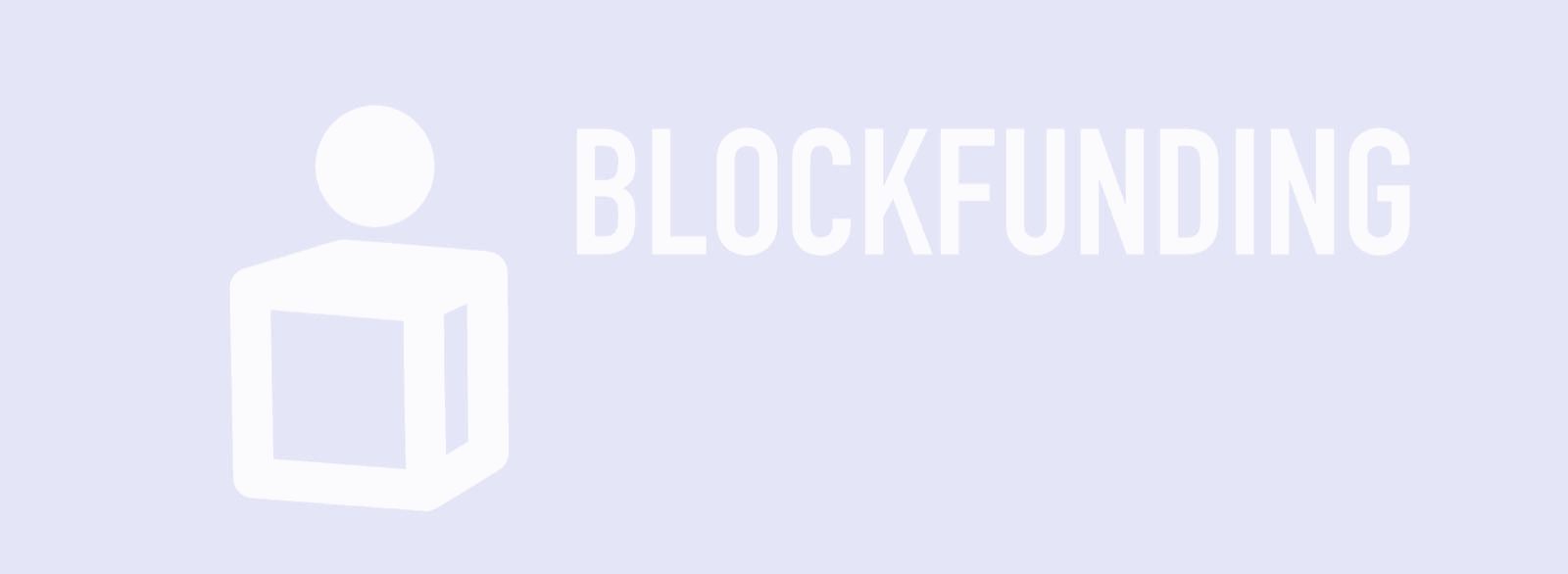
# Dokumentation Blockfunding

Eine Crowdfunding Plattform auf der Blockchain

<https://github.com/LucaLanzo/Blockfunding>

Luca Lanzo, Adrian Mück und Max Hild

## Einleitung

Der Traum, Crowdfunding durch eine dezentralisierte App durchzuführen und es somit möglich zu machen, Investitionen dezentral und ohne Gewinnbeteiligung Dritter empfangen und senden zu können, existiert schon lange. Das Problem hierbei: es wird eine Instanz benötigt, welche die Projekte überprüft und Investoren eine Plattform bietet um die Projekte sehen zu können. Bisher sind die Player in diesem Bereich daher gewinnorientierte Unternehmen wie Kickstarter und Indiegogo, welche Gebühren in Höhe von 4-9% (Indiegogo) erheben. Ein Ansatz, um eine unabhängige Plattform auf dem Markt zu schaffen, ist das Konzept einer DAO (Dezentrale Autonome Organisation). Der Grundgedanke von DAO ist es, dass die Organisation eines Unternehmens auch demokratisch und dezentral organisiert sein kann. So soll bei einer DAO der Wille der Mehrheit im Mittelpunkt stehen um demokratisch zu entscheiden, wie die Mittel des Unternehmens eingesetzt werden sollen.

Mit The DAO sollte dieser Traum bereits im Jahre 2016 wahr werden. Mit großen Vorschusslorbeeren startete das Projekt des deutschen Informatikers Simon Jentzsch, der den Code für das Projekt auf GitHub publizierte, um die Weiterentwicklung mit der Community zu ermöglichen (1). Das Geld, welches das Projekt durch die Finanzierung erhielt, wurden in DAO Token umgewandelt. Ein Ether war 100 DAO Token wert. Sollte ein Projekt finanziert werden, musste ein Antrag eingereicht werden, über den die Shareholder des DAO entschieden. Ab 20% Zustimmung unter den Shareholdern sollte das Geld an die Projekte überwiesen werden. Da die DAO Token nicht von Anfang an investiert wurden, sondern der anfängliche Verkauf drei Wochen anhielt, konnte das Projekt einen unvergleichlichen Hype generieren, der es ermöglichte mehr als 140 Millionen US Dollar zu generieren. Durch einen Fehler in der split Funktion des Programms, die das Abheben von Geld immer und immer wieder ermöglichte, konnte ein Hacker jedoch 3,6 Millionen Ether, das damalige Äquivalent von 50 Millionen Dollar, stehlen (2). Das war ein so großer Anteil an der Gesamtheit des damaligen Ethers, dass sogar ein hard fork nötig war, welcher die Kryptowährung Ethereum Classic erzeugte, die bis heute existiert. Hierfür wurde die Frist von 28 Tagen genutzt, bei dem alle DAO Transaktionen rückgängig gemacht werden konnten, sodass die User ihr gestohlenes Geld als Ethereum Classic zurückerhalten konnten.

