Hochschule **RheinMain**
Fachbereich Design Informatik Medien
Studiengang Medieninformatik

Abschlussarbeit
zur Erlangung des akademischen Grades
Bachelor of Science - B. Sc.

Self Sovereign Identity im grünen Bereich

Vorgelegt von	Chand Mandru
am	19.03.2025
Referent:	Prof. Dr. Philipp Schaible
Korreferent:	Prof. Dr. Holger Hünemohr

Eigenständigkeitserklärung
(gem. ABPO, Ziff. 4.1.5.4 bzw. RPO, §19.6)

Ich bin verantwortlich für die Qualität des Inhalts dieser Arbeit, den ich mit geeigneten wissenschaftlichen Quellen belegt bzw. gestützt habe. Die aus fremden Quellen direkt oder indirekt übernommenen Texte, Gedankengänge, Konzepte, Grafiken, technischen Inhalte und Ähnliches in meinen Ausführungen habe ich eindeutig gekennzeichnet und mit vollständigen Verweisen auf die jeweilige Quelle versehen. Alle weiteren Inhalte der Arbeit (Textteile, Abbildungen, Tabellen etc.) ohne entsprechende Verweise stammen im urheberrechtlichen Sinn von mir.
Ich versichere außerdem, dass ich die Bachelor-Arbeit selbständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel benutzt sowie keine unerlaubte fremde Hilfe in Anspruch genommen habe.

Ich versichere, dass ich KI-Tools lediglich als Hilfsmittel verwendet habe und in der vorliegenden Arbeit mein gestalterischer Einfluss überwiegt. Ich verantworte die Übernahme jeglicher von mir verwendeter KI-generierter Inhalte vollumfänglich selbst.

Die vorliegende Arbeit wurde bisher weder im In- noch im Ausland in gleicher oder ähnlicher Form einer anderen Prüfungsbehörde vorgelegt. Mir ist bekannt, dass ein Verstoß gegen die genannten Punkte als Täuschungsversuch gelten und prüfungsrechtliche Konsequenzen haben kann. Insbesondere kann es dazu führen, dass die Leistung nicht bestanden ist und dass bei mehrfachem oder schwerwiegendem Täuschungsversuch eine Exmatrikulation droht.

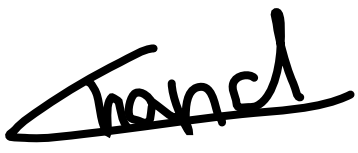
Ort, Datum: Unterschrift:
Kelsterbach, 18.03.2025

Hiermit erkläre ich mein Einverständnis mit den im folgenden aufgeführten Verbreitungsformen dieser Abschlussarbeit:

Verbreitungsform	JA	NEIN
Veröffentlichung des Titels der Arbeit im Internet	×	
Veröffentlichung der Arbeit im Internet	×	

Ort, Datum:
Kelsterbach, 18.03.2025

Unterschrift:



Abkürzungen

DID	Decentralized Identifier
DIDComm	Decentralized Identifier Communication
JWS	JSON Web Signature
JWT	JSON Web Token
SSI	Self-Sovereign Identity
VC	Verifiable Credential
VDR	Verifiable Data Registry
W3C	World Wide Web Consortium
ZKP	Zero-Knowledge-Proof

Inhaltsverzeichnis

1. Einführung

In der vorliegenden Arbeit wird das Konzept der Self-Sovereign Identity(SSI) untersucht. Ziel dieser Arbeit ist es, die Funktionsweise und Potenziale von SSI zu beleuchten und die Technologie anhand einer Fallstudie mit dem Partnerunternehmen „E-Systems“ praktisch zu erproben. Im Fokus steht die Analyse, wie SSI in realen Szenarien eingesetzt werden kann und welche Herausforderungen sowie Chancen bei der Implementierung dieser Technologie bestehen.

Die Arbeit verfolgt zudem die Zielsetzung, zu evaluieren, wie SSI zum aktuellen Zeitpunkt implementierbar ist und welche Voraussetzungen erfüllt sein müssen, um eine erfolgreiche Integration in bestehende Systeme und Prozesse zu ermöglichen. Im Rahmen dieser Untersuchung werden sowohl technologische als auch praktische Aspekte berücksichtigt, die für die Realisierung von SSI entscheidend sind. Dabei wird nicht nur die Theorie, sondern auch die Praxis der Technologie in einer konkreten Anwendungsumgebung analysiert, um ein ganzheitliches Verständnis für SSI zu entwickeln.

Teil der Zielsetzung dieser Arbeit liegt in der Entwicklung eines funktionstüchtigen Prototyps, der im Rahmen einer Fallstudie demonstrieren soll. Der Prototyp soll aufzeigen, wie SSI genutzt werden kann, um die Überprüfung und Dokumentation länderspezifischer Umweltkriterien für importierte Produkte effizienter, fehlerfreier und transparenter zu gestalten. Dabei werden die Stärken und Schwächen von SSI hinsichtlich Automatisierung, Sicherheit, Effizienz und Fehleranfälligkeit untersucht.

Durch diese Fallstudie soll ein praktischer Einblick gewonnen werden, wie SSI in realen Anwendungsfällen abschneidet und ob es als zukunftsfähige Technologie zur Lösung aktueller Herausforderungen im Bereich der Produkt-Authentifizierung und Umweltkonformität beitragen kann.

Aktuell bestehen im Bereich der Produkt-Authentifizierung und der Überprüfung länderspezifischer Umweltkriterien für importierte Produkte verschiedene Probleme, die die Effizienz und Genauigkeit dieses Prozesses beeinträchtigen. Ein zentrales Problem ist die aufwendige manuelle Behandlung von Authentifizierungsdokumenten, die für jedes Land spezifisch sind. Diese Dokumente müssen manuell geprüft und mit anderen Unterlagen verglichen werden, was einen erheblichen Arbeitsaufwand bedeutet. Die Kommunikation und Abstimmung zwischen den verschiedenen Parteien, wie beispielsweise Unternehmen, Behörden und Prüfstellen, ist ebenfalls ein komplexer Prozess, der oft zu Verzögerungen führt.

Ein weiteres Problem besteht in der mangelnden Vollständigkeit und Richtigkeit der Dokumente, was wiederum die Wahrscheinlichkeit von Fehlern erhöht. Der hohe Anteil an manueller Arbeit, die von Menschen durchgeführt wird, macht den gesamten Prozess anfällig für Fehler. Die Ineffizienz des Prozesses ergibt sich aus der Notwendigkeit, mehrere Parteien oder Organisationen in den Ablauf einzubinden, was zu langwierigen Kommunikationsketten und einer hohen Bearbeitungszeit führt.

Die Konsequenzen dieser Probleme sind weitreichend: Verzögerungen bei der Überprüfung und Validierung von Umweltkriterien können zu Lieferverzögerungen und Handelshemmnissen führen. Zusätzliche Kosten entstehen durch die manuelle Bearbeitung und das Nachverfolgen von Dokumenten. In einigen Fällen können Fehler in den Dokumenten oder deren Prüfung auch rechtliche Konsequenzen nach sich ziehen, wie Bußgelder oder Handelsstreitigkeiten. Darüber hinaus führt die mangelnde Transparenz im Prozess zu einem Vertrauensverlust bei den Beteiligten, da es schwer nachvollziehbar wird, wie die Entscheidungen zur Konformität eines Produkts getroffen wurden.

Um die in dieser Arbeit erforschte Fallstudie in einen praktischen Kontext zu setzen, wird nun kurz das Dienstleistungsunternehmen „E-Systems“ vorgestellt. E-Systems ist ein spezialisiertes Unternehmen, das es seinen Kunden ermöglicht, ihre Produkte weltweit zu verkaufen, ohne sich mit den komplexen Anforderungen der Product Compliance oder der Herstellerverantwortung befassen zu müssen. Das Unternehmen übernimmt alle notwendigen Schritte für die Konformität eines Produkts, angefangen bei der Zusammenstellung der erforderlichen Dokumente über die Registrierung in nationalen Systemen bis hin zur Bereitstellung von Kennzeichnungen und Kostenanalysen.

E-Systems bietet eine ideale Plattform, um die Nutzung von SSI im Bereich der Produktkonformität zu untersuchen. SSI könnte die Arbeit von E-Systems erheblich verbessern, indem es sichere, verifizierbare und digital signierte Daten in Form von Verifiable Credentials(VCs) bereitstellt. Dies würde den Prozess der Produktverifizierung und die Sicherstellung der Konformität mit Umweltvorgaben beschleunigen, gleichzeitig aber auch die Genauigkeit und Nachvollziehbarkeit erhöhen.

Im Kontext dieser Thesis zeigt E-Systems, wie SSI praktisch eingesetzt werden kann, um die Einhaltung von Umweltstandards bei Importen und Exporten zu gewährleisten. Die Integration von SSI könnte das Potenzial von E-Systems erwei-

tern und als Fallstudie dienen, um zu zeigen, wie digitale Identitäten bestehende Compliance-Prozesse verbessern und nachhaltiger gestalten können.

SSI ist ein Konzept zur Verwaltung von digitalen Identitäten, das es Individuen und Organisationen ermöglicht, ihre Identitätsdaten selbst zu kontrollieren, ohne auf zentrale Behörden oder Institutionen angewiesen zu sein. Im Gegensatz zu traditionellen Identitätssystemen, bei denen Daten in zentralen Datenbanken gespeichert werden, basiert SSI auf einem dezentralen Ansatz, bei dem die Nutzer die Kontrolle über ihre Identitäten und die damit verbundenen Informationen haben. Diese Identitäten werden durch sogenannte Decentralized Identifiers(DIDs) dargestellt und sind digital signiert und verifiziert.

Ein zentrales Element von SSI sind die VCs, die es ermöglichen, verifizierbare Nachweise über die Identität oder Qualifikationen eines Nutzers zu erbringen. Diese Nachweise können von vertrauenswürdigen Instanzen, wie etwa Universitäten oder staatlichen Behörden, ausgestellt werden und beinhalten nur die notwendigen Informationen, ohne zusätzliche, unnötige Daten preiszugeben. SSI stellt sicher, dass diese Daten sicher, privat und transparent übertragen werden können, sodass sowohl die Authentizität als auch der Datenschutz gewährleistet sind.

In der Fallstudie von E-Systems könnte SSI verwendet werden, um sicherzustellen, dass die von E-Systems verwalteten Produktkonformitätsnachweise nicht nur korrekt, sondern auch nachvollziehbar und fälschungssicher sind. Beispielsweise könnte ein VC über die Erfüllung eines bestimmten Umweltstandards für ein Produkt digital signiert und dem Importeur vorgelegt werden, ohne dass zusätzliche Daten wie Produktionsdetails oder andere interne Unternehmensinformationen offengelegt werden müssen. Dies würde den Prozess der Konformitätsprüfung vereinfachen, Fehlerquellen minimieren und die Effizienz der gesamten Dokumentations- und Prüfprozesse steigern.

Die vorliegende Arbeit verbindet moderne Technologien mit der aktuellen Herausforderung des Umweltschutzes im internationalen Handel. Durch die Anwendung von SSI auf die Produktkonformität im globalen Kontext wird untersucht, wie diese Technologie zu einer effizienteren und nachhaltigeren Lösung beitragen kann.

Der wissenschaftliche Wert der Arbeit liegt darin, aufzuzeigen, wie SSI als neue Technologie in der Praxis funktioniert und welche Vorteile sie im Vergleich zu bestehenden Authentifizierungs- und Identitätssystemen bietet. Dies ermöglicht

es, die Potenziale und Grenzen von SSI zu analysieren und zu bewerten, ob diese Technologie eine zukunftsfähige Lösung für die Probleme im Bereich Umwelt- und Produkt-Authentifizierung darstellen kann.

Zu den erwarteten Ergebnissen dieser Arbeit zählen mehrere wichtige Meilensteine, die sowohl die theoretische als auch die praktische Anwendung von SSI im Kontext der Produkt-Authentifizierung und Umweltkonformität umfassen. Zunächst wird eine Analyse zur aktuellen Implementierbarkeit von SSI durchgeführt. Diese Analyse wird Aufschluss darüber geben, inwieweit SSI bereits in der Praxis eingesetzt werden kann und welche Herausforderungen dabei noch zu bewältigen sind.

Ein weiteres zentrales Ergebnis ist die Erstellung eines lauffähigen Prototyps, der als Demonstration für die Fallstudie dient. Dieser Prototyp soll als greifbarer und wahrnehmbarer Lösungsansatz dazu beitragen, die Potenziale und die Funktionsweise von SSI in einem realen Anwendungsfall aufzuzeigen. Der Prototyp wird nicht nur als praktisches Beispiel dienen, sondern auch als Testumgebung zur Identifikation von möglichen Schwächen und Optimierungspotenzialen in der aktuellen SSI-Technologie. Es wird ein Vergleich angestrebt, der sowohl die Effizienz und Sicherheit als auch die Umweltfreundlichkeit und die Automatisierungspotenziale beleuchtet.

2. Self-Sovereign Identity

SSI ist ein innovatives Konzept zur Verwaltung von Identitäten, das darauf abzielt, die Kontrolle über persönliche Daten vollständig in die Hände der jeweiligen Person zu legen. Im Kontext dieser Arbeit wird Identität als die Gesamtheit von Merkmalen definiert, die eine Person, Organisation oder ein Objekt eindeutig bestimmen und von anderen unterscheiden. Im Gegensatz zu herkömmlichen Identitätssystemen, bei denen zentrale Institutionen oder Organisationen als Mittelsmänner agieren, basiert SSI auf einem dezentralen Ansatz. Dies bedeutet, dass keine zentrale Instanz die Identitätsdaten speichert oder verwaltet, sondern die Nutzer selbst als Verwalter ihrer Identität agieren.

Das Hauptziel von SSI ist es, Nutzern die Möglichkeit zu geben, ihre Identitätsdaten sicher und selbstbestimmt zu verwalten. Es bietet Vorteile sowohl für Einzelpersonen als auch für Organisationen. Für Einzelpersonen bedeutet SSI mehr Kontrolle und Datenschutz, da sie selbst entscheiden können, welche Informationen sie mit wem und in welchem Umfang teilen möchten. Organisationen profitieren wiederum von einem effizienteren Identitätsmanagement und

einer Reduzierung des Risikos, sensible Daten zentral speichern und schützen zu müssen.

Die Funktionsweise von SSI beruht auf der Nutzung moderner Technologien wie der **Blockchain** und kryptographischen Verfahren. Der dezentrale Charakter des Systems wird durch sogenannte DIDs gewährleistet, die es ermöglichen, Identitäten unabhängig von zentralen Behörden oder Plattformen zu verwalten. Ergänzend dazu kommen kryptographische Mechanismen wie digitale Signaturen und Zero-Knowledge-Proofs(ZKPs) zum Einsatz, um Authentizität und Datenschutz zu gewährleisten, ohne dass sensible Daten offengelegt werden müssen. Dieser generelle Ablauf lässt sich somit, wie in Abbildung 1 dargestellt, visualisieren.

