Bachelorarbeit

ZHAW School of Management and Law

Anwendung der Blockchain-Technologie im Vertragsmanagement

Autor:	Manuel Weiss | 16-561-425
Betreuer:	Florian Spychiger
Studiengang:	Wirtschaftsinformatik
Datum:	23. Mai 2019

Management Summary

Verträge bilden die Grundlage aller Geschäftsvorfälle und deren Verwaltung stellt eine alltägliche administrative Arbeit für Unternehmen dar. Ein Vertrag wird abgeschlossen, um die gegenseitige Willenserklärung der Vertragsparteien rechtskräftig festzuhalten. Infolgedessen existiert bei allen Parteien das Interesse, eine rechtlich bindende Vertragsversion in eigenem Besitz aufzubewahren. Aktuell findet die Verwahrung mittels physischen oder digitalen Vertragskopien in einer Ablage oder Datenbank, über eine zentralisierte geteilte Plattform oder durch die Inanspruchnahme eines Drittservices statt. Diese Lösungen haben den Nachteil, dass entweder mehrere gültige Vertragsversionen bestehen oder darauf vertraut werden muss, dass die geteilte Plattform von der innehabenden Partei nicht kompromittiert wird. Nachteile beim Drittservice entstehen durch hohe Kosten und eine zeitaufwendige Vertragsentstehung.

Diese Arbeit umfasst die Konzipierung, Entwicklung und kritische Würdigung eines Prototyps im Rückvergütungs- und Darlehensvertragsmanagement einer Schweizer Brauerei. Mit der Realisation des Prototyps wird untersucht, wie durch die Anwendung der Blockchain-Technologie eine integre Aufbewahrung einer einzig gültigen Vertragsversion umgesetzt werden kann. Darüber hinaus analysiert die Arbeit Optimierungen im Vertragsentstehungsprozess in Bezug auf Effizienz, Transparenz, Rückverfolgbarkeit und Automation.

Die zu Beginn durchgeführte Literaturrecherche wurde dazu verwendet, die Anwendung der Blockchain-Technologie in einem Business-Ökosystem zu bewerten und Vorgehensmodelle sowohl für die Realisation des Prototyps als auch zur Evaluation der geeigneten Blockchain-Plattform zu erstellen. Mit den Ergebnissen aus Analyse und Planung wurde darauffolgend ein auf der Corda-Plattform basierender Prototyp entwickelt und verifiziert.

Aus der Verifizierung des Prototyps konnte geschlossen werden, dass die Blockchain-Technologie durch ihre dezentrale Datenspeicherung die Haltung einer einzig gültigen sowie konsistenten Vertragsversion für alle Vertragsparteien ermöglicht. Die Unveränderbarkeit der Daten in einer Blockchain konnte dazu eingesetzt werden, eine verbesserte Rückverfolgbarkeit und eine erhöhte Transparenz der Verträge zu erreichen sowie einen Informationsverlust bei Vertragsüberträgen zu verhindern. Anhand der Entwicklung eines CorDapps konnte weiter die Realisierbarkeit der Automatisierung von Teilprozessen

nachgewiesen werden. Letztlich konnten Effizienzvorteile durch die Standardisierung des Vertragsprozesses und die Integration der Informationsflüsse in einer einzelnen Applikation aufgezeigt werden, die zu Kosteneinsparungen führen und die Arbeit der Mitarbeitenden erleichtern.

Im Kontext der betrachteten Brauerei wird die Umsetzung eines blockchain-basierten Vertragsmanagementsystems empfohlen, doch sind weitere Untersuchungen für die Transformation des Prototyps in ein produktives System vorzunehmen. Diese sollten eine Kostenanalyse und Machbarkeitsstudie der Erweiterbarkeit des Systems sowie eine Bestätigung der Rechtskräftigkeit von Verträgen in einer Blockchain beinhalten.