Die Idee des Web3 generell ist es, das Internet wieder privater und unabhängiger von großen Organisationen zu machen. Aufgaben der Nutzer Verifikation, Transaktionen, der Speicherung und der Sicherheit, die aktuell von großen Unternehmen wie Google, Facebook oder Amazon Web Services übernommen werden, sollen in Blockchain Anwendungen dezentral organisiert werden. Durch die dezentrale, automatisierte Organisation können gewinnorientierte Unternehmen und Investmentfirmen, mit eigenen Interessen, die nicht unbedingt im Sinne der Nutzenden sind, durch dezentralisierte Anwendungen ersetzt werden. Durch Vorfälle wie den DAO Hack und durch die Volatilität vieler Kryptowährungen gibt es jedoch leider aktuell Misstrauen in Kryptowährungen. Viele Menschen zweifeln an der Zukunftsfähigkeit und der Sicherheit solcher Systeme. Daher kam uns die Idee, eine sichere Crowdfunding Plattform auf der Grundlage der Blockchain zu entwickeln, um zu zeigen, dass Kryptowährungen im alltäglichen Gebrauch einsetzbar sind.

## Problemstellung und Zielsetzung

Die Problemstellung, vor der wir uns sehen, ist:

Wie können Crowdfunding Kampagnen mit Kryptowährungen sicher und zuverlässig durchgeführt werden? Im Rahmen dieses FWPMs soll daher die Grundstruktur einer Plattform für das Crowdfunding implementiert werden. Dafür wollen wir die Projektinhaber über Smart Contracts verifizieren, anstatt eine Prüfung durch ein drittes Unternehmen durchzuführen. Die Kryptowährung Ethereum soll als erstes implementiert werden. Mit dieser Kryptowährung können sichere Transaktionen getätigt werden, die durchgeführt werden, sobald der vom Ersteller des Projekts festgelegte Schwellenwert erreicht wird. Über Erweiterungsmöglichkeiten wie den ERC20 Smart Contract planen wir später die Implementierung von weiteren Kryptowährungen, insbesondere von im Preis stabilen Stablecoins. Erste Priorität in unserem Projekt haben die Bereitstellung einer einfachen Benutzeroberfläche, um in Projekte investieren zu können, sowie ein sicherer Smart Contract. Die Plattform soll langsam wachsen, damit die Sicherheitsfeatures äquivalent zum Marktvolumen wachsen können.

Eine weitere Problemstellung liegt in der Natur der implementierten Kryptowährung Ethereum. Sogenannte Gaskosten werden erhoben. Das sind Gebühren für CRUD Aktionen auf der Blockchain, also auch Transaktionen und das Lesen von Daten. Hierbei müssen wir also darauf achten, dass Zugriffe auf die Blockchain Gaskosten-effizient ausgeführt werden. Durch optimierte Zugriffe wie Lazy Evaluation sollen die Gaskosten so niedrig wie möglich gehalten werden, um das Investieren gerade bei kleineren Projekten so attraktiv wie möglich zu halten.