SSI ist insbesondere für Personen und Organisationen vorteilhaft, die Wert auf Datenschutz und Sicherheit legen. Ein Beispiel ist die Nutzung von SSI zur digitalen Verifikation von Arbeitszeugnissen oder Reisepässen. Anstatt diese Dokumente bei einer zentralen Behörde zu hinterlegen, können Nutzer kryptographisch gesicherte Nachweise in Form von VCs vorlegen, die nur die notwendige Information offenbaren.

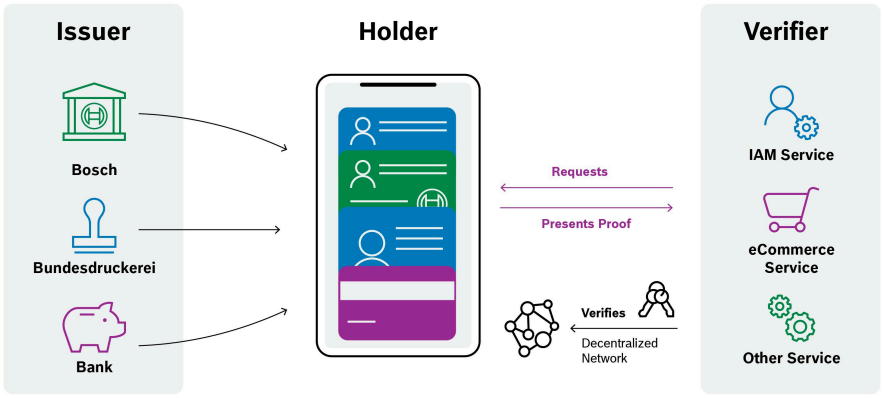


Abbildung 1: Übersichts-Grafik für SSI Ablauf [1]

Im folgenden Abschnitt wird detaillierter auf die einzelnen Komponenten und Akteure eingegangen, die das SSI-Ökosystem ausmachen. Dies umfasst unter anderem die Rolle von DIDs, VCs sowie die Akteure wie Nutzer, Herausgeber (Issuer) und Prüfer (Verifier). Diese Komponenten und Akteure bilden die Grundlage für die Funktionsweise und die Sicherheit von SSI.

2.1. Decentralized Identifiers

Ein DID ist ein zentraler Bestandteil des Self-Sovereign Identity Ökosystems. Es handelt sich dabei um eine weltweit eindeutige, dezentrale und digitale Kennung, die unabhängig von zentralisierten Behörden oder Organisationen erstellt und verwaltet wird. DIDs sind darauf ausgelegt, den Nutzern die Kontrolle über ihre Identität und die damit verbundenen Daten zu geben, ohne dass Dritte die Verwaltung dieser Daten übernehmen müssen.
Funktionsweise und Struktur Ein DID besteht aus mehreren Komponenten, die gemeinsam eine eindeutige Identifikation ermöglichen:

1. **DID-Schema:** Ein vorangestellter Teil, der die Kennung als DID ausweist, zum Beispiel did.
2. **Methodenspezifischer Teil:** Dieser Teil definiert, welche Methode oder Technologie verwendet wird, um den DID zu erstellen und zu verwalten, beispielsweise did:example.
3. **Eindeutiger Identifikator:** Ein individueller Schlüssel oder Wert, der den DID innerhalb der gewählten Methode einzigartig macht

Ein vollständiger DID könnte also wie in Abbildung 2 dargestellt werden.

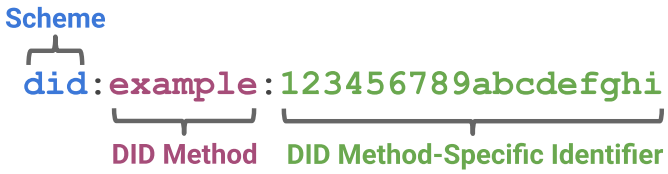


Abbildung 2: Vollständiger Aufbau einer DID [2]

DIDs werden in einem sogenannten **DID-Document** beschrieben. Dieses Dokument enthält Metadaten und kryptographische Informationen wie öffentliche Schlüssel, die für die Authentifizierung und den sicheren Austausch von Daten notwendig sind. Es ist in einem standardisierten Format, meist JSON-LD, abgelegt und wird entweder in einer Blockchain, einem dezentralen Datenspeicher oder direkt auf Anfrage bereitgestellt.

DIDs werden genutzt, um Identitäten sicher und dezentral zu verwalten. Sie dienen als Grundlage für die Verknüpfung mit VCs, welche spezifische Nachweise wie Führerscheine, Zertifikate oder Mitgliedschaften darstellen. Mit einem DID kann ein Nutzer beispielsweise beweisen, dass er bestimmte Berechtigungen besitzt, ohne seine vollständigen persönlichen Daten preiszugeben.

2.2. Verifiable Credentials

VCs sind ein zentraler Bestandteil des SSI-Systems und ermöglichen es, digitale Nachweise über Identitätsattribute sicher und verifiziert auszutauschen. Ein VC stellt dabei eine Art digitale Urkunde dar, die von einer vertrauenswürdigen Instanz ausgestellt wurde und die Authentizität und Integrität der darin enthaltenen Informationen garantiert. Im Gegensatz zu traditionellen Identitätsnachweisen, die auf zentralen Datenbanken beruhen, werden VCs im Rahmen von SSI dezentral gespeichert und verwaltet, sodass die Kontrolle über diese Daten in den Händen des Nutzers bleibt.

Die Funktion von VCs besteht darin, Nutzern zu ermöglichen, ihre Identitätsdaten in einer sicheren und datenschutzfreundlichen Weise zu präsentieren, ohne sensible Informationen unnötig preiszugeben. Beispielsweise kann ein Nutzer ein VC vorlegen, das seine akademische Qualifikation bestätigt, ohne dass dabei Details wie die Studienrichtung oder das genaue Abschlussdatum offengelegt werden. Diese Art der selektiven Offenlegung wird durch SSI ermöglicht, da es dem Nutzer die Kontrolle über die spezifischen Informationen gibt, die er mit anderen teilt.

Die Vorgaben und Standards für die Erstellung und Nutzung von VCs wurden vom World Wide Web Consortium(W3C) festgelegt, einer internationalen Organisation, die Richtlinien für Webtechnologien definiert. Das W3C hat ein spezifisches Standarddokument veröffentlicht, das die Anforderungen und Formate für VCs beschreibt, um sicherzustellen, dass diese nachweislich authentisch und interoperabel sind.

Ein konkretes Beispiel für die Nutzung eines VCs könnte die Verifikation eines Alters sein: Eine Person könnte ein VCs eines ausstellenden Unternehmens oder einer Behörde erhalten, das bestätigt, dass sie über 18 Jahre alt ist. Bei der Nutzung dieses VCs könnte die Person lediglich die Bestätigung ihrer Altersangabe weitergeben, ohne weitere persönliche Daten preiszugeben.

Ein VCs interagiert direkt mit anderen Komponenten des SSI-Systems, wie etwa dem DIDs, der als eindeutiger Identifikator für die Person dient, die das VCs erstellt hat. Zudem spielt das VC eine Rolle im Austausch zwischen dem Issuer, dem Verifier und dem Holder, wobei der **Issuer** das VC ausstellt, der **Holder** es aufbewahrt und der **Verifier** es überprüft, um die Richtigkeit der Angaben zu bestätigen.

2.3. Akteure: Agent, Holder, Issuer, Verifier

In einem SSI-System sind Akteure Personen, Organisationen oder Entitäten, die eine spezifische Rolle in der Verwaltung und Nutzung von Identitäten übernehmen. Jeder Akteur spielt eine wichtige Rolle in der Interaktion mit den verschiedenen Komponenten von SSI, wie DIDs und VCs. Diese Akteure arbeiten zusammen, um die Authentifizierung, Verifikation und Verwaltung von Identitätsinformationen sicher und effizient zu gestalten. Im Folgenden werden die drei wichtigsten Akteure des SSI-Systems vorgestellt: **Holder**, **Issuer** und **Verifier**[3], dessen Interaktionsmöglichkeiten wie in Abbildung 3 dargestellt werden können.

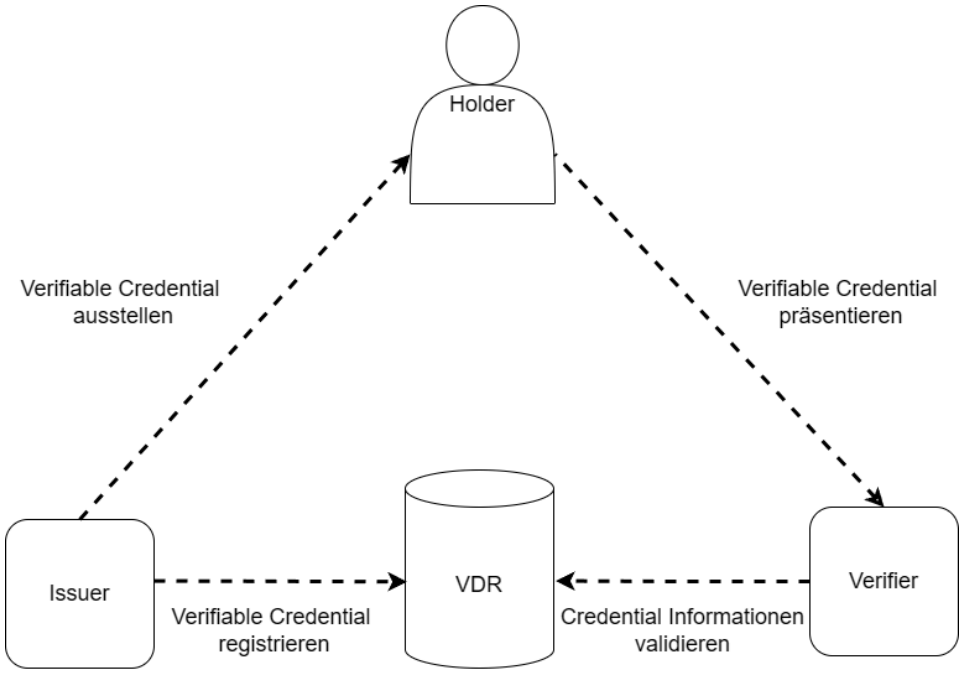


Abbildung 3: Abbildung der Interaktionsmöglichkeiten zwischen SSI Akteuren

Ein **Agent** im Kontext von SSI ist eine Entität, die in der Lage ist, mit dem Identitätssystem zu interagieren, um Identitätsdaten zu empfangen, zu speichern und weiterzugeben. Agenten können sowohl Menschen als auch Software-Anwendungen sein, die in der Rolle eines Holders, Issuers oder Verifiers agieren können.

Sie sind somit flexibel und können verschiedene Funktionen innerhalb des SSI-Ökosystems übernehmen.

Agenten haben direkte Zusammenhänge mit den anderen Akteuren, da sie in der Lage sind, Daten auszutauschen, zu verarbeiten und zu validieren. Wenn ein Agent beispielsweise in der Rolle eines Holders agiert, kann er ein VC von einem Issuer erhalten und es später einem Verifier präsentieren, um eine Identität zu verifizieren. Diese Interaktionen mit anderen Akteuren erfolgen meist mithilfe von DIDs zur eindeutigen Identifikation und zur Sicherstellung der Authentizität der übertragenen Informationen. Agenten spielen somit eine Schlüsselrolle in der Kommunikation und im Datenfluss innerhalb des SSI-Systems.

Der **Issuer** ist eine vertrauenswürdige Entität, die VCs ausstellt. Dies können Organisationen wie Universitäten, Behörden oder Unternehmen sein, die Informationen über eine Person oder Entität bestätigen und als vertrauenswürdig zertifizieren. Der Issuer stellt sicher, dass das VC korrekt und authentisch ist, indem er es mit einer digitalen Signatur versieht.

Issuer interagieren direkt mit anderen Akteuren wie dem Holder, der das VC aufbewahrt und dem Verifier, der es überprüft. Der Issuer kann auch mit den DIDs der entsprechenden Entitäten interagieren, um zu garantieren, dass das ausgestellte VC die richtige Identität widerspiegelt. Issuer sind daher eine zentrale Instanz in der Authentifizierung und Validierung von Identitätsinformationen innerhalb des SSI-Systems.

Der **Verifier** ist eine Entität, die die Gültigkeit von VCs überprüft, um die Authentizität und Integrität der darin enthaltenen Informationen zu bestätigen. Der Verifier könnte eine Organisation oder eine Person sein, die sicherstellen möchte, dass eine Identität oder Information korrekt ist. Diese Überprüfung erfolgt in der Regel durch die Prüfung der digitalen Signatur des Issuers sowie durch die Validierung der DIDs und anderer kryptographischer Mechanismen, die in SSI eingebaut sind.

Der Verifier arbeitet eng mit dem Holder zusammen, der das VC vorlegt, und mit dem Issuer, um zu bestätigen, dass die präsentierten Informationen korrekt sind. Verifier interagieren auch mit anderen Komponenten von SSI, wie etwa den DIDs, um sicherzustellen, dass die Identität des Holders im System eindeutig und überprüfbar ist. Durch diese Interaktionen stellt der Verifier sicher, dass nur verlässliche und authentische Informationen weitergegeben werden.

Der **Holder** ist der Akteur, der VCs empfängt, speichert und bei Bedarf weitergibt. Der Holder ist in der Regel die Person oder Entität, deren Identität oder spezifische Attribute bestätigt werden. Sie haben die Kontrolle über ihre VCs und entscheiden selbst, wann und mit wem sie diese teilen möchten. Der Holder kann VCs von einem Issuer erhalten und sie später einem Verifier vorlegen, um ihre Identität zu bestätigen.

Der Holder interagiert sowohl mit dem Issuer als auch mit dem Verifier. Die Rolle des Holders im SSI-System ist entscheidend, da sie die Kontrolle über ihre eigenen Identitätsdaten behalten und sicherstellen können, dass diese nur in dem Umfang und zu dem Zeitpunkt weitergegeben werden, den sie selbst bestimmen. Der Holder nutzt DIDs zur Identifikation und um sicherzustellen, dass die geteilten Informationen authentisch und vertrauenswürdig sind.

2.4. Zero-Knowledge-Proof

Ein ZKP[4] ist ein kryptographisches Verfahren, mit dem eine Partei (der Holder) einer anderen Partei (dem Verifier) die Wahrheit einer Aussage beweisen kann, ohne dabei irgendwelche zusätzlichen Informationen über die Aussage selbst preiszugeben. Im Wesentlichen ermöglicht ein ZKP dem Holder, die Gültigkeit einer Behauptung zu bestätigen, ohne Details zu offenbaren, die über das notwendige Minimum hinausgehen. Ein klassisches Beispiel für ein ZKP ist der Beweis, dass jemand eine bestimmte Information kennt, ohne diese Information tatsächlich preiszugeben.

ZKP sind besonders nützlich in Bereichen, in denen der Schutz von sensiblen Daten und die Wahrung der Privatsphäre von zentraler Bedeutung sind. Ein praktisches Beispiel im Kontext von SSI wäre der Nachweis, dass eine Person volljährig ist, ohne das genaue Geburtsdatum oder andere private Daten preiszugeben. Durch die Verwendung von ZKP kann die Identität validiert werden, ohne dass zusätzliche persönliche Informationen über die betroffene Person geteilt werden müssen.