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

Abkürzungsverzeichnis

API	Application Programming Interface
B2B2C	Business-to-Business-to-Customer
CRM	Customer Relationship Management
CRUD	Create Read Update Delete
CSS	Cascading Style Sheets
DAO	Dezentralisierte Autonome Organisation
DLT	Distributed Ledger Technology
ERP	Enterprise Resource Planning
HTML	Hypertext Markup Language
HTTP	Hypertext Transfer Protocol
IDE	Integrated Development Environment
JDBC	Java Database Connectivity
JDK	Java Development Kit
JPA	Java Persistence API
JSON	JavaScript Object Notation
JVM	Java Virtual Machine
KMU	Kleine und mittlere Unternehmen
MDWE	Model Driven Web Engineering
MVP	Minimum Viable Product
NGO	Non-Governmental Organization
ORM	Object Relational Mapping
POJO	Plain Old Java Object
REST	Representational State Transfer
RPC	Remote Procedure Call
SQL	Structured Query Language

1 Einleitung

Diese Arbeit untersucht anhand einer praktischen Umsetzung eines Prototyps die Eignung und Reife der Blockchain-Technologie für das Rückvergütungs- und Darlehensvertragsmanagement einer Schweizer Brauerei. Beginnend mit einer Literaturrecherche und einer theoretischen Analyse des Business-Ökosystems der Brauerei werden potenzielle Einsatzmöglichkeiten der Technologie aufgezeigt sowie der Use Case des Vertragsmanagements hergeleitet.

In diesem Kapitel wird der Bezugsrahmen der Arbeit hergestellt und die im Vertragsmanagement bestehende Problematik ausgeführt. Das Kapitel soll ausserdem zur Ausarbeitung des Forschungsziels und der leitenden Forschungsfragen verwendet werden sowie zur Vorstellung des Praxispartners dienen.

1.1 Ausgangslage

Die zunehmende Globalisierung und die technologischen Fortschritte in Transport und Logistik führen zu komplexen Supply-Chain-Konstrukten (Abeyratne & Monfared, 2016). Darüber hinaus fördert der Anstieg an technologischer und fachlicher Kompetenz in Schwellenländern die Möglichkeiten für Unternehmen selbst komplexe Leistungen wie HR oder IT global zu beziehen (Kolmykova, 2016). Die damit wachsende Anzahl an Schnittstellen und Geschäftsbeziehungen erhöht den Koordinationsaufwand für Unternehmen und stellt das Supply-Chain-Management vor neue Herausforderungen (Grimm, Hofstetter & Sarkis, 2016; Kolmykova, 2016). Ausserdem gestaltet es sich immer schwieriger, die notwendige Transparenz innerhalb der Supply Chain herzustellen, wodurch die Hauptaufgabe des Supply-Chain-Managements, das Organisieren der Güter- und Informationsflüsse entlang der gesamten Wertschöpfungskette, erschwert wird (Abeyratne & Monfared, 2016; Zhu & Xu, 2012). Doch gerade in einem dynamischen, komplexen und stark konkurrierenden Umfeld ist es für Organisationen von bedeutender Wichtigkeit, diese Flüsse holistisch zu orchestrieren, Synergien zwischen den Teilnehmern zu erschliessen und die Ressourcen innerhalb der Supply Chain effizient zu nutzen (Korpela, Hallikas & Dahlberg, 2017).

1.2 Problemstellung

Einen noch weiteren Betrachtungsrahmen der Beziehungen einer Unternehmung wie das Supply-Chain-Management nimmt der Begriff «Business Ecosystem», zu Deutsch Business-Ökosystem, von Moore (1993) ein. Business-Ökosysteme umfassen nicht nur Akteure der Wertschöpfungskette, sondern jegliche Stakeholder einer Unternehmung (Yue, 2013). Dieser erweiterte Betrachtungsrahmen lässt die Anzahl an Beziehungen, Güter- und Informationsflüssen von Unternehmen weiter ansteigen (Mäntymäki, Salmela & Turunen, 2018). Mit der Dynamik in einem solchen System wird es für Unternehmen umso schwieriger, die Interoperabilität zwischen den Akteuren herzustellen, um kosteneffiziente und nutzenstiftende Leistungen für die Mitglieder des Systems zu erbringen (Korpela et al., 2017). Eine der Herausforderungen in Business-Ökosystemen ist die Gewährleistung eines gesicherten Informations-, Daten- und Dokumentaustauschs (Cojocaru, Sarraipa, Jardim-Golcalves & Stanescu, 2014). Eine besondere Bedeutung bekommt diese Aufgabe im Vertragsmanagement von Unternehmen.