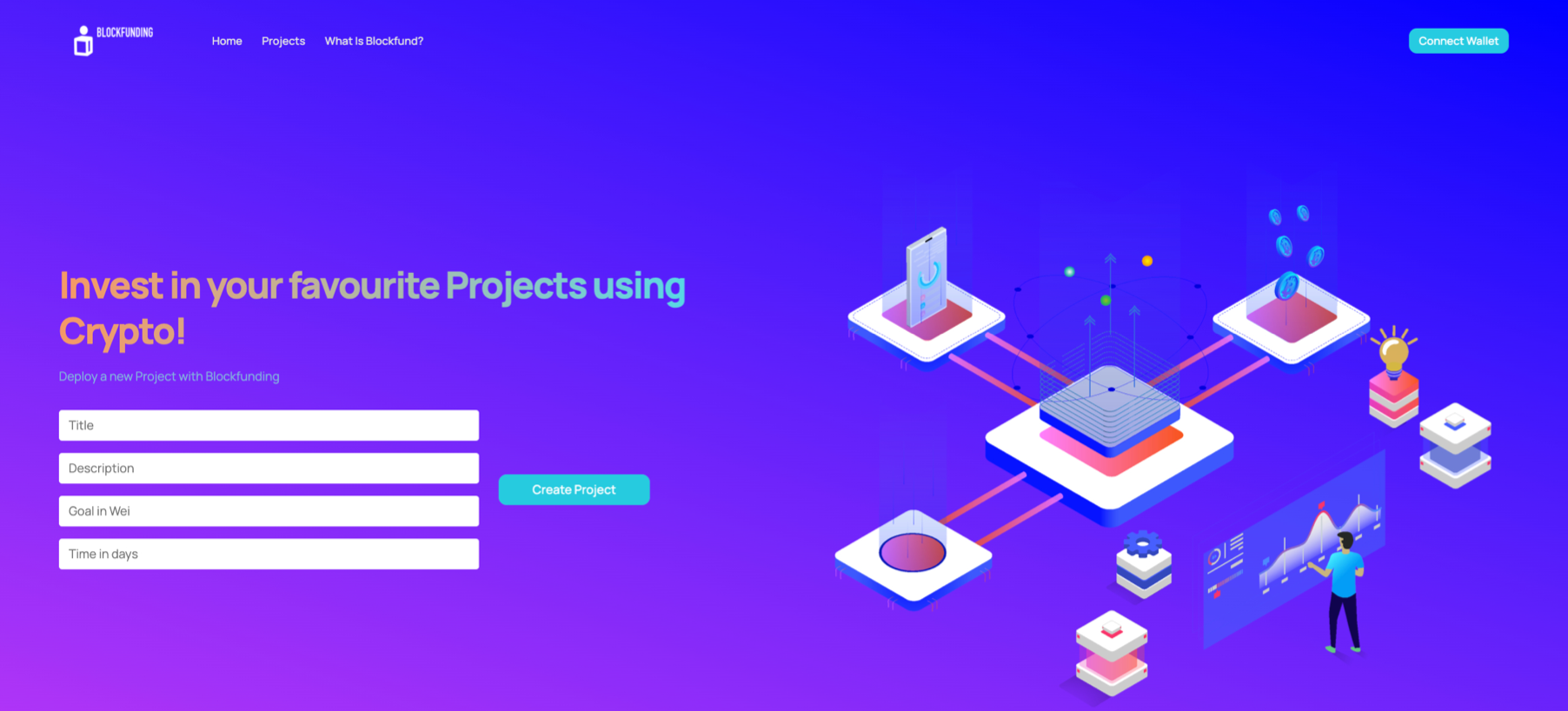
## Lösung

Das Projekt wurde mit den Programmiersprachen Solidity (für die Smart Contracts) und JavaScript, genauer gesagt den JavaScript Libraries React.js und Web3.js implementiert. Die DApp ermöglicht es, Projekte anzulegen und bestehende Projekte anzuzeigen. Außerdem ist in den Smart Contracts die Funktionalität verbaut, Projekte per Ethereum zu finanzieren. Zusätzlich können sich Creator, nach Erreichen des Ziels, ohne Gebühren an Dritte auszahlen lassen. Sollte das Ziel nicht in der gegebenen Zeit erreicht werden, können Unterstützer ihr Geld zurückfordern.