Ein weiterer Vorteil von ZKP ist, dass sie die Sicherheit und Privatsphäre in einem dezentralen Identitätssystem wie SSI erhöhen. Sie ermöglichen es Nutzern, sensible Daten wie Geburtsdatum, Adresse oder Finanzinformationen zu schützen, während sie dennoch in der Lage sind, bestimmte Eigenschaften oder Qualifikationen zu beweisen. Ein ZKP stellt sicher, dass keine zusätzlichen Daten offengelegt werden, was das Risiko von Datenmissbrauch oder -diebstahl reduziert.

ZKP können in verschiedenen Szenarien eingesetzt werden, beispielsweise bei der Authentifizierung von Nutzern, dem Nachweis von Berechtigungen oder der Verifikation von Transaktionen. Sie haben das Potenzial, viele traditionelle Sicherheitssysteme zu ersetzen, indem sie den Fokus auf die Wahrung der Privatsphäre und die Minimierung von Datenlecks legen. ZKP sind somit ein wichtiger Bestandteil in der Weiterentwicklung sicherer und datenschutzfreundlicher Technologien.

2.5. Selective-Disclosure

Selective Disclosure[5] ist ein Mechanismus im Rahmen von SSI, der es einem **Holder** ermöglicht, nur bestimmte Attribute eines VC offenzulegen, anstatt das gesamte Dokument weiterzugeben. Dies erhöht die Privatsphäre und Datenminimierung, indem beispielsweise ein Altersnachweis erbracht werden kann, ohne das genaue Geburtsdatum preiszugeben. Ein **Verifier** kann die Authentizität der offengelegten Informationen überprüfen, ohne Zugriff auf die verborgenen Attribute zu erhalten. Dies wird häufig durch kryptografische Verfahren wie ZKPs oder selektiv signierte JSON Web Signatures(JWSs) realisiert.

3. Technologien

Um die Umsetzung der zuvor genannten Konzepte zu ermöglichen, müssen verfügbare Technologien betrachtet und schließlich diejenigen ausgewählt werden, die als Grundlage für die Implementierung dienen.

3.1. Optionen

Nach der initialen Recherche ergaben sich mehrere Ergebnisse für die Auswahl möglicher Technologien. Im Folgenden werden die drei Optionen vorgestellt, zwischen denen letztlich eine Entscheidung getroffen wurde.

3.1.1. Hyperledger Aries

Als eine der ersten funktionsfähigen Optionen stand Hyperledger Aries[6] da, Hyperledger Aries ist eine Open-Source-Infrastruktur, die auf die Entwicklung dezentraler Identitätssysteme ausgerichtet ist. Sie nutzt die Programmiersprachen Go und Python und stellt Tools zur Verfügung, die die Verwaltung von DIDs und VCs ermöglichen. Hyperledger Aries basiert auf Agenten, die über Decentralized Identifier Communication(DIDComm)-Protokolle miteinander kommunizieren.

Die Infrastruktur unterstützt verschiedene Ledger-Technologien, die die Speicherung und Verwaltung von Identitätsdaten auf einer Blockchain ermöglichen. Hyperledger Aries stellt auch eine Reihe von Bibliotheken und Frameworks bereit, die eine Implementierung und Interoperabilität zwischen verschiedenen SSI-Systemen ermöglichen.

3.1.2. DIDKit

Als weitere Option präsentete sich DIDKit[7], DIDKit ist eine Bibliothek, die auf Rust basiert und für die Arbeit mit DIDs und VCs entwickelt wurde. Sie bietet Funktionen zur Erstellung und Verifizierung von DIDs und VCs, wobei sie Standards wie W3C unterstützt. DIDKit nutzt gängige kryptografische Mechanismen wie JWS und JSON Web Token(JWT) und lässt sich in verschiedene Systeme integrieren. Die Bibliothek bietet die Möglichkeit, dezentrale Identitätssysteme zu implementieren, ohne dass eine tiefe Auseinandersetzung mit der zugrunde liegenden Technologie erforderlich ist.

3.1.3. Veramo

Zuletzt bot sich auch Veramo[8] als funktionsfähige Option an, Veramo ist ein Framework, das in TypeScript entwickelt wurde und eine Architektur für die Verwaltung von DIDs und VCs bereitstellt. Es bietet eine Sammlung von Plugins, die verschiedene Funktionen zur Verwaltung von Identitätsdaten integrieren. Veramo unterstützt unterschiedliche Verifiable Data Registrys(VDRs) und ermöglicht die Anbindung an verschiedene Blockchain-Technologien, desweiteren ermöglicht die API von Veramo die zentrale Verwaltung der Identitätsinfrastruktur.

3.2. Auswahl

Die Auswahl der Technologien für den Prototypen gliederte sich in zwei Bereiche: SSI-relevante und SSI-unabhängige Technologieentscheidungen. Im Folgenden wird erläutert, welche Technologien letztendlich ausgewählt wurden und aus welchen Gründen diese Entscheidung getroffen wurde.

3.2.1. SSI-Relevant

Auf Grundlage der in Abschnitt 3.1 analysierten Technologien fiel die Wahl auf das **Veramo**-Framework. Ausschlaggebend dafür war der Mangel an aktueller und funktionsfähiger Dokumentation bei den alternativen Optionen sowie Einschränkungen durch die Programmiersprachen wie Go und Python. Im Vergleich zu DIDKit spielte zudem die fehlende Unterstützung zusätzlicher Funktionen, wie die

Integration von Blockchains als VDR, eine entscheidende Rolle.

Letztendlich erwies sich Veramo als die vielseitigste Option, da es umfangreiche Funktionen bietet. Dazu gehören die automatische Nutzung von DIDComm für gesicherte Kommunikation zwischen DIDs, die integrierte Verwaltung kryptografischer Schlüssel, die Speicherung von VCs und DIDs sowie die Unterstützung von Ethereum als zugrunde liegende Blockchain.

Abgesehen von aktiven Entscheidungen fiel auch passiv die Wahl auf die Nutzung der W3C-Standards. Das W3C ist eine internationale Organisation, die offene Webstandards entwickelt. Im Kontext von SSI definiert das W3C zentrale Spezifikationen für DIDs und VCs. Die **DID Core Specification**[2] legt fest, wie DIDs strukturiert sind und verwaltet werden können, während die **Verifiable Credentials Data Model Specification**[9] ein standardisiertes Modell für die Erstellung, Ausstellung und Verifizierung von VCs bereitstellt. Diese Standards gewährleisten Interoperabilität und ermöglichen dezentrale Identitätslösungen, die unabhängig von zentralen Autoritäten funktionieren.

3.2.2. SSI-Unabhängig

Der Vollständigkeit halber werden nun die SSI-unabhängigen Technologien aufgeführt, die für die Implementierung ausgewählt wurden. Dazu gehören **React** für die Frontend-Entwicklung, **TypeScript** zur Kompatibilität mit den Veramo-Bibliotheken, **Shadcn/UI** und **Tailwind CSS v4** zur Gestaltung und Verwaltung der Benutzeroberfläche, **Ganache** zur Bereitstellung eines lokalen Ethereum-Netzwerks als VDR für Veramo sowie **Truffle** zur Verwaltung von Smart Contracts[10] auf diesem Netzwerk.

4. Konzept

Um die Anforderungen und Funktionen des Prototyps zu erfassen, müssen spezifische Szenarien und Interaktionen zwischen den Akteuren des Systems festgelegt werden. Diese Anforderungen bilden die Grundlage für eine erfolgreiche Entwicklung des Prototyps. Im folgenden Kapitel wird zunächst ein konkretes Szenario erstellt, das dazu dient, die Umstände und Interaktionsmöglichkeiten zu verstehen. Anschließend wird durch die Definition relevanter Use Cases detailliert auf die im Szenario beschriebenen Interaktionen eingegangen, um ein umfassendes Verständnis für die Konzeption des Prototyps zu gewährleisten.

Unter dem Begriff „Konzept“ versteht man dabei die grundlegende Idee und Struktur, die als Leitfaden für die Entwicklung eines Software-Prototypen dient. Im konkreten Fall bezieht sich der globale technische Rahmen auf die Implementierung einer Webapplikation mit integrierter SSI.

Zentral für das Konzept sind die drei definierten Akteure – Holder, Issuer und Verifier – die jeweils spezifische Rollen im System übernehmen:

1. **Der Issuer** repräsentiert vertrauenswürdige Parteien, wie etwa verschiedene Regierungen, die für die Ausstellung von Zertifikaten verantwortlich sind.
2. **Der Holder** steht für alle Nutzer, die im Besitz eines Zertifikats sind, beispielsweise Kunden von E-Systems oder Firmeninhaber und Mitarbeiter.
3. **Der Verifier** wiederum bezeichnet jene, die die Zertifikate verifizieren, wozu neben E-Systems auch weitere staatliche Institutionen zählen können.

Diese klare Rollenunterscheidung ist essenziell, um die Verantwortlichkeiten und Interaktionen innerhalb des Systems eindeutig zu definieren. Neben den konditionalen Anforderungen, die sich aus den spezifischen Anwendungsfällen ergeben, existieren darüber hinaus auch technische Anforderungen, auf die später im Dokument ausführlich eingegangen wird.

4.1. Szenario

Das Szenario bietet einen kompakten Überblick über die wesentlichen Abläufe, ohne sich in zu vielen Details zu verlieren. Dabei werden die beteiligten Akteure, Komponenten und Interaktionen in einer spezifischen Situation dargestellt.

Das gewählte Szenario fokussiert sich auf die Einhaltung der Produktkonformität, ein Bereich, der sich besonders für den Einsatz von Zertifikaten eignet. Aufgrund der hohen Komplexität vieler Produkte besteht ein natürlicher Bedarf an Konformitätsnachweisen, da deren Herstellung zahlreiche Prozesse und Inhaltsstoffe umfasst. Diese müssen dokumentiert werden, um die Einhaltung der jeweiligen Kriterien zu gewährleisten.

Diese Anforderungen variieren je nach Zielland, da jedes Land spezifische Einreisebestimmungen für Produkte besitzt. Daher sind zahlreiche Dokumente und Zertifikate erforderlich, die in unterschiedlichen Formaten vorliegen können – von physischen Dokumenten bis hin zu digitalen Zertifikaten. Die derzeitigen Prozesse zur Überprüfung dieser Nachweise sind jedoch zeitaufwendig, manuell und fehleranfällig, da sie die aktive Beteiligung verschiedener Parteien erfordern. Durch die Implementierung von SSI und Smart Contracts kann dieser Prozess

effizienter, sicherer und schneller gestaltet werden.

SSI ermöglicht die digitale Erfassung und Bündelung aller relevanten Produktinformationen in einem VC, das die Produktkonformität bestätigt. Die Nutzung eines Smart Contracts gewährleistet zudem eine automatisierte und transparente Interaktion zwischen den beteiligten Parteien, während die Anbindung an eine Blockchain die sichere Verwaltung von Zahlungen und Dokumentenübertragungen unterstützt. Das in Abbildung 4 dargestellte Diagramm dient der Visualisierung des darauf folgenden Szenarios und soll parallel dazu betrachtet werden.

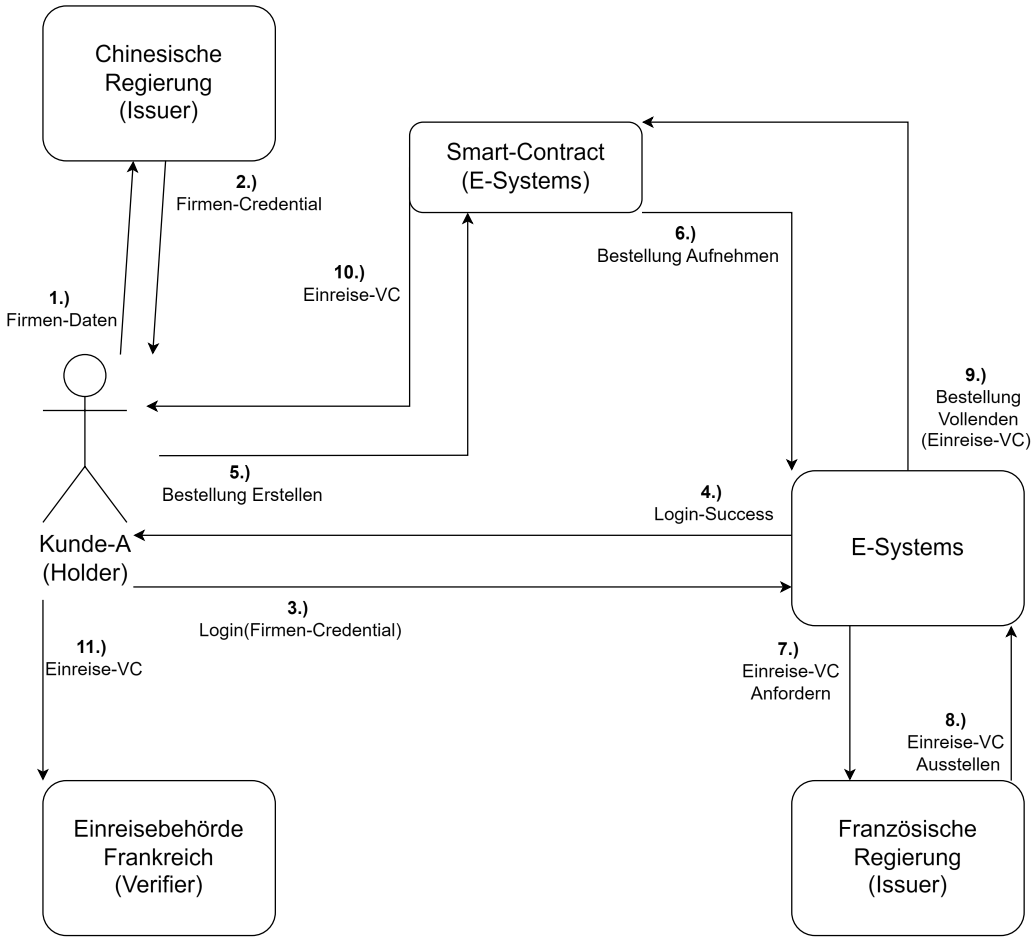


Abbildung 4: Nummeriertes Interaktions-Diagramm des Szenarios

0. Kondition: `Kunde-A` möchte ein VC für sein Produkt erhalten, um die Einreise nach `Land-B` zu ermöglichen. Dazu kontaktiert er `E-Systems`, das für die Ausstellung des Zertifikats zuständig ist.

1. Um sich als berechtigter Antragsteller zu legitimieren, benötigt `Kunde-A` zunächst ein VC von `Land-A`, das bestätigt, dass er im Namen einer registrierten Firma handelt. Hierfür fordert `Kunde-A` ein VC von `Land-A` an welches dies bestätigt.

2. Nach erfolgreicher Prüfung stellt `Land-A` das entsprechende **Credential** aus.

3. `Kunde-A` nutzt dieses **Credential**, um sich bei `E-Systems` zu verifizieren.