Verträge sind laut Peukert (2018) ein Mittel zur rechtlichen Gestaltung persönlicher oder wirtschaftlicher Verhältnisse durch übereinstimmende Willenserklärung zweier oder mehrerer Parteien. Dabei bilden sie die Grundlage aller Geschäftsbeziehungen und Transaktionen eines Unternehmens mit anderen Parteien (Governatori et al., 2018; Liu & Partovi, 2007). Grundsätzlich werden Verträge abgeschlossen, da ein Fehlverhalten einer beteiligten Partei nicht ausgeschlossen werden kann (Dorschel & Manz, 2017). Ein Vertrag bezieht sich deshalb immer auf ein Misstrauensverhältnis (Dorschel & Manz, 2017). Durch den im wirtschaftlichen Bereich meist schriftlichen Vertrag kann die betroffene Partei vor rechtlichen Instanzen den von allen Parteien unterzeichneten Vertragsinhalt geltend machen, um den entstandenen Schaden zu mindern (Dorschel & Manz, 2017).

Das Verwalten von physischen und digitalen Verträgen (abgelegt als PDF in Datenbanken oder ERP-Systemen) ist eine aufwendige administrative Aufgabe, die hohe Kosten verursacht (Aitken, 2017). Doch neben betriebswirtschaftlichen Ineffizienzen bestehen für das Vertragsmanagement von Unternehmen weitere ungelöste Probleme. Verträge werden oft an verschiedenen Orten archiviert und verwaltet, sodass innerhalb einer Unternehmung Silostrukturen von Datenbanken oder physischen Ablagen entstehen (Aitken, 2017; Konfidio, 2018). Extern werden aufgrund des besagten Misstrauensverhältnisses verschiedene Versionen und Kopien eines Vertrags aufbewahrt, wodurch keine einzelne gültige Version des Vertrages besteht (Konfidio, 2018; Dorschel & Manz, 2017). Weiter

ist der Prozess der Entstehung und der Erfüllung des Vertrags meist intransparent (Boshell, 2018). So ist es für das Management schwierig, den aktuellen Status von Verträgen zu überwachen und in einem Schlichtungsfall sämtliche relevanten Daten für den Nachweis bereitzustellen (Boshell, 2018). Zuletzt sind die Entstehungsprozesse eines Vertrags oft langsam und nicht standardisiert (Accenture, o. J.). Sowohl Mailverkehr und Postversand während der Vertragsentstehung als auch der sequenzielle Signierungsprozess benötigen Zeit und verursachen weitere Kosten, die minimiert werden können (Boshell, 2018). Ausserdem gibt es mit physischen oder digitalen Vertragskopien keine Sicherheit, dass der verhandelte Vertragsinhalt ohne Kompromiss ausgeführt wird (Boshell, 2018).

Diese Herausforderungen im Vertragsmanagement beschäftigen auch die Feldschlösschen Getränke AG, die führende Brauerei und der grösste Getränkehändler der Schweiz. Als Tochterunternehmen der Carlsberg-Gruppe werden viele Transaktionen und Verträge zu Produktion und Sourcing über das zentrale Supply-Chain-Management von Carlsberg abgewickelt. Der Fokus in dieser Arbeit liegt deshalb auf anderen Stakeholdern von Feldschlösschen: den nachgelagerten Gastrobetrieben. Mit den Gastrobetrieben schliesst Feldschlösschen neben Kaufverträgen, die über das Customer-Relationship-Management (CRM) abgewickelt werden, Rückvergütungsverträge mit oder ohne Darlehensverträge ab. Dabei sieht Feldschlösschen die folgenden Punkte im Rückvergütungs- und Darlehensvertragsmanagement als problematisch an:

- hoher Personalaufwand durch die Verwaltung der Verträge
- eine nicht ausschliesslich digitale Ablage der unterzeichneten Verträge und die redundante Haltung von Kopien bei den Vertragspartnern
- nicht ausreichende Transparenz über laufende Verträge, zugehörige Dokumente und relevante Informationen
- Informationsverlust bei der Überschreibung des Vertrages auf einen neuen Kunden (Pächter), wenn der Vertrag auf der Lokalität abgeschlossen wurde
- nicht automatisierte und nicht standardisierte Entstehung- und Verhandlungsprozesse der Verträge, die dadurch eine hohe Durchlaufzeit aufweisen

1.3 Praktische Relevanz

Das Verwalten von Verträgen ist ein alltäglicher Prozess eines Unternehmens. Bis heute stehen für das Vertragsmanagement drei grundlegende Optionen zur Verfügung, die in der Praxis eine unterschiedliche Ausprägung und Zusammensetzung aufweisen können (Dorschel & Manz, 2017; Finnegan, Malone, Maranon & Guillen, 2007).