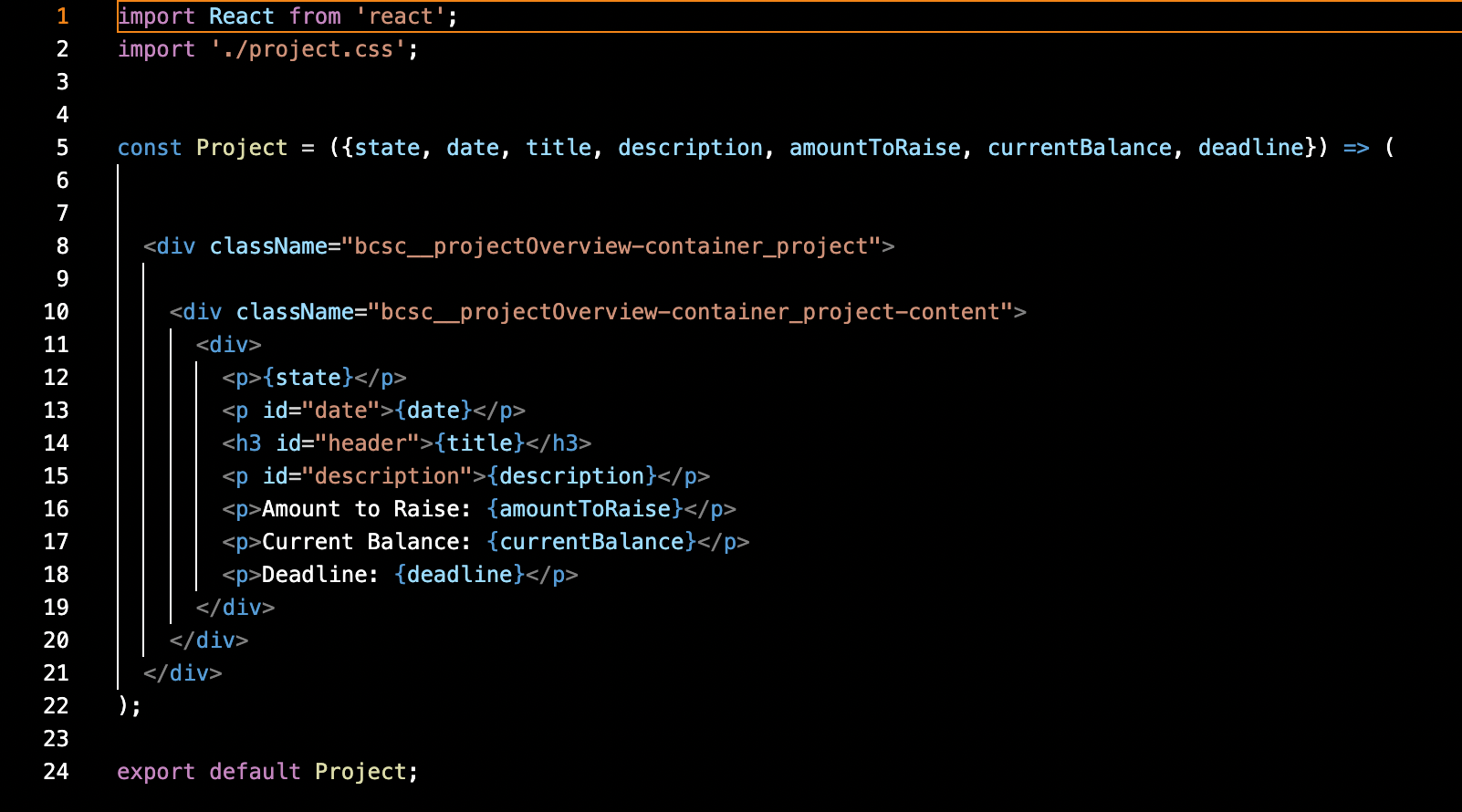
### 

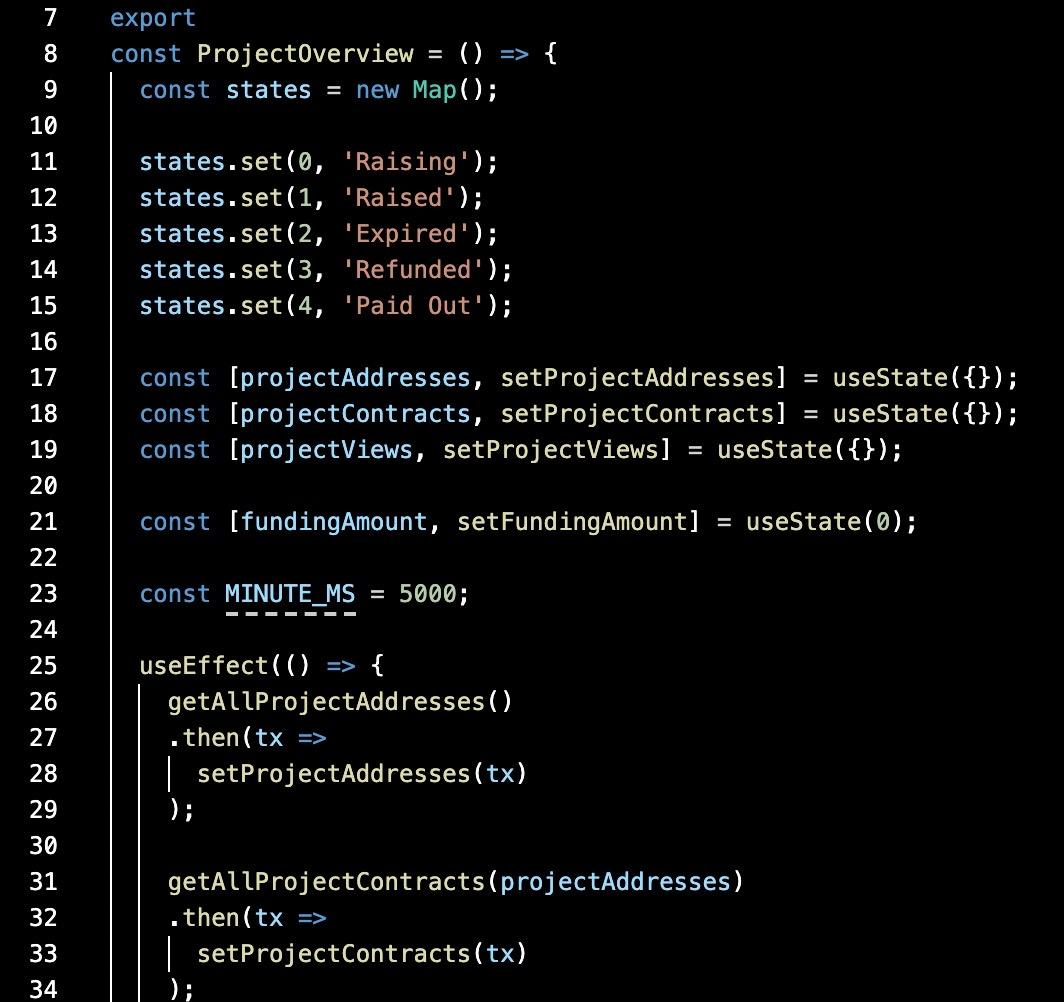
### Frontend: React

Das Frontend ist mit React.js implementiert worden. Hier wurde der besondere Fokus auf eine schnelle Ladegeschwindigkeit sowie ein ansprechendes, einfach zu verstehendes Design gelegt.



Projekte auf der Seite sind ein Beispiel für ein mit einem React Component angelegter Seitenabschnitt. Diese Components werden definiert, und können dann über die Funktionen auf verschiedenen Teilen der Seite ausgespielt werden. Hierfür werden Sie über die index.js Datei im Components Ordner exportiert.



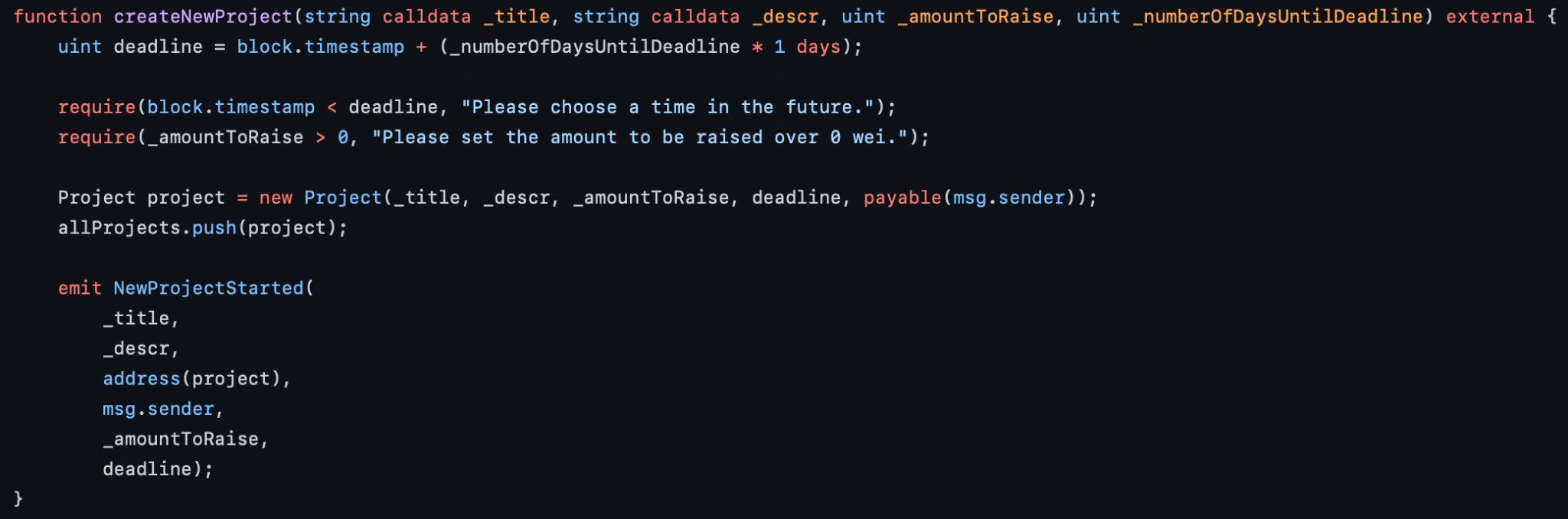
Container sind Components mit erweiterter Logik (4). Diese erzeugen HTML Code je nachdem welcher Input gegeben ist. Im folgenden Beispiel sieht man einen Ausschnitt aus dem Container ProjectOverview, welche die Component Project aufruft und aus den Daten der Projekte eine Übersicht erzeugt.  