4. `E-Systems` bestätigt daraufhin die Validität des **Credentials**.

5. Nach erfolgreicher Verifizierung durch `E-Systems` gibt `Kunde-A` eine Bestellung über einen **Smart Contract** auf. Die Bestellung enthält die erforderlichen Produktdaten sowie die Zahlung für die Zertifikatsausstellung.

6. `E-Systems` nimmt die Bestellung entgegen und prüft die bereitgestellten Informationen.

7. Nach erfolgreicher Prüfung fordert `E-Systems` nun ein **Einreise-VC** bei `Land-B` an.

8. Nach Validierung der Daten stellt `Land-B` das erforderliche Zertifikat aus und übermittelt es an `E-Systems`.

9. `E-Systems` vollendet die Bestellung durch die Einreichung des **Einreise-VC** an den **Smart Contract**.

10. Der **Smart Contract** veranlasst die Freigabe der Zahlung und leitet das **Einreise-VC** an `Kunde-A` weiter.

11. `Kunde-A` präsentiert das erhaltene Zertifikat der **Einreisebehörde** von `Land-B`, die das **Credential** verifiziert und daraufhin die Einreise genehmigt.

Durch die Definition dieses Szenarios wurde eine grobe Grundlage geschaffen, um die in Abschnitt 4.2 beschriebenen Interaktionen detaillierter in relevanten Use Cases darzustellen.

4.2. Use-Cases

Um einen Überblick über die verschiedenen Anwendungsfälle zu erhalten, werden im Folgenden die zentralen Use-Cases erläutert. Dabei stehen unter anderem die Erstellung einer DID, die Anfrage sowie die Verifizierung eines Verifiable Credentials im Mittelpunkt. Darüber hinaus beinhalten diese auch die Erstellung einer Bestellung für E-Systems sowie die Annahme und Abhandlung offener Bestellungen durch E-Systems. Ziel dieser Ausführungen ist es, die zuvor definierten Akteure in eine klarere Beziehung zueinander zu setzen, die in Abschnitt 4.1 definierten Interaktionen zu detailieren und aufzuzeigen, wie die Endnutzer des Systems mit den jeweiligen Funktionen interagieren. Die im Folgenden beschriebenen Use-Cases werden durch Abbildung 5 weiter veranschaulicht.

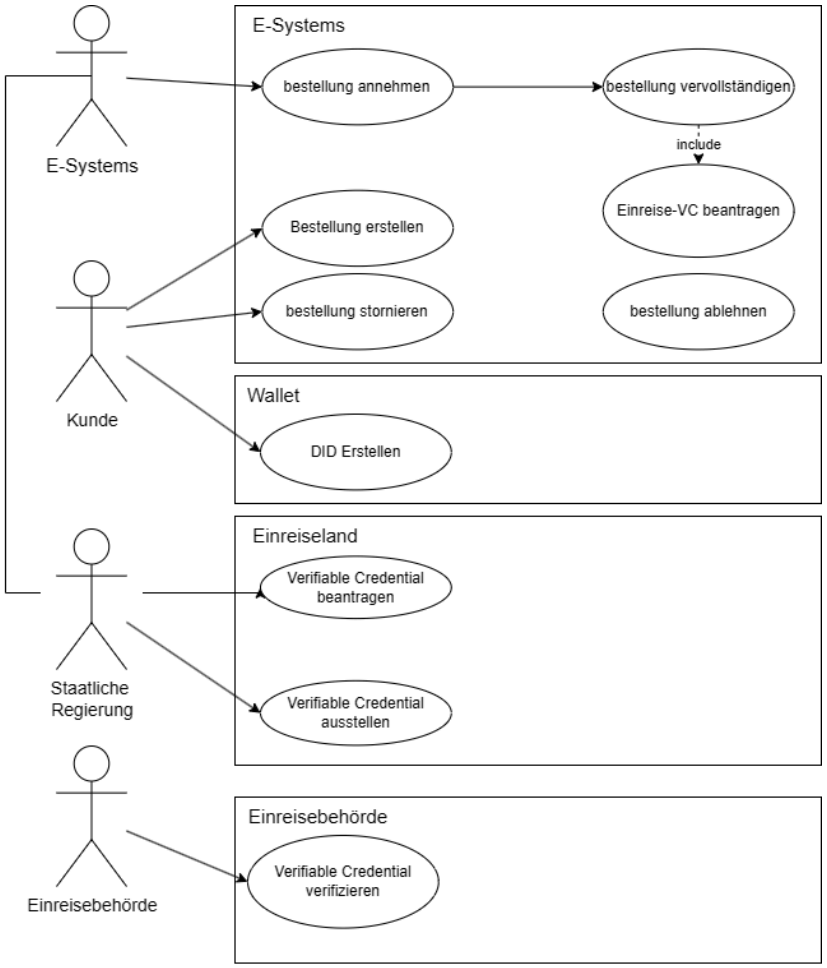


Abbildung 5: Vollständiges Use-Case Diagramm zum Prototypen

Ein zentraler Schritt bei der Nutzung eines SSI-Systems ist die **Erstellung einer DID**, die als eindeutige Kennung für den Nutzer dient. Um sich im System zu

registrieren, gibt der Kunde zunächst einen Alias an, der als benutzerfreundliche Bezeichnung für seine Identität dient. Anschließend wird ihm durch das System eine eindeutige DID zugewiesen. Diese DID ermöglicht es dem Kunden, sich gegenüber anderen Teilnehmern des Netzwerks zu authentifizieren und digitale Nachweise zu empfangen oder weiterzugeben. Die DID kann anschließend in einer digitalen Wallet gespeichert und zur Identifikation in verschiedenen Anwendungen des SSI-Ökosystems genutzt werden.

Ein weiterer Anwendungsfall ist die **Ausstellung eines VCs**. In diesem Prozess fordert ein Kunde von einer Regierungsbehörde ein VC an, das als digitaler Nachweis für eine bestimmte Eigenschaft oder Berechtigung dient. Zunächst authentifiziert sich der Kunde gegenüber der Behörde unter Verwendung seiner DID. Anschließend stellt er eine offizielle Anfrage für das gewünschte VC und übermittelt dabei alle erforderlichen Nachweise oder Dokumente. Die Behörde überprüft die Angaben des Kunden und entscheidet, ob die Ausstellung des VC gerechtfertigt ist.

Nach erfolgreicher Prüfung **erstellt** die Behörde das VC und signiert es kryptografisch, um dessen Echtheit sicherzustellen. Das ausgestellte VC wird daraufhin an den Kunden übertragen. Dieser kann das VC nun nutzen, um sich gegenüber Dritten – beispielsweise Unternehmen oder anderen Behörden – zu verifizieren, ohne seine sensiblen Originaldokumente weitergeben zu müssen.

Die **Verifizierung eines VCs** beginnt, wenn ein Kunde sein VC einem Verifizierer vorlegt. Das VC enthält kryptografisch signierte Informationen über eine bestimmte Eigenschaft oder Berechtigung des Kunden. Der Verifizierer überprüft zunächst die Signatur des Issuers, um sicherzustellen, dass das VC von einer vertrauenswürdigen Quelle stammt. Anschließend wird geprüft, ob das VC nicht widerrufen wurde und ob es noch gültig ist.

Die **Erstellung einer Bestellung** bei E-Systems erfolgt über einen Smart Contract, der speziell für die Abwicklung von Bestellungen entwickelt wurde. Dieser Smart Contract stellt verschiedene Funktionen bereit, um Bestellungen dezentral zu verwalten und automatisierte Prozesse zu ermöglichen. Der Bestellvorgang beginnt damit, dass ein Kunde eine neue Bestellung über den Smart Contract aufgibt. Hierfür übermittelt der Kunde die erforderlichen Produktdaten sowie die entsprechende Zahlung in Ether. Der Smart Contract verarbeitet die Eingaben und erstellt daraufhin eine neue offene Bestellung, die auf der Blockchain gespeichert wird.

Nachdem E-Systems eine Bestellung erhalten hat, beginnt die **Bearbeitung des Bestellprozesses**. Zunächst interagiert E-Systems mit dem Smart Contract, um die Bestellung mit einer spezifischen Bestellungs-ID anzunehmen. Dieser Schritt stellt sicher, dass die Bestellung korrekt im System erfasst und für die weitere Bearbeitung vorbereitet wird. Im Anschluss beauftragt E-Systems die Anforderung eines VCs beim Importland, das für die Einfuhr des Produkts erforderlich ist. Sobald das VC bei E-Systems eingegangen ist, wird der Bestellprozess über den Smart Contract vervollständigt. Das VC wird dabei direkt an die Bestellung im Smart Contract übermittelt.

Nachdem das VC in das System integriert wurde, erfolgt die abschließende Zahlungstransaktion. E-Systems erhält die Zahlung des Kunden in Form von Ether, und der Kunde erhält das benötigte Einreise-VC, das ihm die Einfuhr des Produkts ins Zielland ermöglicht. Dieser automatisierte Prozess stellt sicher, dass sowohl die Zahlung als auch die erforderlichen Dokumente effizient und fälschungssicher abgewickelt werden.

5. Prototyp

Die prototypische Ausarbeitung bildet den Kern dieser Arbeit. Das folgende Kapitel deckt alle Implementierungsschritte ab, basierend auf dem Szenario und den dazugehörigen User Stories. Dabei kommen die in Abschnitt 3 genannten Technologien zum Einsatz, um die Umsetzung konkret vorzunehmen. Abschließend werden auch die Hindernisse und Probleme thematisiert, die während der Ausarbeitung aufgetreten sind.

5.1. Software-Architektur

Um die Entwicklung und Implementierung des Prototyps zu ermöglichen, wurden auf Basis des zuvor definierten Konzepts und Szenarios eine Software-Architektur entwickelt. Der folgende Abschnitt geht detailliert auf diese Architektur ein. Dabei werden die einzelnen Komponenten des Systems spezifiziert, einschließlich ihrer jeweiligen Rolle im Prototyp, der Art der verarbeiteten Daten sowie ihrer Position innerhalb der Architekturebenen. Zudem werden die Hauptfunktionalitäten der Komponenten beschrieben und ihre Interaktionen mit anderen Teilen des Systems analysiert. Diese strukturierte Definition stellt sicher, dass alle Komponenten klar abgegrenzt sind und ein funktionales SSI-System ergeben.

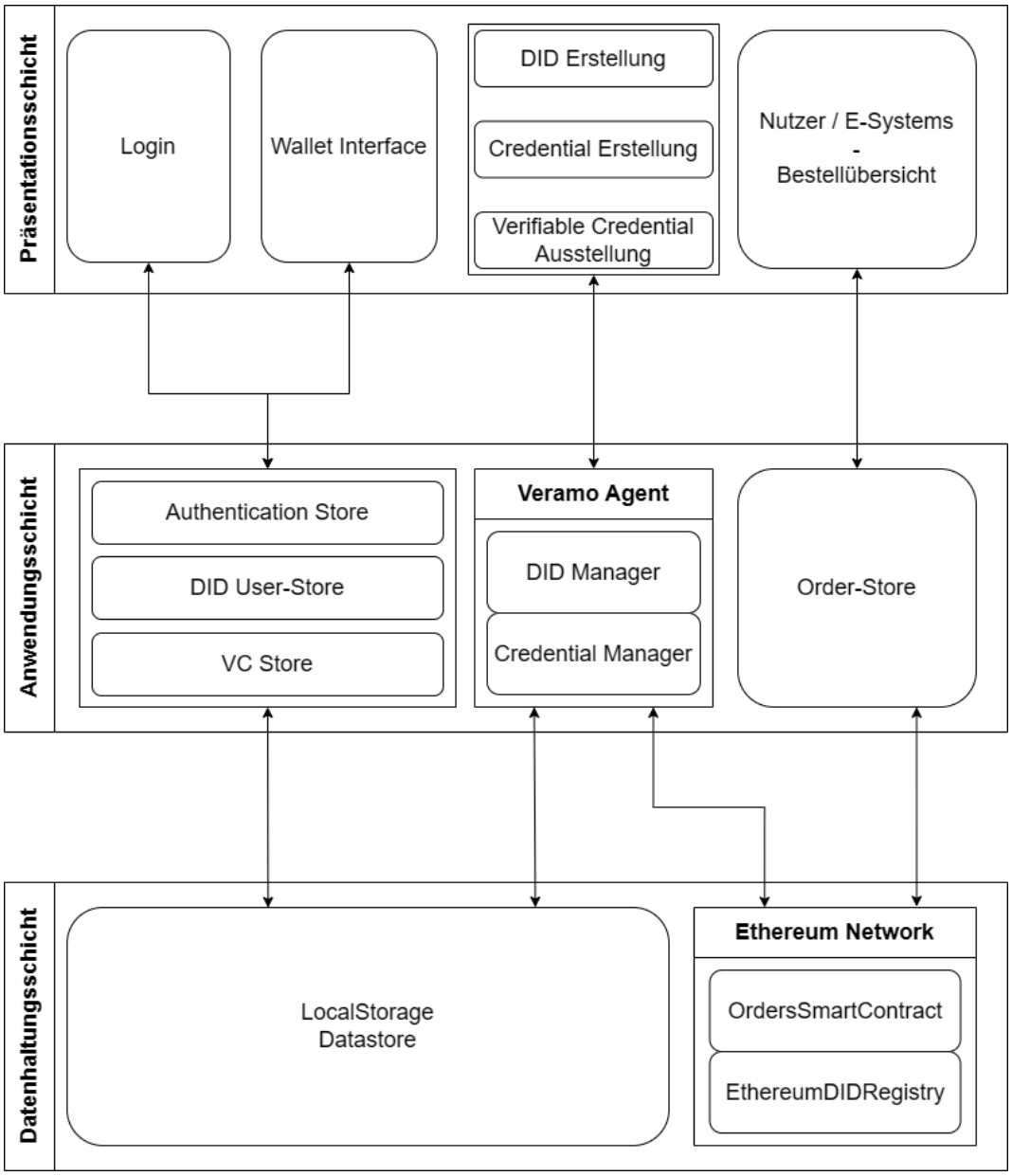


Abbildung 6: Software-Architektur Diagramm des Prototypen

Im Gesamtkontext des Projekts bildet die mehrschichtige Architektur die Grundlage für die Umsetzung eines SSI-Systems, das auf die Verwaltung und Verifizierung von VCs ausgerichtet ist. Das System besteht, wie in Abbildung 6 zu sehen, aus drei Hauptschichten:

- **Präsentationsschicht:** Diese Schicht ist für die Benutzeroberfläche verantwortlich und ermöglicht die Interaktion der Akteure mit dem SSI-System. Sie stellt sicher, dass Benutzer visuelle Schnittstellen haben, um ihre Identitäten zu verwalten, DIDs zu erstellen und VCs anzufordern, einzusehen und zu verifizieren.

- **Anwendungsschicht:** In dieser Schicht werden die Kernprozesse des SSI-Systems abgewickelt. Dazu gehört die Erstellung, Verwaltung und Überprüfung von VCs sowie die Authentifizierung und Autorisierung der Benutzer. Diese Schicht enthält die Logik, um mit den Blockchain-basierten Smart Contracts zu kommunizieren, die für die Validierung der Identität und die Integrität der VCs zuständig sind.