Die erste Methode ist das manuelle Vertragsmanagement, indem Verträge persönlich verhandelt, schriftlich erstellt und anschliessend signiert werden (Dorschel & Manz, 2017). Je nach Vertragsinhalt lassen sich die daraus resultierenden Prozesse und Workflows (Laufzeitprozesse) wie Bestellung, Lieferung und Bezahlung in die operativen Systeme (ERP, CRM) der Unternehmen eintragen (Uher & Davenport, 2009). Der schriftlich signierte Vertrag wird von einer Partei physisch aufbewahrt und jeweils eine Kopie davon von allen anderen beteiligten Parteien (Dorschel & Manz, 2017). Gerade in KMUs ist das manuelle Vertragsmanagement weit verbreitet (Dangl, 2018). Laut Dangl (2018) ist diese Vorgehensweise aber aufwendig, intransparent, fehleranfällig und befriedigt nicht die Anforderungen aus der Problemstellung.

Eine weitere Möglichkeit, einen Vertrag beglaubigen zu lassen und eine einzig gültige Version sicherzustellen, ist es, Notariatsdienste in Anspruch zu nehmen (Crosby, Pattanayak, Verma & Kalyanaraman, 2016). Notare stellen die Rahmenbedingungen für die Vertragsunterzeichnung her, schützen die Integrität und stellen die korrekte Ausführung des Vertrags über die Laufzeit sicher (Sury, 2016). Ein notarieller Service einer Drittpartei löst die Probleme der Integrität von Verträgen durch eine einzige beglaubigte Version, jedoch nicht die der langen Entstehungs- und nicht automatisierten Laufzeitprozesse. Ausserdem verursachen solche Dienste hohe Kosten (Sury, 2016).

Die letzte in dieser Arbeit betrachtete Option zur Vertragsverwaltung sind elektronische Verträge (e-contracts), welche von einem System verwaltet und ausgeführt werden (Krishna, Karlapalem & Dani, 2005). Die Anwendung von Vertragsmanagement-Software findet in Unternehmen eine immer häufigere Verwendung (von Waldegge, 2018). Mittels elektronischem Vertragsmanagement können Verträge digital erstellt, unterzeichnet, verwaltet und die Workflows digital abgebildet werden (von Waldegge, 2018; Chiu, Karlapalem, Li & Kafeza, 2002; Krishna et al., 2005). Für die Persistenz werden die Verträge und zugehörigen Daten in einer Datenbank oder in einem Cloudsystem gespeichert (Schrauzer, 2012, S. 85). Zudem können Verträge und Workflows sicher zwischen den Systemen der Parteien ausgetauscht und die meisten Prozesse im Vertragsmanagement

damit automatisiert werden (Wang, Grefen & Vonk, 2008; Sury, 2016; Merz, 2004). Einziges Problem bei dieser Lösung bleibt, dass entweder alle Parteien eine Vertragskopie in solch einem System verwalten müssen und somit eine nicht konsistente Redundanz entsteht oder die übrigen Parteien Zugriff auf das System der vertragshaltenden Partei erhalten und dabei ein Vertrauensverhältnis bestehen muss (Wang, Grefen & Vonk, 2008; Sury, 2016). Auch wäre möglich, auf eine stetig online vertraute Drittpartei (Trusted Third Party – TTP) zurückzugreifen, welche die Integrität des Vertrages wahrt, wobei jedoch dieselben Nachteile wie bei Notariatsdiensten auftreten. Die Problemstellung ist mit heutigen Systemen daher noch nicht vollumfänglich gelöst, weshalb ein möglicher Lösungsansatz in dieser Arbeit untersucht wird.