### Backend: Smart Contracts

#### Smart Contract Crowdfunding

Der Smart Contract für das Crowdfunding wurde in Solidity programmiert. Der Vertrag heißt Crowdfunding und agiert als Container für die Projekte. Der Crowdfunding Vertrag beinhaltet eine Liste von Projektverträgen welche bei der Initialisierung leer ist.

Diese kann durch die Methode *createNewProject* gefüllt werden. Dabei wird ein neuer Projekt Contract erzeugt und gespeichert.



Um als User alle Projekte bekommen zu können wurde die *viewAllProjects* Methode zur Verfügung gestellt, welche eine Liste von Projektadressen zurückgibt.

#### 

#### Smart Contract Project

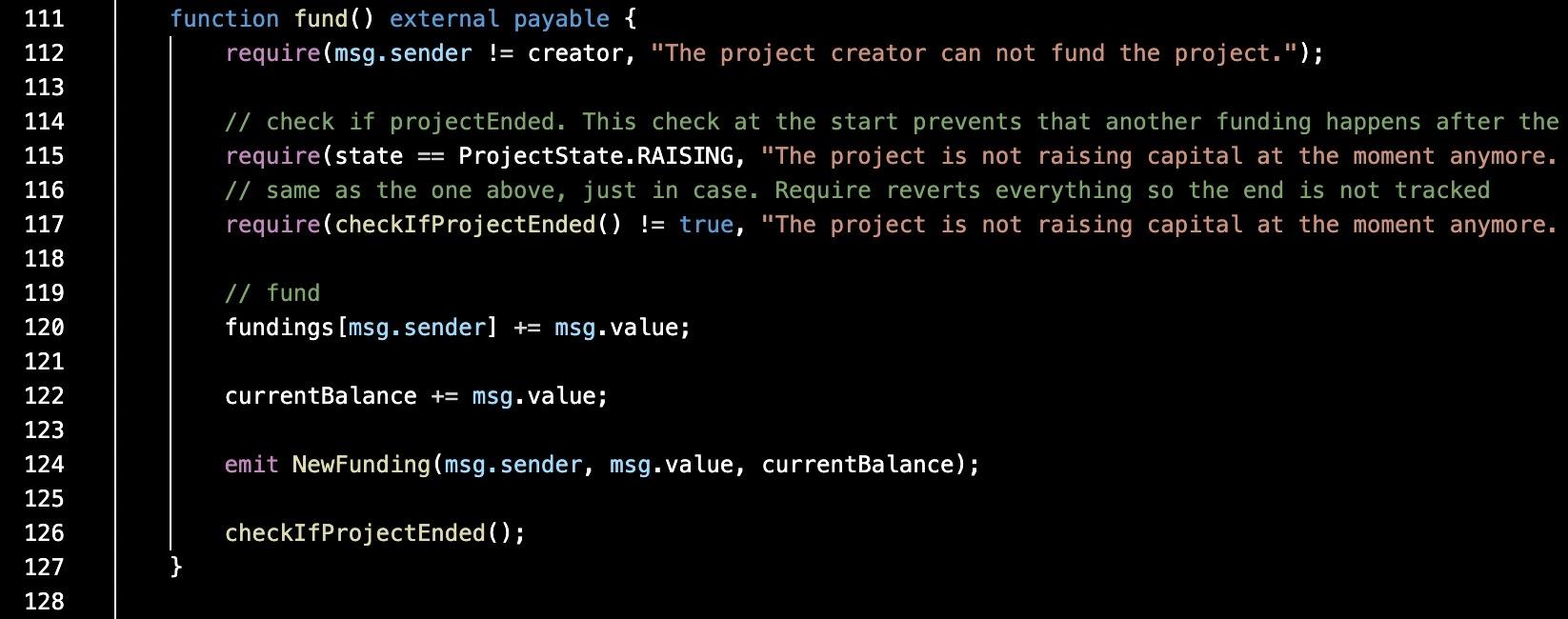
Im Inneren des Smart Contracts Crowdfunding befindet sich der Smart Contract eines Projekts. Er beinhaltet die Projekte mit den Daten der Projekte. Diese werden direkt auf der Blockchain gespeichert.



Nach der Erzeugung stehen den Nutzern verschiedene Funktionen zur Verfügung. Backer können über die *fund* Methode das Projekt unterstützen und sobald das Ziel erreicht wurde kann über die *payOut* Methode das Geld an der Creator ausgezahlt werden. Sollte die Deadline erreicht werden ohne, dass genug Geld eingenommen wurde, steht den Backern eine *refund* Methode zur Verfügung mit welcher sie ihr gezahltes Geld zurückbekommen.

Bei allen Methoden wird auf die Sicherheit geachtet. Sollte es bei Transaktionen zu Fehlern kommen wird jede Transaktion rückgängig gemacht. Gleichzeitig gibt es Restriktionen die sicherstellen, dass Methoden geblockt werden, sollte das Projekt in einem anderen Status sein. Wenn das Projekt beispielsweise noch im Status “Erhebung” ist, wird niemals die *refund* oder *payout* Methode aufgerufen werden können. Es wird auch darauf geachtet, dass manche Methoden nur vom Creator oder Backer aufgerufen werden können.