- **Datenhaltungsschicht:** Die Datenhaltungsschicht speichert alle für das SSI-System relevanten Daten. Diese Schicht ist in zwei Teile untergliedert:

  - **Ethereum-Blockchain:** Hier werden Daten mithilfe von Smart Contracts gespeichert und verwaltet. Die Blockchain sorgt für die dezentralisierte und manipulationssichere Speicherung der Identitätsinformationen und VCs.

  - **Lokale Datenspeicherung:** Diese Komponente dient der Verwaltung von Daten, die nicht direkt auf der Blockchain gespeichert werden müssen, wie beispielsweise der Authentifizierungsstatus der Benutzer oder die Inhalte der Wallet-Simulation.

5.1.1. Präsentationsschicht

Zu den **Hauptkomponenten der Präsentationsschicht** gehören die Bestellübersicht, die Wallet-Oberfläche sowie drei kleinere, aber ebenso wichtige Komponenten: DID Erstellung, Credential Erstellung und VC Ausstellung. Jede dieser Komponenten hat eine eigene Aufgabe, trägt jedoch zur gesamten Funktionalität des Systems bei.

Die **Bestellübersicht** ist eine der zentralen Komponenten der Präsentationsschicht. Es dient als Übersichtsseite, auf der Benutzer ihre Bestellungen einsehen, verwalten und nachverfolgen können. Hier werden die Interaktionen mit den Bestelldaten angezeigt, wobei die Benutzer in der Lage sind, auf verschiedene Details zuzugreifen und neue Bestellungen zu initiieren. Das Dashboard ist dabei eng mit der Anwendungsschicht verbunden, da es die Benutzeranfragen an den internen OrderStore weiterleitet, der für die logische Verarbeitung von Bestellungen

zuständig ist. Die Kommunikation zwischen der Bestellübersicht und der Anwendungsschicht ermöglicht es, Bestellungen in das System zu integrieren und ihren Status zu aktualisieren, sodass eine reibungslose und transparente Verwaltung der Bestellprozesse gewährleistet ist.

Das **Wallet-Interface** ist eine weitere Schlüsselkomponente der Präsentationsschicht. Es dient als die visuelle Darstellung der Identität des Benutzers innerhalb der Anwendung. Über dieses Interface können Benutzer ihre VCs einsehen und verwalten sowie ihre Identitätsdaten aktualisieren. Das Wallet-Interface stellt sicher, dass Benutzer jederzeit Zugriff auf ihre gespeicherten Identitätsinformationen haben und diese bei Bedarf vorzeigen oder verwenden können. Diese Komponente ist von zentraler Bedeutung, da sie die Benutzererfahrung mit den grundlegenden SSI-Funktionen, wie der Verwaltung von Identitäten und Credentials, verknüpft.

Die kleineren Komponenten, die ebenfalls zur Präsentationsschicht gehören, sind auf die spezifische Verwaltung der Identitätsdaten des Benutzers ausgerichtet. Die **DID-Erstellung** ermöglicht es den Benutzern, einen dezentralen Identifikator zu generieren, der als Basis für ihre digitale Identität dient. Die **Credential-Erstellung** wiederum bietet die Möglichkeit, neue VCs zu generieren, die von vertrauenswürdigen Dritten ausgestellt werden.

Schließlich sorgt die **Verifiable-Credential-Ausstellung** dafür, dass diese Credentials nach den Regeln des SSI-Systems gültig gemacht und an die Benutzer ausgegeben werden. Diese kleineren Komponenten ermöglichen eine grundlegende Interaktion mit der Identitätsverwaltung des Systems und stellen sicher, dass Benutzer ihre digitalen Identitäten auf einfache Weise erstellen und pflegen können.

Insgesamt bildet die **Präsentationsschicht** das Interface, das den Benutzern die Nutzung des SSI-Systems ermöglicht. Sie bietet die notwendigen Tools, um mit Identitäten, Credentials und Bestellungen zu interagieren, und stellt sicher, dass die komplexen Systemprozesse hinter den Kulissen für die Benutzer auf einfache und intuitive Weise zugänglich sind.

5.1.2. Anwendungsschicht

Die **Anwendungsschicht** bildet den Kern des SSI-Systems, in dem die Geschäftslogik und die wesentlichen Prozesse zur Verwaltung von Identitäten und VCs implementiert sind. Sie dient der Verarbeitung von Daten, die von der Präsentationsschicht übermittelt werden, und stellt sicher, dass diese in der Datenhaltungsschicht gespeichert und abgerufen werden können. In der vorliegenden Architektur umfasst die Anwendungsschicht vier wesentliche Komponenten: den Authentication Store, den DID User Store, den Veramo `agent` sowie den `Order Store`.

Der **Veramo `agent`** stellt die zentrale Komponente der Anwendungsschicht dar. Er ist verantwortlich für die Verwaltung von DIDs und VCs und stellt sicher, dass die Anforderungen der Präsentationsschicht zur Verwaltung von Identitäten korrekt umgesetzt werden. Der Veramo `agent` agiert dabei als Schnittstelle zwischen der Anwendungsschicht und der Datenhaltungsschicht, insbesondere durch die Nutzung des DID Managers und des Credential Managers.

Der **DID Manager** kümmert sich um die Erstellung und Verwaltung von DIDs, während der **Credential Manager** die Erstellung, Verwaltung und Validierung von VCs übernimmt. Diese beiden Komponenten arbeiten zusammen, um den gesamten Lebenszyklus von DIDs und VCs im SSI-System zu steuern. Durch den Einsatz des Veramo `agent` wird eine Integration von SSI-Funktionalitäten gewährleistet, wobei der Agent auch die Interaktion mit der Blockchain über Smart Contracts ermöglicht.

Der **`Order Store`** ist eine weitere zentrale Komponente der Anwendungsschicht, die für die interne Verwaltung von Bestellungen verantwortlich ist. Hier werden alle Bestellvorgänge verarbeitet und die relevanten Daten gespeichert. Der `Order Store` sorgt für die logische Verwaltung und Verarbeitung von Bestellungen und sorgt dafür, dass die entsprechenden Bestellinformationen zwischen der Anwendungsschicht und der Präsentationsschicht synchronisiert werden. Diese Komponente ermöglicht es, dass Benutzer über die Bestellübersicht interaktive Funktionen zur Verwaltung ihrer Bestellungen nutzen können.

Der DID **User Store** dient der Identitätsrepräsentation innerhalb des Prototyps. Er speichert die DIDs und andere relevante Identitätsinformationen der Benutzer und stellt sicher, dass diese korrekt im System abgebildet werden.

Die Interaktion der **Anwendungsschicht** mit der Datenhaltungsschicht konzen-

triert sich vor allem auf das Speichern und Abrufen von Daten sowie die Durchführung von CRUD-Operationen (Create, Read, Update, Delete).

5.1.3. Datenhaltungsschicht

Die **Datenhaltungsschicht** stellt die Grundlage für die Speicherung und Verwaltung aller relevanten Daten im SSI-System dar. Sie besteht aus zwei zentralen Komponenten: dem LocalStorage (DataStore) und dem unterliegenden Ethereum Netzwerk. Diese beiden Komponenten übernehmen unterschiedliche Aufgaben welche die notwendige Persistenz für Daten sicherstellen.

Der **LocalStorage (DataStore)** ist für die Speicherung und Verwaltung lokaler Daten verantwortlich. Diese Komponente stellt sicher, dass alle nicht-blockchain-basierten Daten innerhalb des Systems persistent abgelegt werden. Ein Beispiel hierfür ist die Speicherung von Zustandsinformationen für die Benutzeroberfläche. Diese werden benötigt, um die Login-Komponente zu unterstützen und den Authentifizierungsprozess der Benutzer zu ermöglichen.

Der LocalStorage enthält auch wichtige Metadaten, die für die Simulation einer Wallet erforderlich sind, insbesondere zur Verwaltung von Benutzerinformationen, die in der Präsentationsschicht, zum Beispiel über das Wallet-Interface, angezeigt werden.

Das **Ethereum Netzwerk** spielt eine entscheidende Rolle in der Datenhaltungsschicht, insbesondere zur Umsetzung der Blockchain-basierten VDR. Es handelt sich dabei um ein lokal simuliertes Netzwerk, das als Grundlage für die dezentrale Identitätsverwaltung dient. Der Einsatz des Ethereum Netzwerks ermöglicht es, die Vorteile einer Blockchain wie Sicherheit, Unveränderbarkeit und Dezentralisierung zu nutzen. Das Ethereum Netzwerk übernimmt typische Aufgaben eines VDR, wie etwa:

- **Verifizierung der Identität:** Durch die Bereitstellung einer verlässlichen, dezentralen Quelle für DIDs und VCs.
- **Datenintegrität:** Gewährleistung, dass gespeicherte Daten nicht nachträglich verändert werden können.
- **Transparenz:** Alle Transaktionen, die auf der Blockchain durchgeführt werden, sind öffentlich einsehbar und nachvollziehbar, was zu einer erhöhten Vertrauenswürdigkeit führt.

Für die Verwaltung der DIDs und der VCs innerhalb des Ethereum Netzwerks ist der **EthereumDIDRegistry** SmartContract verantwortlich. Dieser Smart Contract stellt sicher, dass DIDs in der Blockchain registriert und verwaltet werden können. Er ermöglicht die Registrierung, das Update und die Abfrage von DIDs und stellt somit die Grundlage für die dezentrale Identitätsverifizierung dar.

Der **OrdersSmartContract** ist für die Verwaltung von Bestellungen und Zahlungen zuständig ist. Dieser Smart Contract speichert wesentliche Bestelldaten in Form von „Orders“, die direkt auf der Blockchain abgelegt werden. Zu den gespeicherten Daten gehören unter anderem die Zahlung in Ether, die vom Kunden benötigten Daten und VCs.

Der Datenfluss innerhalb der Architektur folgt einem geordneten Schichtmodell. Das bedeutet, dass Komponenten nicht direkt mit weiter entfernten Schichten interagieren, sondern ausschließlich mit der unmittelbar darunterliegenden Schicht kommunizieren.

Die Kommunikation zwischen den Komponenten erfolgt hauptsächlich durch die Speicherung und den Zugriff auf relevante Daten im localStorage. Einzelne Komponenten lesen und schreiben dort essenzielle Informationen, wodurch eine indirekte Interaktion zwischen den Komponenten ermöglicht wird. Zusätzlich besteht eine weitere Kommunikationsmöglichkeit über Smart Contracts auf der Ethereum-Blockchain. Diese dienen als verbindendes Element und ermöglichen es beispielsweise dem OrderStore, relevante Daten über das Ethereum-Netzwerk zu verifizieren und zu verwalten.

5.2. Implementierung

In diesem Abschnitt wird die eigentliche Ausarbeitung des Prototyps behandelt. Dabei wird auf den Entwicklungsprozess eingegangen, dies umfasst insbesondere die Aufsetzung des Projekts, die Integration der benötigten Technologien sowie die Einbindung der Software-Architektur und ihrer Komponenten. Grundsätzlich wird in diesem Abschnitt nur dann auf den Code der Benutzeroberfläche eingegangen, wenn dies erforderlich ist. Der Schwerpunkt liegt auf der Entwicklung des SSI-Systems sowie der Einbindung der Komponenten und deren Interaktionen.

5.2.1. Veramo-Agent

Der erste Schritt bei der Einrichtung des Prototyps bestand in der Initialisierung des „React“-Frameworks. Hierfür wurde mithilfe von `yarn` und `Vite` ein initiales Projekt aufgesetzt. Die Einrichtung erfolgte über die Kommandozeile, wie in Snippet 1 dargestellt. Dabei wurde auch die Entscheidung für TypeScript getroffen. Das resultierende Projekt ist eine leere React-Anwendung, die als Grundlage für die weitere Entwicklung dient.

```
$ yarn create vite SSI-Prototype --template react
```

Snippet 1: React Projekt-Initialisierungs script

Nach der Initialisierung des Frontends wurden die für das Veramo-Framework erforderlichen Bibliotheken mithilfe von `yarn` installiert. Diese stellen die Kernfunktionalitäten zur Interaktion mit SSI bereit. Um eine erste Orientierung zu erhalten, wurde die offizielle Dokumentation von Veramo herangezogen. Insbesondere wurde das „Node-Tutorial“ verwendet, um einen **`agent`** für das Management von Identitäten zu erstellen. Dieser `agent` entspricht einer Entität innerhalb des Systems, beispielsweise einem Kunden, E-Systems oder einer staatlichen Behörde. Die Initialisierung dieses `agent` basiert auf fünf sogenannten `Plugins`: `KeyManager`, `DIDResolverPlugin`, `DIDManager`, `CredentialPlugin` und `DataStoreJson`. Diese übernehmen verschiedene Aufgaben innerhalb des Systems, die wie folgt strukturiert sind:

- `KeyManager`: Ist für die Verwaltung kryptographischer Schlüssel zuständig.
- `CredentialPlugin`: Bietet dem `agent` Funktionalitäten, um mit VCs zu arbeiten. Dazu gehören unter anderem die Erstellung, Bearbeitung, Verifizierung und Ausstellung von VCs.
- `DataStoreJson`: Ermöglicht es dem `agent`, Daten wie beispielsweise VCs und DIDs zu persistieren.
- `DIDManager`: Erlaubt dem `agent`, DIDs zu verwalten.
- `DIDResolverPlugin`: Erweitert den `agent` um die Fähigkeit, DIDs aufzulösen, um daraus ein DID-Dokument zu erhalten.

Abgesehen von den zuvor beschriebenen Funktionalitäten von `DIDResolverPlugin` und `DIDManager` umfassen diese erweiterte Komponenten eine Anbindung an eine Blockchain. Für das `DIDResolverPlugin` besteht diese Anbindung, wie in Snippet 2 dargestellt, aus einem `...ethrDidResolver`-Objekt, das ein `networks`-Objekt benötigt. Die erforderlichen Daten bestehen neben dem Namen aus einer `rpcUrl` und einer `registry`. Der Begriff `rpcUrl`[11] bezeichnet einen „Remote Procedure Call"-Endpunkt, der es Anwendungen ermöglicht, mit einem Blockchain-Netzwerk zu kommunizieren. Der Wert `registry` erwartet eine sogenannte Contract-Adresse[12]. Dabei handelt es sich um die Adresse, die ein Smart Contract nach erfolgreichem Deployment auf der Blockchain erhält, um die Kommunikation mit ihm zu ermöglichen.

```
new DIDResolverPlugin({
  resolver: new Resolver({
    ...ethrDidResolver({
      networks: [
        {
          name: "local",
          rpcUrl: "http://127.0.0.1:7545",
          registry: contractAddress,
        },
      ],
    }),
    ...webDidResolver(),
  }),
})
```

Snippet 2: Code-Aufbau des `DIDResolverPlugin`

Für den `DIDManager` besteht ebenfalls eine ähnliche Struktur, die erforderlich ist, um die Auflösung von DIDs an das zugehörige Blockchain-Netzwerk weiterzuleiten.

```
new DIDManager({
  ...
  defaultProvider: "did:ethr",
  providers: {
    "did:ethr:local": new EthrDIDProvider({
      networks: [
        {
          name: "local",
          rpcUrl: "http://127.0.0.1:7545",
          registry: contractAddress,
        },
      ],
      ...
    }),
  },
}),
```

Snippet 3: Code-Aufbau des `DIDManager`

Um die Anbindung an ein Blockchain-Netzwerk zu gewährleisten, wurde Ganache verwendet. Ganache ermöglicht durch die Nutzung der Windows-Anwendung die Erstellung lokaler Ethereum-Netzwerke, die für Entwicklungs- und Testzwecke konzipiert sind. Da es sich bei dieser Implementierung um einen Prototyp handelt, bietet sich ein lokales Testnetzwerk optimal an. Um mit Ganache erfolgreich ein lokales Ethereum-Testnetzwerk aufzusetzen, muss lediglich der **Quickstart**-Button in der Benutzeroberfläche betätigt werden. Anschließend erstellt Ganache ein Ethereum-Netzwerk mit zehn Accounts, die jeweils mit 100 Ethereum ausgestattet sind. Darüber hinaus stellt Ganache über die Benutzeroberfläche die in Snippet 3 und Snippet 2 benötigte `rpcUrl` bereit.