Laut Dorschel & Manz (2017) ist das Vertragsmanagement ein naheliegendes und in zahlreichen theoretischen Projekten untersuchtes Anwendungsszenario der Blockchain-Technologie. Namhafte Firmen und Start-ups wie Accenture, Inhubber, ChainThat und Konfidio arbeiten aktuell an generellen blockchain-basierten Lösungen im Vertragsmanagement, was die Aktualität sowie die Relevanz dieser Arbeit bekundet (Accenture, o. J.; Inhubber, o. J.; Konfidio, o. J.; ChainThat, o. J.). Aus der Literaturrecherche geht aber noch keine marktfähige Lösung in diesem Bereich hervor.

1.4 Forschungsziel

Im theoretischen Teil dieser Arbeit soll das Business-Ökosystem der Feldschlösschen Getränke AG betrachtet werden. Dabei werden Möglichkeiten evaluiert, wie mithilfe der Blockchain-Technologie Teile dieses Ökosystems digitalisiert werden können. Die Ergebnisse sollen Feldschlösschen helfen, die Blockchain-Technologie und ihre Anwendungsbereiche in einem Business-Ökosystem besser einordnen und bewerten zu können.

Für ein mögliches Einsatzgebiet der Blockchain-Technologie im Business-Ökosystem von Feldschlösschen, dem Rückvergütungs- und Darlehensvertragsmanagement, soll im praktischen Teil ein Prototyp erarbeitet und validiert werden. Dazu soll zuerst die passende Blockchain-Plattform eruiert, anschliessend der Prototyp nach einer ausgewählten Methodik konzipiert und programmiert sowie an den festgelegten Anforderungen getestet werden. Mittels Prototyp sollen das Potenzial, die aktuellen Möglichkeiten, Limitierungen und die Reife der Blockchain-Technologie für den Einsatz einfacher Business-Applikationen aufgezeigt und bewertet werden. Zudem soll die Arbeit Feldschlösschen als Grundlage zur Beurteilung eines möglichen blockchain-basierten Folgeprojekts dienen.

1.5 Forschungsfragen

Aus der in Kapitel 1.2 beschriebenen Problemstellung und den in Kapitel 1.4 festgelegten Zielen ergeben sich für diese Arbeit folgende Fragestellungen:

- Welche Blockchain-Plattform soll für das Rückvergütungs- und Darlehensvertragsmanagement von Feldschlösschen eingesetzt werden? Welche Möglichkeiten, Herausforderungen und Limitierungen entstehen durch diese Wahl?
- In welchem Mass ist es schon heute möglich, einzelne Elemente während der Vertragsentstehung und Vertragsverwaltung in einer Blockchain zu automatisieren? Welche Vorteile hat ein digital geführter Vertrag?
- Wie kann durch den Einsatz der Blockchain-Technologie die Transparenz der laufenden und vergangenen Verträge erhöht und dadurch eine Rückverfolgbarkeit auf Kundenebene ermöglicht werden?

1.6 Abgrenzung

Eine Implementation einer blockchain-basierten Business-Ökosystem-Plattform in einem schweizweit tätigen und global vernetzten Unternehmen ist sehr umfangreich und geht über den Rahmen dieser Arbeit hinaus, weshalb eine Eingrenzung vorgenommen werden muss. Dadurch soll der nötige Fokus innerhalb der Arbeit gewahrt und eine zielgerichtete Bearbeitung sichergestellt werden.

Business-Ökosystem: Der Business-Ökosystem-Ansatz wird mit dem Ziel eingeführt, daraus einen passenden Use Case für eine Blockchain-Anwendung ableiten zu können. Aus diesem Grund soll durch eine theoretische Einführung, ein für diese Arbeit geltendes gemeinsames Verständnis des Begriffs erreicht werden. Weiter sind Vorgehensmodelle vorzustellen, die zur Identifikation des Business-Ökosystems von Feldschlösschen verwendet werden sollen. Die detaillierte Aufarbeitung und kritische Würdigung des Begriffs sind nicht Bestandteil dieser Arbeit. Für die darauffolgende Umsetzung eines Prototyps ist nicht das gesamte Business-Ökosystem nach Moore (1993) zu betrachten, sondern ein abgesteckter Rahmen aus Stakeholdern, welcher in Abschnitt 4.3 näher vorgestellt wird. Dieser Rahmen soll für die gesamte Arbeit beibehalten werden, das heisst, für die Konzeption, Umsetzung und Validierung des Prototyps.