Zur Aktualisierung des Status des Projekt wird eine Lazy Evaluation genutzt, da es keine Möglichkeit gibt, das Projekt automatisch zur Deadline enden zu lassen. So wird bei jedem funding die Methode *checkIfProjectEnded* ausgeführt, welche überprüft, ob genug Krypto eingenommen wurde oder die Deadline erreicht wurde. Mithilfe des Status (Raising, Raised, Expired, …) können andere Methoden feststellen an welchem Punkt sich das Projekt befinden und so variabel auf Änderungen reagieren.



Mithilfe der viewProject Funktion kann eine genauere Beschreibung des jeweiligen Projekts angezeigt werden. Hierbei handelt es sich um eine View Methode, das heißt es muss nicht auf den internen Speicher des Smart Contracts zugegriffen werden, sondern diese Eigenschaften können aus der sogenannten Calldata gelesen werden. Die Calldata ist eine auch von Methoden außerhalb des Smart Contracts sichtbare Möglichkeit, in einem 64 Bit Hash Daten über den Contract zu speichern für die nicht auf die Blockchain zugegriffen werden muss. Das spart Gaskosten und erhöht die Sicherheit bei Zugriffen. So kann im Frontend auf Knopfdruck immer der neueste Status des Projekts angezeigt werden, ohne dass Gaskosten gezahlt werden müssen.

### 

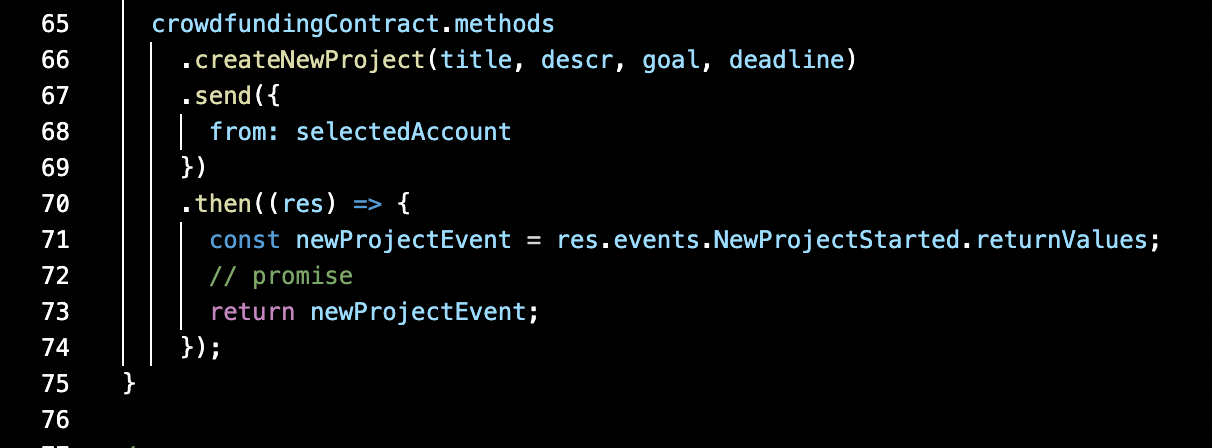
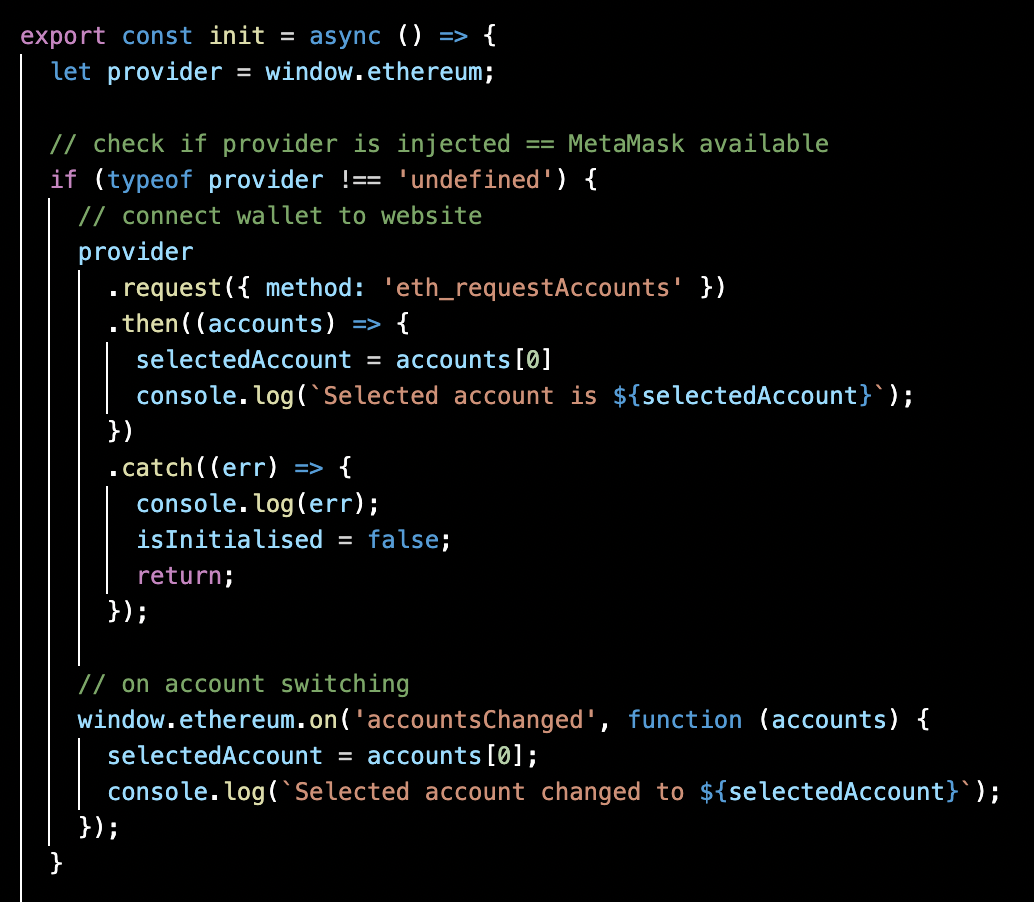
### Web3.js - Konnektivität zwischen Backend und Frontend