Um die Einrichtung des lokalen Ethereum-Netzwerks abzuschließen und die Integration mit Veramo zu vervollständigen, wird ein Smart Contract benötigt. Dieser ist dafür zuständig, die vom `agent` ausgeführten Funktionen korrekt an das Ethereum-Netzwerk zu übermitteln. Als Smart Contract[10] bezeichnet man Programme, die in einer Blockchain gespeichert sind und automatisch ausgeführt werden, sobald bestimmte Bedingungen erfüllt sind. Veramo stellt hierfür einen Smart Contract namens `EthereumDIDRegistry.sol` bereit.

Um diesen in das bestehende System zu integrieren, muss er im Ethereum-Netzwerk bereitgestellt werden – ein Prozess, der als „Deployment“[13] bezeichnet wird. Für das Deployment von `EthereumDIDRegistry.sol` wird Truffle verwendet. Truffle bietet über die Kommandozeile Funktionen zur Interaktion mit Ethereum-Netzwerken, einschließlich der Bereitstellung von Smart Contracts.

Zunächst muss der Smart Contract mit dem Befehl `$ truffle compile` kompiliert werden. Anschließend erfolgt das Deployment in das Netzwerk mit `$ truffle migrate --network development`. Das Ergebnis dieser Skripte ist eine `contractAddress`. Der Begriff `contractAddress`[12] bezeichnet eine eindeutige Kennung, die erforderlich ist, um mit dem Smart Contract zu kommunizieren. Diese Adresse muss in den beiden relevanten Plugins des Veramo-`agent` eingebunden werden.

Die zuvor genannten Schritte führen letztlich zur erfolgreichen Erstellung eines `agent`, wie in Snippet 4 dargestellt. Die Erstellung dieses `agent` ermöglicht nun die Hauptfunktionen von SSI, wie beispielsweise das Erstellen, Speichern und Löschen von DIDs sowie das Erstellen, Speichern, Löschen und Ausstellen von VCs. Darüber hinaus ermöglicht die Nutzung des lokalen Ethereum-Netzwerks die fortlaufende Implementierung von Smart Contracts.

```
agent = createAgent
        <IDIDManager &
        IKeyManager &
        IDataStore &
        IDataStoreORM &
        IResolver &
        ICredentialPlugin>
    ({
      plugins: [...],
    });
```

Snippet 4: Erstellung des `agent` mit Veramo

5.2.2. Decentralized Identifiers

Um die im Abschnitt 4.1 erwähnten Interaktionen zwischen den verschiedenen Teilnehmern des Systems zu ermöglichen, mussten die in Abschnitt 5.1 erstellten Komponenten in der Anwendungsschicht entwickelt werden. Ein wichtiger Aspekt des Prototypens war die Nutzung von VCs zur Authentifizierung gegenüber E-Systems. Hierfür waren jedoch die erfolgreiche Erstellung, Speicherung und Verifizierung von VCs sowie die Existenz und Verwaltung von DIDs als Voraussetzung erforderlich. Um diese Voraussetzungen zu erfüllen, wurden verschiedene Komponenten erstellt, welche die zuvor beschriebenen `agent`-Funktionalitäten nutzen.

Eine dieser Komponenten ist die `CreateDID.tsx`-Komponente. Abgesehen von der Definition der Benutzeroberfläche stellt diese Komponente auch das Erstellen von DIDs dar. Hierfür wird unter `alias` die Eingabe des Nutzers verwendet, um eine neue DID für diesen `alias` zu erstellen. Dies spiegelt sich in der Abbildung 7 unter dem Eingabefeld wieder.

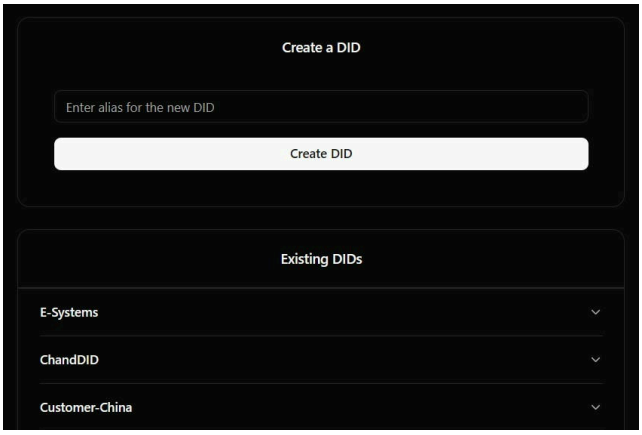


Abbildung 7: Benutzeroberfläche der DID-Erstellungs Komponente

Der Ablauf dieser Interaktion ist in Abbildung 8 nachzuvollziehen. Hierbei wird intern die Nutzung des zuvor erstellten `agent` hervorgerufen, welcher durch `agent.didManagerCreate` eine neue DID erzeugt. Diese neu erstellte DID wird durch den zuvor genannten `DataStoreJson`-Teil des `agent` automatisch im localStorage gespeichert und durch die vorherige Konfiguration des Ethereum-Netzwerks über `provider`, wie in Snippet 5 zu sehen, angebunden.

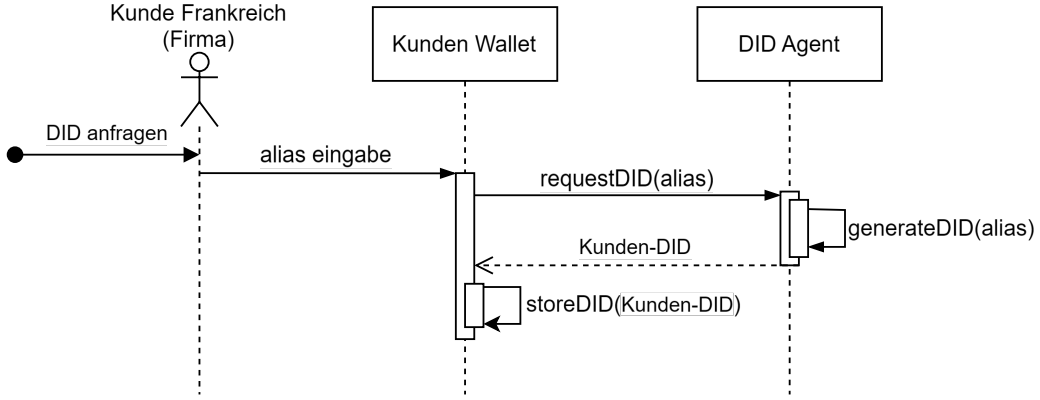


Abbildung 8: Vollständiges Sequence-Diagram zur Erstellung einer DID

Dies spielt eine wichtige Rolle, da hierdurch die Identitätsrepräsentation im Prototypen, die eigentlich durch eine Wallet implementiert werden sollte, stattfindet. Diese Funktionalität wird durch die in Abbildung 6 abgebildete Komponente `Wallet Interface` und den `DID-User Store` dargestellt. Da, wie eben angemerkt, alle erstellten DIDs lokal gespeichert werden, nutzt die `Wallet Interface`-Komponente über den `DID-User Store` diese gespeicherten Einträge, um die Nutzer-Auswahl und Repräsentation zu ermöglichen.

```
createIdentifierVera(alias: string) {
  const identifier = await agent.didManagerCreate({
    alias: alias,
    provider: "did:ethr:local",
    kms: "local",
  });
}
```

Snippet 5: Erstellung einer DID mithilfe des Veramo `agent`

5.2.3. Verifiable Credentials

Da die Verwaltung und Erstellung von DIDs ermöglicht wurde, minimieren sich die zuvor erwähnten Voraussetzungen auf die Verwaltung von VCs. Die Anbindung zur Ermöglichung der Verwaltung von VCs erwies sich als ein umfangreicher Teil des Prototypens. Hierbei teilte sich die Aufgabe in vier Unterkategorien auf: die Speicherung, die Beantragung, die Ausstellung und die Verifizierung von VCs.

Durch die vorherig erwähnte Integration des `DataStoreJson`-Plugins erwies sich die Speicherung von VCs als eine einfache Umsetzung. Hierbei wird letztendlich durch die Nutzung der `agent`-Methode `agent.dataStoreSaveVerifiableCredential({verifiableCredential})` ein bereits erstelltes Credential im lokalen Speicher abgelegt.

5.2.3.1. Beantragung

Für die Beantragung eines VC sind keine spezifischen Funktionen des Veramo-`Agent` erforderlich. Der Prozess beginnt mit der Auswahl eines Issuers, welcher in Abbildung 9 als `Select Recipient` dargestellt wird. Zu diesem Zweck wurde eine lokale Speicherinstanz implementiert, die für die Verwaltung der Zustände aller VCs verantwortlich ist. Diese Instanz entspricht dem in Abbildung 6 dargestellten `VC Store`. Der `VC Store` verwaltet lokale VCs Instanzen, welche der Form in Snippet 6 entsprechen.

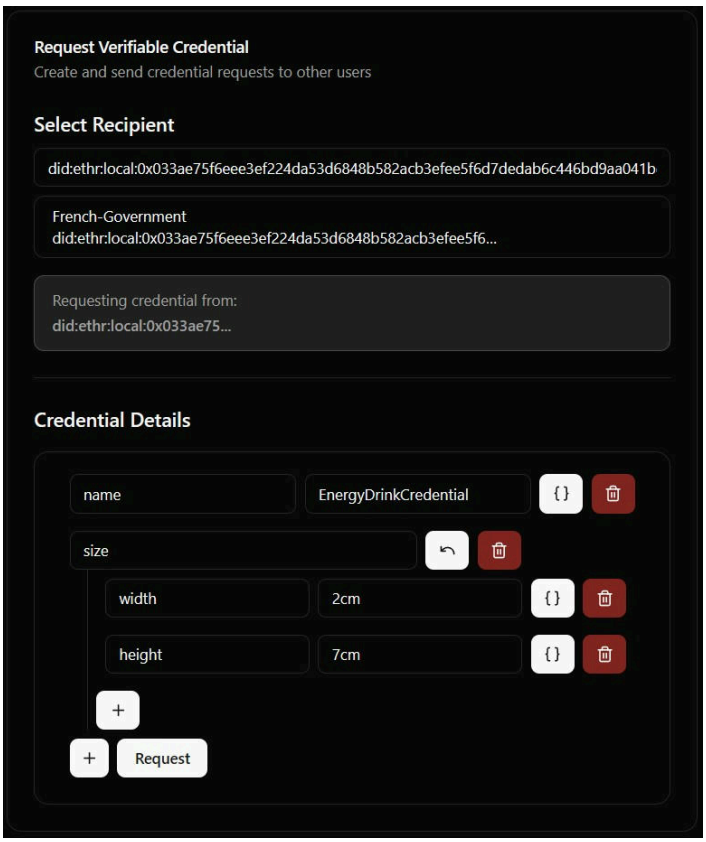


Abbildung 9: Benutzeroberfläche zur Beantragung eines VCs

```
interface VC {
  isIssued?: boolean;
  id: string;
  orderId?: number;
  credentialSubject: CredentialSubject;
  requestedFrom?: string;
  vcHash?: string;
  isVcClaimed?: boolean;
  isVcRequested?: boolean;
}
```

Snippet 6: Struktur eines `VC` Objekts des `VC Store`

Es wurden verschiedene Variablen erstellt, die für die Verwaltung der VCs zuständig sind, beispielsweise `credentialSubject`, `isVcRequested` und `requestedFrom`, die speziell für die Beantragung eines VC verantwortlich sind. Dabei spiegelt `credentialSubject` die Eingabedaten des Nutzers bezüglich der gewünschten Aussagen des Zertifikats wider, `requestedFrom` stellt den Issuer dar, und

`isVcRequested` repräsentiert schließlich, ob sich die aktuelle VC Instanz im Status der Beantragung befindet.

Um nun ein VC in den Status der Beantragung zu versetzen, wird nach Betätigung des `Request` Buttons in Abbildung 9 die zugehörige Funktion `addVC(credentialSubject,isVcRequested,requestedFrom)` des `VC Store` aufgerufen. Dadurch entsteht ein neuer Eintrag in der lokalen Speicherinstanz, den die Ausstellungs-Komponente nutzen kann, um VC Elemente zu finden, deren `isVcRequested` auf `true` gesetzt wurde.

Abgesehen von den positiven Fällen wurde mit der Implementierung der Methoden in Snippet 7 ebenfalls für die korrekte Behandlung der anderen Fälle gesorgt, wobei die Implementierung ähnlich wie in `addVC()` erfolgte.

```
getVC: (id)
updateVC: (id, updates)
deleteVC: (id)
clearAllVCs: ()
getVCsByFilter: (filter)
```

Snippet 7: Verwaltungsmethoden des `VC Store` für `VC` Instanzen

5.2.3.2. Ausstellung

Die Ausstellung eines VCs erfordert einige Daten, welche der Nutzer, wie in Abbildung 9 zu sehen, über Benutzeroberflächenelemente zur Verfügung stellt. Dies wird auch in Abbildung 10 durch die Methode `requestVC(KundenDID, credentialData)` dargestellt. Diese Daten wären eine DID, die den `issuer`, also den Aussteller des VCs, darstellt, eine Auswahl des VC-Typs `type`, ein `proofFormat`, welches sich auf JWT bezieht, und ein `credentialSubject`-Objekt, welches ein Objekt aus Key-Value-Paaren ist, das durch JSON repräsentiert wird.