Blockchain: Bei der Blockchain-Technologie wird eine ähnliche Abgrenzung vorgenommen. Der theoretische Aspekt soll dahingehend erörtert werden, dass dem Leser das Verstehen dieser Arbeit ermöglicht wird. Ausführliche Behandlungen der konzeptionellen und technischen Grundlagen der Blockchain-Technologie sind jedoch nicht Teil dieser Arbeit. Vielmehr wird auf die technische Umsetzung, Möglichkeiten und Hürden gegenwärtig verfügbarer Technologien eingegangen und anhand des Prototyps validiert.

Prototyp: Die primäre Aufgabe des Prototyps ist die Erstellung eines Artefakts zur Validierung der unter Punkt 1.2 aufgeführten Problemstellungen und zur Beantwortung der in Punkt 1.5 gestellten Forschungsfragen. Es handelt sich mehr um ein technologisches Proof-of-Concept zur Vertragsverwaltung von Feldschlösschen, als um eine Untersuchung einer vollends ausgestalteten Applikation. Damit soll der Fokus auf die momentane technische Umsetzbarkeit der Blockchain-Technologie im Vertragsmanagement gelegt werden, anstatt auf die Überprüfung der Benutzerführung (UX-Design) oder die Anbindung an die Umsysteme. Eine Funktionalität ist immer soweit auszugestalten, damit eine Validierung dieser oder einer Problemstellung möglich ist. Die Entwicklung des Prototyps findet in einem isolierten Raum statt. Schnittstellen zu benötigten operativen Dritt-Systemen sollen wenn nötig nachgebildet werden.

1.7 Vorstellung Praxispartner

Die Feldschlösschen Getränke AG, mit dem Hauptsitz in Rheinfelden (AG), ist die führende Brauerei und der grösste Getränkehändler der Schweiz. Feldschlösschen produziert nationale Biere unter gleichem Namen, Mineralwasser wie Rhäzünser und Arkina und Softgetränke der Marke Queen's. Zusätzlich braut Feldschlösschen Biere für Tochterunternehmen wie Cardinal, Hürlimann, Warteck und Valaisanne und ist Besitzerin der Schweizer Lizenz für Schweppes.

Seit dem Jahr 2000 gehört Feldschlösschen zur dänischen Carlsberg-Gruppe und übernimmt den Vertrieb internationaler Biermarken (Grimbergen, Corona, Brooklyn etc.) und Weine in der Schweiz. Durch die Übernahme entstand eine Abhängigkeit zum Mutterkonzern und Feldschlösschen ist heute an die Ziele und Strategien der Carlsberg-Gruppe gebunden. Ausserdem werden verschiedene Funktionen und Leistungen wie Einkauf, Sourcing und IT zentral von Carlsberg gesteuert. Dies engt einerseits den Handlungsspielraum und die transparente Sicht über die Supply Chain von Feldschlösschen ein,

ermöglicht andererseits jedoch die Bildung von Synergien und Economy-of-Scale für die Gruppe.

Diese Arbeit wird mit der Unterstützung von Manfred Weiss (Sales Service Director) und Samuel Schorno (Leiter Sales Support) durgeführt. Manfred Weiss ist der Initiator der Zusammenarbeit und unter anderem Verantwortlicher für die Rückvergütungs- und Darlehensgeschäfte bei Feldschlösschen. Über Samuel Schorno findet der direkte Kontakt und Informationsaustausch für die Ausarbeitung des Prototyps in dieser Arbeit statt. Angesiedelt ist das Projekt bei Feldschlösschen im On-Trade, einer der beiden Verkaufsabteilungen von Feldschlösschen. Das On-Trade umfasst Gastronomiekunden, B2C-Verkäufe und Rückvergütungs- sowie Darlehensgeschäfte. Ausserdem werden im On-Trade digitale Plattformen für Feldschlösschen und die gesamte Carlsberg-Gruppe entwickelt.