Um die Smart Contracts und das Frontend zu verbinden, wurde ein Web3.js Script eingefügt. Dieses enthält Methoden, um die Daten welche im Frontend in ein Formular eingegeben werden, an den Smart Contract weiterzugeben sowie um den Lesezugriff zu ermöglichen. Außerdem werden auch noch die Transaktionen *fund*, *payOut* und *refund* zur Verfügung gestellt.



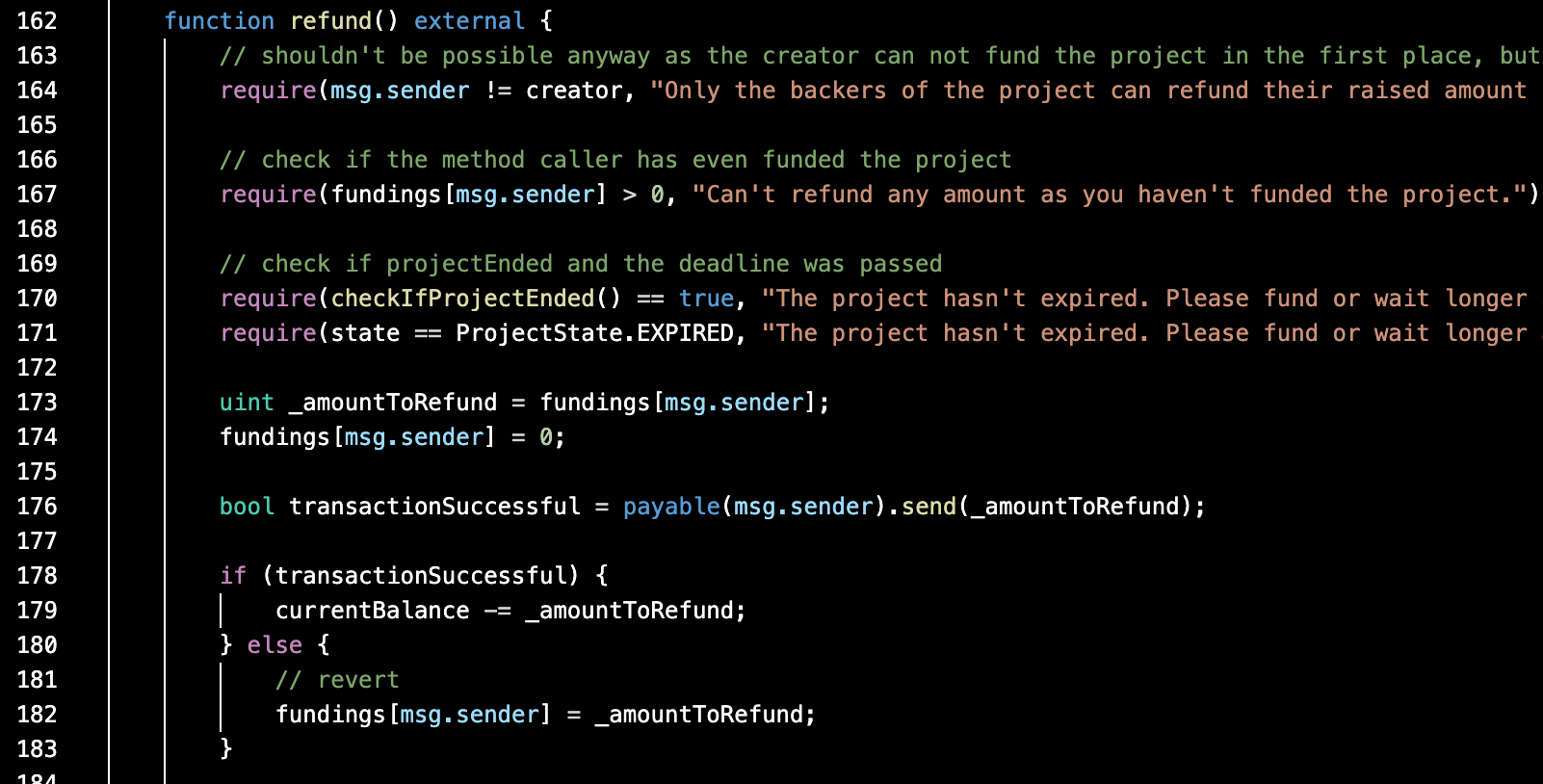
Mit der Funktion createNewProject wird zuerst gewartet, bis der Smart Contract initialisiert wurde. Das verhindert eine Fehlermeldung wenn der Contract sich noch nicht zurückgemeldet hat, aber bereits versucht wird Funktionen auszuführen.

Um mit dem Contract zu interagieren ist es nötig eine ABI mitzuschicken. Diese ist eine Zusammenfassung der Funktionen eines Smart Contracts. Durch Truffle werden die ABI`s der Smart Contracts im Ordner build automatisch bei einem migrate erstellt.