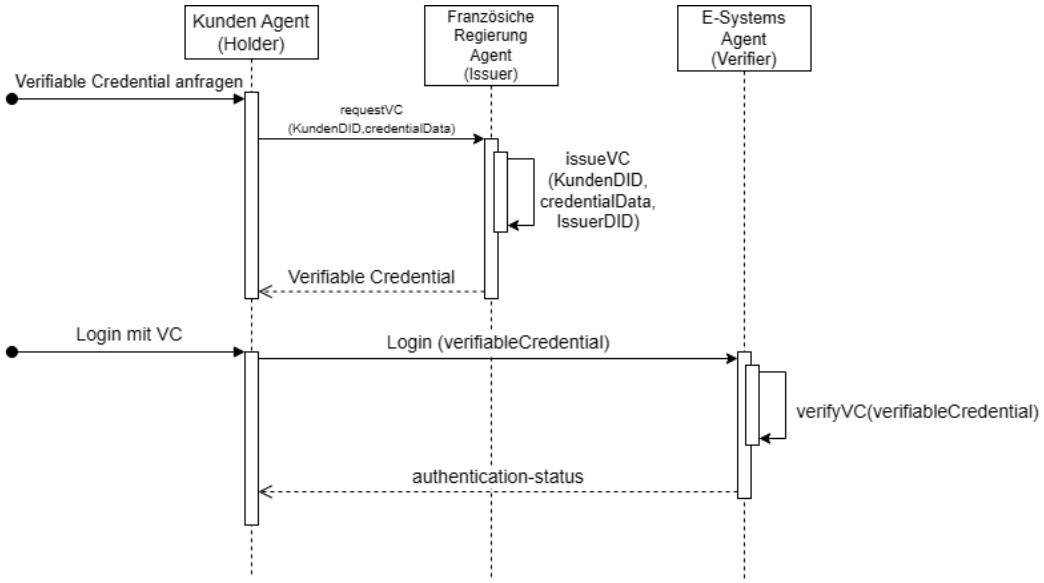


Abbildung 10: Vollständiges Sequence-Diagram zur Anforderung und Ausstellung eines VCs gefolgt von der Nutzung zum Login mit diesem VC

Abgesehen von der `issuer` DID und dem `credentialSubject`, welche wie in Abbildung 9 durch den Nutzer bestimmt werden, sind die restlichen Werte im Prototypen manuell festgelegt. So bot sich bei der Wahl des `type` der Wert „ProductCredential“ optimal an, und die Wahl des `proofFormat` wurde als JWT fest ausgewählt.

Der vollständige Überblick zur Erstellungssyntax eines VCs ist in Snippet 8 zu finden, wobei `credentialSubject` und `issuer` sich in Abbildung 9 als `Credential Details` Sektion und dem Eingabe Feld unter dem Titel „Select Receipient“ wiederfinden lassen.

```
createVerifiableCredential({
    credential: {
        issuer: issuer.did,
        type: ["VerifiableCredential", "ProductCredential"],
        credentialSubject: subject,
    },
    proofFormat: "jwt",
})
```

Snippet 8: Erstellung eines VC mithilfe des Veramo `agent`

5.2.3.3. Verifizierung

Durch die zuvor ausgeführten Schritte besteht bereits die Möglichkeit, mit SSI-Identitäten und Zertifikaten in Form von DIDs und VCs zu interagieren. Eine der Komponenten, welche die ermöglichten Funktionalitäten effektiv nutzt, ist die Login-Komponente. Diese verwendet die Methode `agent.verifyCredential({credential: vc})`, um ein übergebenes `vc`, wie beispielsweise in Abbildung 11, zu verifizieren.

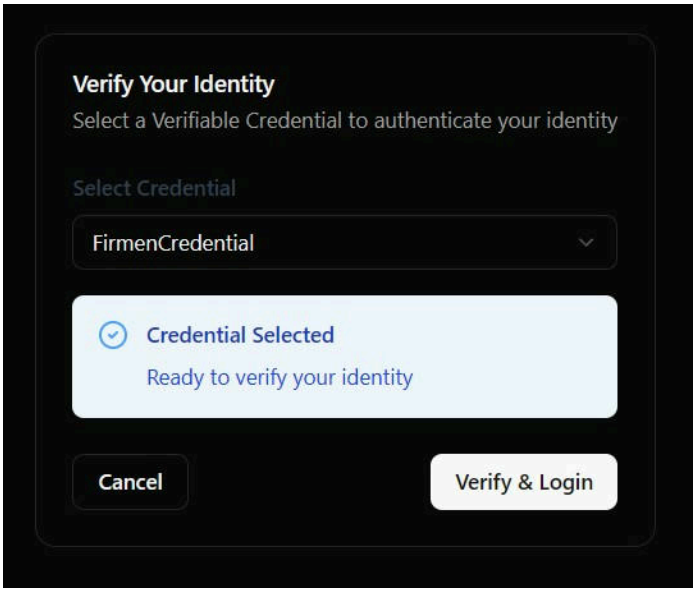


Abbildung 11: Login-Komponente Benutzeroberfläche

Basierend auf diesem Ergebnis wird die Autorisierung für die Interaktion mit nachfolgenden Komponenten, wie beispielsweise der Bestellübersicht, gewährleistet. Dies geschieht durch die aktive Nutzung des `Authentication-Store`. Ein `store` ist hierbei für die Datenhaltung bestimmter Werte zuständig, die entweder temporär oder persistent gespeichert werden können. Im Falle des Prototypens wurde für alle `store`-Implementationen die persistente Speicherform genutzt. Durch die Verifizierung des VCs und die Anbindung an den `Authentication-Store` besteht nach Nutzung der Login-Komponente ein Zustand namens `isAuthenticated`, welcher durch die Persistierung global verwaltbar ist.

5.2.4. Blockchain, SmartContracts und Bestellungen

Abgesehen von der Verwaltung SSI-relevanter Komponenten besteht ein Bedarf an weiteren Komponenten, die sich mit bestellungsrelevanten Interaktionen befassen, wie in Abbildung 6 ersichtlich ist. Diese Komponenten sind die `Bestellübersicht`, der `OrderStore` und der dazugehörige `OrdersSmartContract`. Die folgende Grafik Abbildung 12 stellt den generellen Ablauf dieser Interaktion dar und sollte beim Lesen dieser Sektion referenziert werden.

Die `Bestellübersicht`-Komponente befasst sich umfassend mit der Verarbeitung sogenannter `Order`-Objekte und bietet hauptsächlich die Benutzeroberflächeninteraktion an. Diese `Order`-Objekte werden genutzt, um eine Instanz einer Bestellung zu repräsentieren. Ein `Order`-Objekt besteht aus verschiedenen Attributen, die der weiteren Verwaltung von Bestellungen und deren Interaktionsmöglichkeiten dienen. Eine Übersicht zur Struktur eines `Order`-Objekts ist in Snippet 9 zu finden. Des Weiteren bietet die `OrderStore`-Komponente die folgenden Verwaltungsfunktionen für `Order`-Objekte an:

- `createOrder` - zum erstellen von Bestellungen.
- `acceptOrder` - zum akzeptieren von Bestellungen.
- `declineOrder` - zum ablehnen von Bestellungen.
- `cancelOrder` - zum stornieren von Bestellungen.
- `completeOrder` - zum erfüllen von Bestellungen.
- `fetchOrders` - - zum abrufen aller Bestellungen.

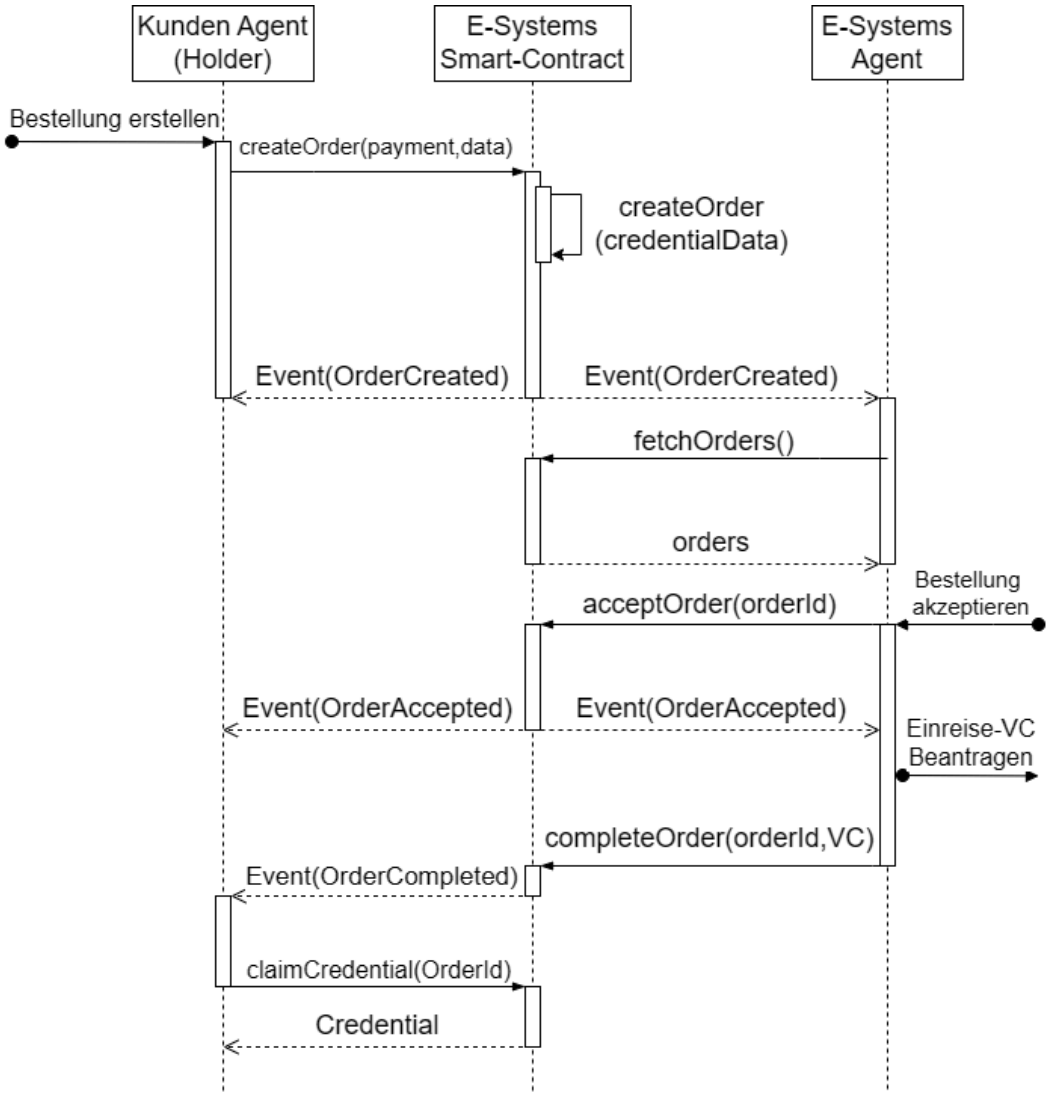


Abbildung 12: Vollständiges Sequence-Diagram zur Erstellung und Bearbeitung einer Bestellung

```
interface Order {
  id: number;
  user: string;
  payment: string;
  data: string;
  goods: string;
  status:| "Requested" | "Accepted" | "Declined" | "Pending" |
"Completed" | "Canceled";
  createdAt: number;
  acceptedAt?: number;
}
```

Snippet 9: Struktur eines `Order`-Objektes des `VC Store`

Um die Funktionalität der `OrderStore`-Komponente zu gewährleisten, wird eine Smart-Contract-Instanz benötigt, die dieselben Verwaltungsfunktionen bietet. Dieser Smart Contract ist in Abbildung 6 als `OrdersSmartContract` repräsentiert. Der `OrdersSmartContract` musste, ähnlich wie bei der Erstellung des `EthereumDIDRegistry.sol`-Smart Contracts, in das zugrunde liegende Ethereum-Netzwerk eingebunden werden.

Ein wesentlicher Aspekt ist die Interaktion der relevanten Daten zwischen den On-Chain- und Off-Chain-Komponenten. Als On-Chain werden Entitäten bezeichnet, die sich auf einer Blockchain befinden, während Off-Chain-Komponenten außerhalb der Blockchain liegen. Die Kommunikation zwischen diesen beiden Arten von Komponenten erfolgte durch die Interaktion zwischen `OrderStore` und `OrdersSmartContract`. Um diese Kommunikation zu gewährleisten, musste eine Verbindung erstellt werden, wie in Snippet 10 dargestellt.

```
const contractAddress =
"0x4acce4ab8ce135f8df5627D40dbf93d55A5D9bEB";
const web3 = new Web3("http://127.0.0.1:7545"); // Ganache URL
const ssiContract = new web3.eth.Contract(contractABI.abi,
contractAddress);
```

Snippet 10: Verbindungsherstellung zum `OrdersSmartContract`

Durch die erfolgreiche Einrichtung der Kommunikationsmöglichkeit zwischen `OrderStore` und dem `OrdersSmartContract` wurde die Grundlage für die Implementierung der Verwaltungsfunktionen des `OrdersSmartContract` geschaffen.

Da in einem Smart Contract der Austausch zwischen einer Zahlung und dem Erhalt einer Leistung für eine automatisierte Abwicklung prädestiniert ist, bestand die Anforderung, diese Funktionalitäten im `OrdersSmartContract` umzusetzen. Hierfür wurden drei wesentliche Anforderungen definiert: die Bereitstellung der zu verarbeitenden Daten in Form eines `credentialSubject`, die Einreichung eines `Einreise-VC` zur Vervollständigung einer Bestellung und die Eingabe einer Zahlungsmenge für den Austausch. Diese Anforderungen wurden über die `Bestellübersicht` von den jeweils beteiligten Parteien angefordert.

Durch die Nutzung der Funktion `createOrder(data,paymentEth)` des `OrderStore` wird die zugrunde liegende Methode `ssiContract.methods.createOrder(data)`, wie in Snippet 11 dargestellt, aufgerufen. Diese Methode übernimmt die Kommunikation mit dem `OrdersSmartContract` und erstellt daraufhin eine Bestellung, die nun On-Chain abrufbar ist. Das Ergebnis dieser Interaktion ist ein „Created“-Eintrag im

`OrdersSmartContract`, wie in Abbildung 13 ersichtlich. Somit ist die Kommunikation zwischen On-Chain- und Off-Chain-Komponenten erfolgreich gewährleistet.

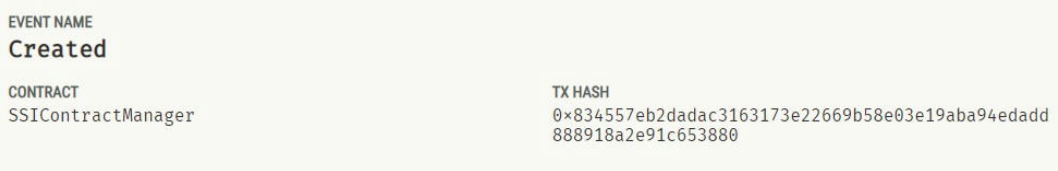


Abbildung 13: Resultat des `OrdersSmartContract` bei Erstellung einer Bestellung

```
await ssiContract.methods.createOrder(data).send({
  from: account,
  value: paymentWei,
  gas: "500000",
});
```

Snippet 11: Erstellung einer Bestellung mit `OrderStore`

Ein weiterer zentraler Aspekt zur Vervollständigung der in Abbildung 4 dargestellten Interaktionen ist die abschließende Bearbeitung einer Bestellung. Für diesen Vorgang ist die Funktion `completeOrder(orderId, goods)` des `OrdersSmartContract` zuständig, die durch den Aufruf der Methode `completeOrder(orderId,goods)` des `OrderStore` initiiert wird.

