gekauft werden.



The screenshot shows the MetaMask mobile app interface for confirming a transaction. At the top, it says 'MetaMask Notification' and 'Ganache'. Below that, it shows 'Account 18' and a transaction hash '0xE98...A176'. A message states: 'Neue Adresse erkannt! Klicken Sie hier, um sie zu Ihrem Adressbuch hinzuzufügen.' Below this is a 'BUY TOKENS' button. The transaction amount is shown as '10' with the Ethereum logo. There are tabs for 'DETAILS', 'DATA', and 'HEX'. Under 'DETAILS', there is an 'EDITIEREN' link. The 'Estimated gas fee' is shown as '0.00193746' and '0.001937 ETH'. The 'Site suggested' is 'Max fee: 0.00193746 ETH'. The 'Gesamt' (Total) is shown as '10.00193746' and '10.00193746 ETH'. At the bottom, there are two buttons: 'Ablehnen' (Reject) and 'Bestätigen' (Confirm).



Thesis Token wurden dann erworben (zu sehen im Balance-Feld neben dem Output), im Gegenzug wurde der Betrag an Ether abgezogen. Somit wurden Thesis Token gegen Ether zu einem bestimmten Wechselkurs gekauft.

The screenshot shows a web interface with a dark background. At the top, there are four tabs: 'Buy', 'Sell', 'Stake', and 'Unstake'. The 'Buy' tab is selected. Below the tabs, there are two input fields. The first field is labeled 'Input' and has a balance of 89.99870836. It contains the number '0' and is followed by a dropdown menu showing 'ETH'. The second field is labeled 'Output' and has a balance of 1000. It contains the number '0' and is followed by a dropdown menu showing 'THES'. Below these fields, there is a text label 'Exchange Rate: 1 ETH = 100 THES'. At the bottom, there is a blue button labeled 'SWAP!'.

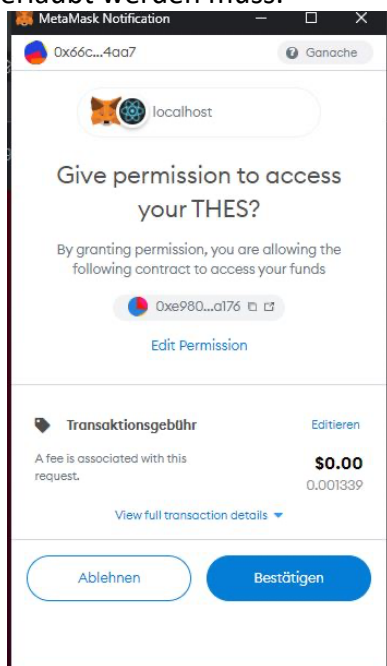
#### 4.2) Thesis Token verkaufen

Über das Sell-Formular können erworbene Thesis Token gegen Ether verkauft werden. Im Input-Feld wird der Tokenbetrag angegeben, der verkauft werden soll (größer 0, kleiner als Balance an Thesis Token). Im Output-Feld wird dann der auszuzahlende Betrag an Ether angezeigt.

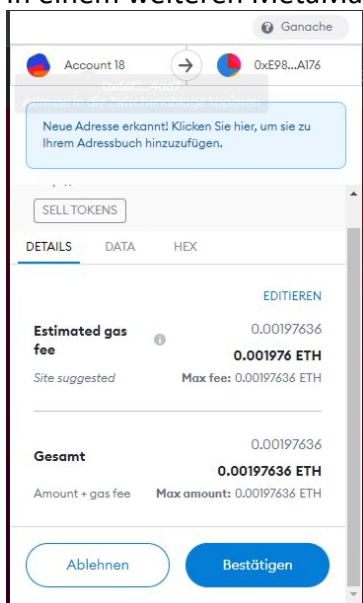
Durch Klick auf den „Swap“-Button wird die Transaktion gestartet.

The screenshot shows a web interface with a dark background. At the top, there are four tabs: 'Buy', 'Sell', 'Stake', and 'Unstake'. The 'Sell' tab is selected. Below the tabs, there are two input fields. The first field is labeled 'Input' and has a balance of 1000. It contains the number '500' and is followed by a dropdown menu showing 'THES'. The second field is labeled 'Output' and has a balance of 89.99870836. It contains the number '5' and is followed by a dropdown menu showing 'ETH'. Below these fields, there is a text label 'Exchange Rate 100 THES = 1 ETH'. At the bottom, there is a blue button labeled 'SWAP!'.