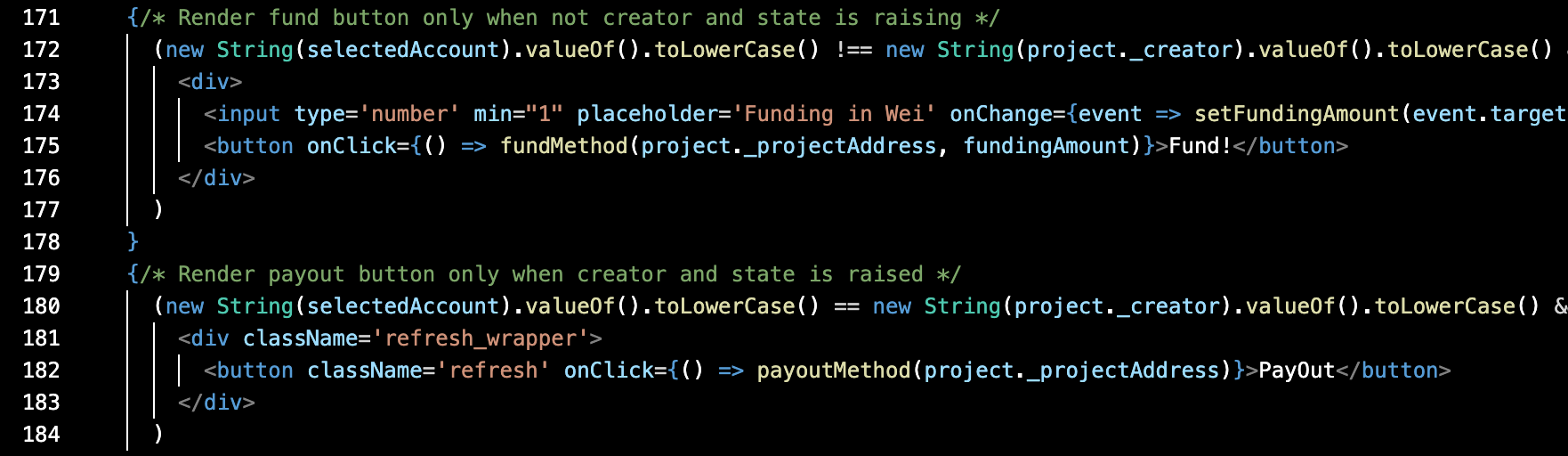
Außerdem muss aus Sicherheitsgründen der Provider (für alle Zugriffe auf die Blockchain) sowie der Signer (für Schreibzugriffe auf die Blockchain) mitgeschickt werden. Diese werden in der Datei Web3Client.js über Funktionen abgefragt. Der Signer wird wie im 2. Beispiel, der Funktion createNewProject, über den Befehl from: selectedAccount.

### Sicherheit

Der Ersteller des Projekts sieht eine andere Ansicht als Benutzer, die Projekte unterstützen oder einen Refung erhalten wollen. Im Backend wird dafür bereits über verschiedene Require Bedingungen, die zum Beispiel abfragen ob der Benutzer das Projekt überhaupt unterstützt hat oder ob das Projekt schon geendet ist, überprüft ob die Auszahlung oder Einzahlung valide ist.



Zusätzlich wird im Frontend zum Beispiel an den Buttons ebenfalls der Status des Benutzers der Seite (mit der Prüfung ob es sich um einen Project Creator handelt) sowie der Bezahlstatus des Nutzers überprüft, um eine falsche Auszahlung nicht zu erlauben.



### 

### 

### Fazit

Im Rückblick war eine der Hürden, dass die Verbindung zwischen Smart Contracts und dem Frontend mit Web3.js komplexer war, als wir es zu Beginn eingeplant hatten. Außerdem gab es das Problem, dass CRUD Operationen in den Smart Contracts länger dauerten, als das Frontend zu laden. Deshalb mussten wir einige asynchrone Zugriffe verwenden.

Unsere Lösung ist insbesondere deshalb interessant, weil sie den Kontakt zwischen den Projekten, die um Unterstützung bitten, und den Unterstützern sehr direkt implementiert hat. Zusätzlich wurde durch die view Methoden und das das Lazy Loading erreicht, dass die Transaktionsgebühren so gering wie möglich gehalten werden.

Im Vergleich zu anderen Crowdfunding Plattformen fehlt es bei vielleicht noch an dem Anreiz für Unterstützer in Projekte zu investieren. Eine Möglichkeit wäre das Ausgeben von Tokens für jede Unterstützung. Außerdem sollten noch Bilder oder Videos von den Projekten implementiert werden.

Auch fehlt eine Unterstützung für unterschiedliche Kryptowährungen. Gerade in diesem Bereich wollen die Projekte natürlich so viel Geld wie möglich umsetzen, weshalb die Gaskosten von Ethereum uns eher dazu neigen lassen in Zukunft auf Layer 2 Kryptowährungen wie Loopring zu setzen.

Ansonsten glauben wir, dass die Crowdfunding Plattform eine sinnvolle Sache für Einzelpersonen, Innovateure sowie Erfinder ist und mit einer Layer 2 Implementierung vor allem Einzelpersonen aus ärmeren Länder die Möglichkeit gegeben wird, ihre Ideen und Träume mit einer guten finanziellen Unterstützung zu verwirklichen.

Uns hat das Projekt sehr viel Spaß gemacht und wir haben viel gelernt. Wir werden weiterhin an dezentralen Apps und Smart Contracts arbeiten, um unsere

### 

### Quellen

1. Alle Codes zum DAO sind unter <https://github.com/blockchainsllc/DAO> zu finden.
2. <https://www.zeit.de/digital/internet/2016-06/the-dao-blockchain-ether-hack?utm_referrer=https%3A%2F%2Fwww.google.com%2F>
3. <https://ethereum.org/en/developers/docs/standards/tokens/erc-20/>
4. <https://www.section.io/engineering-education/container-components-in-react/>