Die in Snippet 12 dargestellte Funktion ermöglicht die abschließende Verarbeitung einer Bestellung mithilfe der `orderId`, die als eindeutiger Identifikator der Bestellung dient, sowie des `goods`-Objekts, das das ausgestellte VC in Form eines `string` repräsentiert. Die Variable `gas`[14] gibt hierbei die Transaktionskosten für Interaktionen im Ethereum-Netzwerk an. Diese Kosten stellen eine Vergütung für die Instandhaltung des Netzwerks dar.

Durch den erfolgreichen Abschluss dieser Funktion wird die `goods`-Variable des entsprechenden `Order`-Objekts, wie in Snippet 13 dargestellt, mit dem ausgestellten VC befüllt. Um den On-Chain-Datenaustausch abzuschließen, muss der anfragende Teilnehmer der Bestellung lediglich die `goods`-Variable seiner Bestellung abrufen, wodurch er das angeforderte VC erhält. Durch die in Snippet 12 gezeigte Funktion wird somit die `goods`-Variable belegt und die im `OrdersSmartContract` hinterlegte Zahlung an den Abschlussberechtigten der Bestellung ausgezahlt.

```
await ssiContract.methods.completeOrder(orderId, goods).send({
        from: accounts[currentAccountIndex],
        gas: gasLimit.toString(),
});
```

Snippet 12: Vollendung einer Bestellung mit `OrderStore`

```
struct Order {
    address user;
    uint96 payment;
    uint32 createdAt;
    uint32 acceptedAt;
    uint8 status;
    string data;
    string goods;
}
```

Snippet 13: `Order` Objektstruktur im `OrdersSmartContract`

Durch die Umsetzung dieser Implementierungsschritte konnten die in Abschnitt 4.2 und Abschnitt 4.1 erfassten Anforderungen sowie die in Abschnitt 5.1 definierte Architektur in einen funktionalen Prototypen überführt werden.

5.3. Schwierigkeiten und Probleme

Der folgende Abschnitt befasst sich mit den Problemen und Schwierigkeiten, die während der Ausarbeitung aufgetreten sind. Diese werden in drei Punkte unterteilt: das aufgetretene Problem, der Grad der Auswirkungen dieses Problems auf die Ausarbeitung und schließlich die Lösung, die zur Behebung des Problems eingesetzt wurde.

Problem: Die initiale Aufsetzung des `React`-Projektes nach Befolgung der offiziellen Veramo-Dokumentation erwies sich als problematisch, da die Dokumentation veraltet war und dadurch zu einem nicht funktionalen initialen Projekt führte.
Grad der Auswirkung: Dies hatte eine erhebliche Auswirkung auf die initiale Entwicklung des Prototyps, da durch die Problematik zwei Tage für die Lösung aufgewendet werden mussten.
Lösung: Zur Lösung dieses Problems wurde Kontakt zu den Instandhaltern des Veramo-Projektes aufgenommen. Die fehlerhaften Punkte wurden durch eine saubere Neuaufsetzung des React-Projektes schrittweise identifiziert und durch funktionsfähige Alternativen ersetzt.

Problem: Die Kompilierung von Solana-Smart Contracts mit Truffle führte bei der Nutzung älterer Compiler-Versionen zu nicht funktionsfähigen Ergebnissen.
Grad der Auswirkung: Die Auswirkungen waren gering, da der Lösungsansatz durch eine kurze Recherche zu Solana-Compilern schnell gefunden werden konnte.
Lösung: Die Aktualisierung des Solana-Compilers ermöglichte die erfolgreiche Kompilierung und Nutzung der selbst erstellten Smart Contracts.

Problem: Die initiale Nutzung von JavaScript brachte Probleme mit sich, da die Interaktion mit vordefinierten Objekttypen aus anderen Bibliotheken nicht möglich war.
Grad der Auswirkung: Dies hatte einen starken Einfluss auf die Ausarbeitung, da die Nutzung von Objekttypen die Entwicklung sicherer und effizienter gestaltet.
Lösung: Die Lösung bestand darin, TypeScript zu verwenden, um die Interaktionen mit den Objekttypen zu ermöglichen.

Problem: Die Ausarbeitung einer logischen und nutzungsfähigen Benutzeroberfläche, die den Umfang des Prototyps effektiv widerspiegelt, erwies sich als problematisch. Dies lag daran, dass aufgrund der Vielzahl an Interaktionsmöglichkeiten oft auch eine Vielzahl an Benutzeroberflächenelementen erstellt werden musste.
Grad der Auswirkung: Dies hatte einen erheblichen Einfluss auf die Entwicklung, da aufgrund der zeitlichen Begrenzung ein unvermeidbarer Arbeitsaufwand entstand.
Lösung: Das Problem wurde durch die Nutzung von Benutzeroberflächen-Bibliotheken wie `Shadcn/UI` gelöst, die vordefinierte Benutzeroberflächenelemente bieten, die vollständig anpassbar und flexibel einsetzbar sind.

6. Fazit

Durch die Durchführung dieser Arbeit konnten verschiedene Ergebnisse erzielt werden, die in den folgenden Abschnitten näher betrachtet werden. Diese Sektion gliedert sich in eine Zusammenfassung sowie einen Ausblick. Im Abschnitt zur Zusammenfassung wird auf die Zielerreichung, die Implementierung, die Verfügbarkeit von Ressourcen sowie die Fallstudie eingegangen. Der Ausblick hingegen thematisiert die Weiterentwicklung von SSIs-Technologien, mögliche Kompatibilitäten mit SSI sowie zukünftige Perspektiven.

6.1. Zusammenfassung

Die Zielsetzung dieser Arbeit bestand in der erfolgreichen Implementierung von SSI in einer von E-Systems angebotenen Fallstudie in Form eines Prototyps. Um dieses Ziel zu erreichen, waren mehrere Schritte erforderlich: Zunächst wurde eine umfassende Recherche zu aktuellen SSIs-Technologien durchgeführt, gefolgt von der Konzeption einer geeigneten Softwarearchitektur. Abschließend wurde diese Architektur in einem funktionalen Prototyp umgesetzt. Die Recherche basierte überwiegend auf Online-Ressourcen wie Fachartikeln, Dokumentationen und Code-Beispielen. Dabei zeigte sich, dass die Verfügbarkeit und Vielfalt der Quellen den Reifegrad von SSI widerspiegelt. Insbesondere wurde deutlich, dass das Thema aktiv weiterentwickelt wird und bereits in zahlreichen Variationen und Anwendungsformen existiert.

Neben den Anforderungen, die sich aus der Zielsetzung ergaben, wurden auch bestimmte erwartete Ergebnisse definiert, wie beispielsweise die Entwicklung eines Prototyps. Dieser Prozess verlief jedoch nicht ohne Herausforderungen, insbesondere im Bereich der Recherche zu SSIs-Technologien. Es wurde deutlich, dass sich diese Technologien weiterhin in aktiver Entwicklung befinden, was sich unter anderem in der mangelnden Interoperabilität verschiedener Lösungen, beispielsweise bei Wallets, widerspiegelte. Trotz dieser Herausforderungen konnten alle Zielvorgaben mit Ausnahme der Implementierung von ZKPs und Selective Disclosure erreicht werden. Darüber hinaus führten die Implementierung und Ausarbeitung von Smart Contracts auf der Ethereum-Blockchain zu innovativen Ergebnissen, die in dieser Form bislang nicht in SSI integriert waren.

Die Umsetzung der definierten Ziele in konkrete Ergebnisse erwies sich als anspruchsvoll, da die Entwicklung eines Prototyps mit erheblichem Aufwand verbunden war. Dies zeigte sich insbesondere in der Erstellung und Konzeption einer geeigneten Softwarearchitektur. Beide Aspekte wurden durch mehrere Iterationen weiterentwickelt und optimiert, bis schließlich die finale Version entstand, die für die Implementierung des Prototyps genutzt wurde. Ein zentraler Faktor für diesen iterativen Prozess war der Reifegrad der verfügbaren SSIs-Technologien. Infolgedessen mussten mehrere Ansätze nach initialem Entwicklungsaufwand verworfen werden, da sich im Nachhinein herausstellte, dass sie nicht den erforderlichen Funktionsumfang für eine erfolgreiche Implementierung boten.

Durch die Entwicklung des finalen Prototyps konnte der aktuelle Stand der SSI-Technologie erfolgreich analysiert werden. Die zuvor beschriebenen Herausforderungen verdeutlichen, dass sich SSI derzeit in einer fortgeschrittenen Entwick-

lungsphase befindet. Der Prototyp bietet eine umfassende Einsicht in zentrale Aspekte der Technologie, darunter die einfache Verwaltung von DIDs und VCs, die stabile und sichere Kommunikation mittels DIDComm sowie die nahtlose Anbindung von Smart Contracts und der Ethereum-Blockchain. Allerdings wurden auch Schwachstellen identifiziert, wie die begrenzte Verfügbarkeit von Wallet-Implementierungen und deren mangelnde Interoperabilität mit bestehenden SSIs-Lösungen.

Zusammenfassend lässt sich feststellen, dass sich SSI in den vergangenen Jahren kontinuierlich weiterentwickelt und technologisch bedeutende Fortschritte erzielt hat. Dennoch besteht weiterhin ein Bedarf an höherer Stabilität und einer breiteren Auswahl an Plattformen. Daher kann der Entwicklungsstand der SSIs-Technologien als „experimentell“ eingestuft werden, da die Analyse gezeigt hat, dass sie den Herausforderungen ausgereifter Industrielösungen noch nicht vollständig gewachsen sind.

6.2. Ausblick

Abgesehen vom Reifegrad der Technologie erwies sich die Entwicklung eines vollständigen Prototyps innerhalb von drei Monaten als nahezu realisierbar. Mit etwas mehr verfügbarer Zeit wäre auch die vollständige Implementierung von ZKPs und Selective Disclosure gut vorstellbar gewesen. Darüber hinaus eröffnet die Natur der SSI-Technologie zahlreiche Möglichkeiten für zukünftige Erweiterungen und Interaktionen.

Ein Beispiel hierfür ist die bereits im Prototyp umgesetzte Nutzung von Smart Contracts zur automatischen Ausführung von Aktionen unter bestimmten Bedingungen. Ein weiterer vielversprechender Aspekt wäre die Integration eines KI-Agenten, der den Prozess der Erfassung von Credential-Informationen erheblich optimieren könnte. Da dieser Prozess häufig mit einem hohen manuellen Aufwand verbunden ist – insbesondere durch die Verarbeitung einer Vielzahl von Dokumenten und Zertifikaten –, könnte ein KI-Agent diesen Arbeitsaufwand erheblich reduzieren. Er könnte Dokumente in Form von PDFs, Bildern oder Fließtexten analysieren, relevante Informationen extrahieren und nach vordefinierten Kriterien filtern. Das Ergebnis wäre eine effiziente und präzise Extraktion wichtiger Credential-Daten aus mehreren Informationsquellen.

Ein weiterer interessanter Aspekt ist die zukünftige Entwicklungsrichtung von SSI-Technologien. Sollte sich SSI weiterhin mit der gleichen Dynamik wie in den vergangenen Jahren weiterentwickeln, ist davon auszugehen, dass sich die

verfügbaren Technologien und Plattformen kontinuierlich verbessern werden.

Konkrete Beispiele für diese Entwicklung wären eine vereinfachte Implementierung und Interoperabilität neuer und bestehender Wallet-Lösungen, die endgültige Standardisierung von DIDs und VCs sowie die Entwicklung einer Blockchain-Technologie, die speziell für die Nutzung von SSI konzipiert ist. Eine solche spezialisierte Blockchain könnte die verfügbaren und potenziellen Interaktionen innerhalb von SSI gezielt optimieren. Dies könnte wiederum zu einer stärkeren Verbreitung der Technologie beitragen, da eine klar definierte Struktur für ihre Nutzung geschaffen würde.

Bibliographie

[1] BOSCH, „Digital identity – enabling secure collaboration with blockchain technology“. Zugegriffen: 4. März 2025. [Online]. Verfügbar unter: https://www.bosch.com/stories/self-sovereign-identities/

[2] W3C, „Decentralized Identifiers (DIDs) v1.0“. Zugegriffen: 16. März 2025. [Online]. Verfügbar unter: https://www.w3.org/TR/did-1.0/

[3] A. Mühle, A. Grüner, T. Gayvoronskaya, und C. Meinel, „A survey on essential components of a self-sovereign identity“. [Online]. Verfügbar unter: https://www.sciencedirect.com/science/article/pii/S1574013718301217

[4] M. Dieye *u. a.*, „A Self-Sovereign Identity Based on Zero-Knowledge Proof and Blockchain“.

[5] A. De Salve, A. Lisi, P. Mori, und L. Ricci, „Selective Disclosure in Self-Sovereign Identity based on Hashed Values“.

[6] Hyperledger Foundation, „Hyperledger Aries – Infrastruktur für dezentrale Identitäten“. Zugegriffen: 12. März 2025. [Online]. Verfügbar unter: https://www.hyperledger.org/use/aries

[7] didkit, „DIDKIT GitHub“. Zugegriffen: 16. März 2025. [Online]. Verfügbar unter: https://github.com/spruceid/didkit

[8] Veramo, „Veramo Docs“. Zugegriffen: 16. März 2025. [Online]. Verfügbar unter: https://veramo.io/docs/basics/introduction

[9] W3C, „Verifiable Credentials Data Model v1.1“. Zugegriffen: 16. März 2025. [Online]. Verfügbar unter: https://www.w3.org/TR/vc-data-model/

[10] IBM, „Smart Contracts definiert“. Zugegriffen: 11. März 2025. [Online]. Verfügbar unter: https://www.ibm.com/de-de/topics/smart-contracts

[11] Thirdweb, „RPC URLs“. Zugegriffen: 11. März 2025. [Online]. Verfügbar unter: https://portal.thirdweb.com/glossary/rpc

[12] alchemy, „How do I distinguish between a contract address and a wallet address?“. Zugegriffen: 10. März 2025. [Online]. Verfügbar unter: https://docs.alchemy.com/reference/contract-address-vs-wallet-address#:~:text=A%20contract%20address%20is%20a,when%20specific%20conditions%20are%20met.

[13] DevHub, „Deploy Your First Smart Contract“. Zugegriffen: 10. März 2025. [Online]. Verfügbar unter: https://docs.chain.link/quickstarts/deploy-your-first-contract#:~:text=Compile%3A%20Pass%20your%20smart%20contract,smart%20contract%20to%20the%20blockchain.

[14] Coinbase, „What are gas fees?“. [Online]. Verfügbar unter: https://www.coinbase.com/en-de/learn/crypto-basics/what-are-gas-fees#:~:text=Gas%20fees%20are%20transaction%20costs,during%20periods%20of%20network%20congestion.