Dadurch wird folgendes MetaMask-Fenster geöffnet, in dem der Transfer von Thesis Token erlaubt werden muss.



In einem weiteren MetaMask-Fenster muss dann der Verkauf bestätigt werden.



Thesis Token wurden dann verkauft (zu sehen im Balance-Feld neben dem Output), im Gegenzug wurde der Betrag an Ether der Adresse gutgeschrieben. Somit wurden Thesis Token gegen Ether zu einem bestimmten Wechselkurs verkauft.

The screenshot shows a dark-themed interface with a top navigation bar containing 'Buy', 'Sell', 'Stake', and 'Unstake'. The 'Sell' tab is active. Below the navigation bar, the 'Input' section shows 'Balance: 500' and a text input field containing '0'. To the right of the input field is a blue button with a THES token icon and the text 'THES'. A downward arrow icon is centered between the input and output sections. The 'Output' section shows 'Balance: 94.99679818' and a text input field containing '0'. To the right of the output field is a blue button with an ETH token icon and the text 'ETH'. Below the output field, the text 'Exchange Rate 100 THES = 1 ETH' is displayed. At the bottom center is a blue button labeled 'SWAP!'.

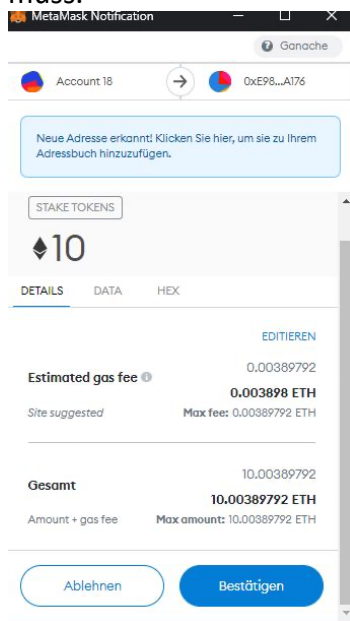
#### 4.3) Ether staken

Über das Stake-Formular können Ether zum Staking eingezahlt werden. Dafür muss im Input-Feld der einzuzahlende Betrag an Ether angegeben werden (größer 0, kleiner als Balance an Ether).

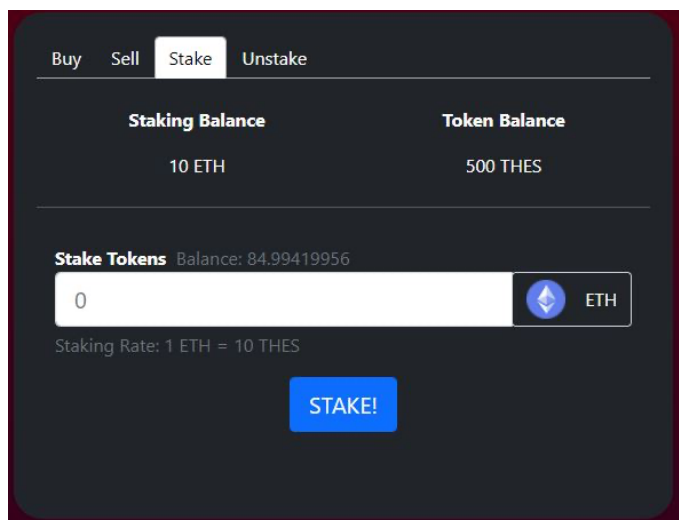
Durch Klicken auf den „Stake“-Button wird die Transaktion gestartet.

The screenshot shows a dark-themed interface with a top navigation bar containing 'Buy', 'Sell', 'Stake', and 'Unstake'. The 'Stake' tab is active. Below the navigation bar, there are two columns: 'Staking Balance' showing '0 ETH' and 'Token Balance' showing '500 THES'. Below these columns, the 'Stake Tokens' section shows 'Balance: 94.99679818' and a text input field containing '10'. To the right of the input field is a blue button with an ETH token icon and the text 'ETH'. Below the input field, the text 'Staking Rate: 1 ETH = 10 THES' is displayed. At the bottom center is a blue button labeled 'STAKE!'.