2 Theoretische Einführung

In der theoretischen Einführung sollen die benötigten Konzepte und Technologien für diese Arbeiten erklärt werden. Da es sich in erster Linie um eine praktische Arbeit handelt, werden jedoch nur jene Elemente theoretisch aufgearbeitet, die später auch für die Umsetzung wichtig sind. Eine detaillierte theoretische Erklärung der Technologie und der betriebswirtschaftlichen Konzepte ist nicht Teil dieser Arbeit.

2.1 Business-Ökosystem

Business-Ökosysteme versprechen sowohl neue Möglichkeiten in der Zusammenarbeit, dem Informationsaustausch und der Wertgenerierung zwischen Stakeholdern als auch zur Erreichung nachhaltiger und langfristiger Wettbewerbsvorteile für Unternehmen (Razavi, Krause & Strømmen-Bakhtiar, 2010). In diesem Kapitel sollen die Grundlagen des Business-Ökosystems anhand von Definitionen erklärt werden. Überdies wird während der Einführung auf ausgewählte wichtige Merkmale eingegangen.

2.1.1 Definition und Einführung

Erstmals wurde der Begriff des Business-Ökosystems von James Moore (1993) eingeführt. Zu Beginn definierte er das Business-Ökosystem als komplexe wirtschaftliche Community, welche in einer sich gegenseitig unterstützenden Wechselwirkung mit anderen Organisationen und Individuen steht (Moore, 1993). Später fügte Moore (1996) hinzu, dass ein Business-Ökosystem Güter und Services mit Nutzen für Kunden herstellt, die selbst Mitglieder dieses Ökosystems sind. Das Business-Ökosystem ist durch eine große Anzahl von lose gekoppelten Teilnehmern gekennzeichnet, die für ihre gegenseitige Wirksamkeit und ihr Überleben voneinander abhängig sind (Moore, 1996). Lose bedeutet, dass es für Teilnehmer einfach sein muss, dem System beizutreten und einen Beitrag zu leisten, sie dieses aber auch wieder verlassen können, wodurch eine hohe Dynamik entsteht (Razavi et al., 2010). Zu den Akteuren eines Business-Ökosystems gehören sämtliche Stakeholder einer Unternehmung wie Kunden und Lieferanten aller Stufen (Tiers), Finanzgesellschaften, Versicherungen, staatliche und nichtstaatliche Organisationen, Gewerkschaften, Universitäten etc. (Yue, 2013). Dabei werden nicht nur momentane, sondern auch zukünftige Partner fokussiert (Yue, 2013). Ähnlich definieren Thomas und Autio (2014) das Business-Ökosystem als organisationales Feld, das alle Beteiligten umfasst, welche um eine kollektive Wertschöpfung bemüht sind. Aus der Literaturrecherche

von Mäntymäki et al. (2018) und Lavikka, Hirvensalo, Smeds und Jaatinen (2017) resultiert, dass Business-Ökosysteme mindestens folgende Merkmale aufweisen:

- Die Mitglieder in einem System befinden sich in einer komplexen Wechselwirkung. Diese zeigt sich durch einen Effekt auf das Gesamtsystem, infolge eines Erfolgs oder Misserfolgs eines einzelnen Mitglieds (diskriminierend).
- Jedes Mitglied bringt einen gewissen Input in das System. Durch die Gesamtheit an Interaktionen im System entsteht Wert für den Kunden und das System entwickelt sich weiter (Analogie zur biologischen Entwicklung).
- In einem Business-Ökosystem muss eine Governance-Struktur vorhanden sein. Laut Mäntymäki et al. (2018) stellt diese oft eine Schlüsselfigur (Keystone) in Form des wichtigsten Akteurs im Ökosystem dar, welche eine grobe Orchestrierung der Aktivitäten im System übernimmt. Um diesen Hub herum nehmen spezialisierte Unternehmen ihre Rolle (Nische) im Ökosystem ein (Razavi et al., 2010). Für Lavikka et al. (2017) muss die Governance hingegen nicht ein einzelner Akteur darstellen, sondern kann aus der Gesamtheit einer autoritären Struktur (mehrere starke Teilnehmer) bestehen. Die Governance-Struktur ist zuständig für die Entscheidungsfindung, Konfliktlösung und Mitgliederkontrolle, ausserdem wahrt sie die Offenheit des Systems und organisiert die wesentlichen Aktivitäten.
- Eine gemeinsame Logik und Richtung verbindet die Teilnehmer des Ökosystems, wodurch eine gegenseitige Legitimität, ein gewisses Vertrauen und Anerkennung zwischen den Mitgliedern des Systems entsteht (Lavikka et al. 2017).