Wodurch ein MetaMask-Fenster geöffnet wird, in dem die Transaktion bestätigt werden muss.



Durch Bestätigung der Transaktion wird der Betrag an Ether als Staking-Balance gespeichert, die Staking-Balance findet sich im Formular. Der eingezahlte Betrag an Ether wird der Adresse abgezogen.



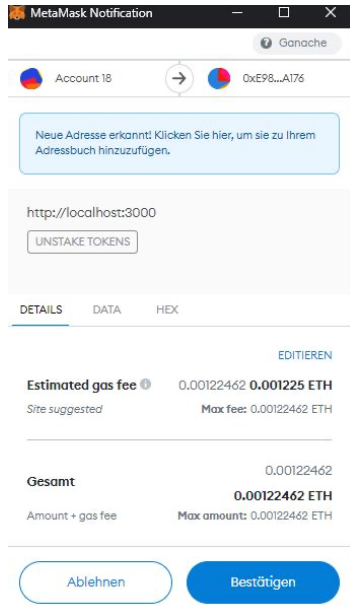
#### 4.4) Ether unstaken

Im Unstaking-Formular kann Ether ausgezahlt werden, das zum Staking eingezahlt wurde. Im Input-Feld kann der auszuzahlende Betrag an Ether angegeben werden (größer 0, kleiner als Balance an gespeicherte Staking-Balance).

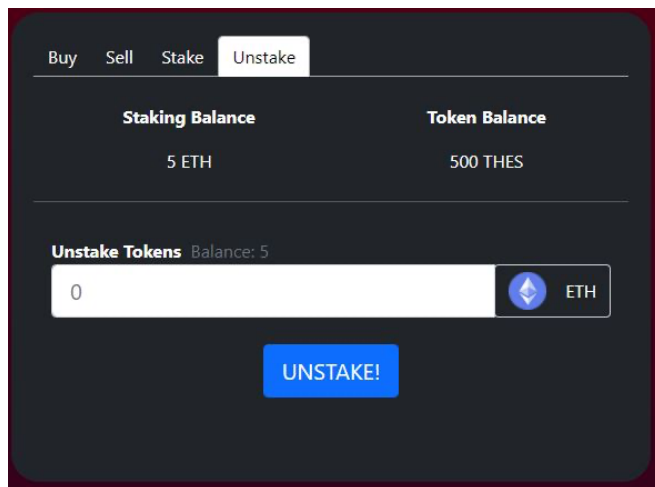
Durch Klicken auf den „Unstake“-Button wird die Transaktion gestartet.

The screenshot shows a dark-themed interface with a top navigation bar containing 'Buy', 'Sell', 'Stake', and 'Unstake'. The 'Unstake' tab is active. Below the navigation bar, there are two columns: 'Staking Balance' showing '10 ETH' and 'Token Balance' showing '500 THES'. Below these, there is a section for 'Unstake Tokens' with a sub-label 'Balance: 10'. An input field contains the number '5'. To the right of the input field is a dropdown menu showing an Ethereum icon and 'ETH'. Below the input field is a blue button labeled 'UNSTAKE!'.

Es öffnet sich ein Fenster in MetaMask, in dem die Transaktion bestätigt werden muss.



Der Betrag wurde dann der Staking-Balance der Adresse abgezogen und der Betrag an Ether wurde der Adresse gutgeschrieben.



#### 4.5) Tokenauszahlung starten

Die Funktionalität ist nur mit der lokalen Entwicklungsumgebung möglich (Variante 2), hierfür müssen die Anweisungen aus Kapitel 2 zur Installation verfolgt werden.

Diese Funktionalität wird nicht durch das Frontend ausgerufen. Stattdessen folgende Anweisungen folgen:

1. Im Terminal die Projekt-Directory öffnen.
2. Den Smart Contract-Folder im Terminal öffnen durch „cd smart\_contracts/“
3. Im nächsten Schritt folgenden Befehl im Terminal ausführen:  
„truffle exec scripts/issue-token.js“

Die Tokenauszahlung wird gestartet, die Investoren erhalten automatisch ihren Staking-Reward.