Es stellt sich allerdings die Frage, wie ein solch komplexes System von einem einzelnen Schlüsselakteur organisiert werden kann. Dies ist laut Mäntymäki et al. (2018) immer noch Gegenstand laufender Untersuchungen. Eine mögliche Lösung ist der von Beer (1984) entwickelte Ansatz der Selbstregulation, indem die notwendigen Rahmenbedingungen geschaffen werden, sodass ein System «überlebensfähig» wird. Dies geht aber über die theoretische Grundlage in dieser Arbeit hinaus.

2.1.2 Abgrenzung zum Supply-Chain-Management

Supply Chains beschreiben die Informations- und Güterflüsse des gesamten Wertschöpfungsprozesses bis zum Endkunden (Slack, Brandon-Jones & Johnston, 2016). Das Supply-Chain-Management beschreibt somit die aktive Gestaltung aller Prozesse, um Kunden oder Märkte wirtschaftlich mit Produkten, Gütern und Dienstleistungen zu versorgen (Voigt, 2018). Im weiteren Sinne gehören zum Supply-Chain-Management auch alle begleitenden Auftragsabwicklungs- und Geldflüsse (Voigt, 2018).

Die eindeutigsten Differenzierungsmerkmale zwischen Supply Chains und Business-Ökosystemen sind erstens der weitere Rahmen an Stakeholdern, welche der Business-Ökosystem-Ansatz einnimmt und dadurch neue Akteure in die Betrachtung der Unternehmen bringt (Rong, Hu, Lin, Shi & Guo, 2015). Zweitens ist das Business-Ökosystem ein entwickelndes, dynamisches Konstrukt, das auf einer mehrseitigen, wechselwirkenden Systemansicht basiert (bidirektional) und zudem einen Lebenszyklus besitzt (Baghbadorani & Harandi, 2012; Rong et al., 2015). Der stärkste Treiber solcher Ökosysteme stellt die Digitalisierung dar (Rong et al., 2015). Grundlage für Business-Ökosysteme sind Plattformen, auf denen der Informationsaustausch stattfindet (Mäntymäki et al., 2018). Über die Plattform soll der einfache Zugang, der Beitrag der Teilnehmer und deren Orchestrierung stattfinden (Mäntymäki et al., 2018). Die Plattform wird meist durch den Keystone des Ökosystems betrieben, allerdings von allen Teilnehmern gefördert und weiterentwickelt (Mäntymäki et al., 2018). Auch Supply Chains verwenden Supply-Chain-Management-Systeme, doch nehmen diese im Vergleich zu einem Buisness-Ökosystem, in welchem das gesamte Geschäftsmodell auf der Plattform basiert, eine weniger zentrale Rolle ein. Jeder Teilnehmer ist um die «Gesundheit» des Ökosystems besorgt, weshalb die Teilnehmer gemeinsam in die Plattform investieren und einander in ihrer Auftragserfüllung unterstützen (Razavi et al. 2010). Von den Teilnehmern ist das Ökosystem daher immer aus einer holistischen Sicht zu betrachten (Razavi et al., 2010).

2.2 Blockchain

Nach der Einführung von Bitcoin, der ersten relevanten Implementation der Blockchain-Technologie durch Satoshi Nakamoto (2008), entstand um die Möglichkeiten der Technologie ein regelrechter Hype. Analysten prognostizierten, dass durch die Blockchain sowohl Unternehmen als auch die Wirtschaft neu definiert würden (Iansiti & Lakhani, 2017). Inwiefern die Blockchain als disruptive Technologie angesehen werden kann, ist man sich in der Literatur aber noch uneinig (Bettín-Díaz, Rojas & Mejía-Moncayo, 2018; Iansiti & Lakhani, 2017). Iansiti und Lakhani (2017) sehen die Blockchain eher als fundamentale mit TCP/IP vergleichbare Technologie, auf der zukünftige wirtschaftliche und soziale Systeme aufgebaut werden können. Die Transformation zur Blockchain werde deshalb nicht abrupt und hart, sondern eher sanft und linear vonstattengehen, auch wenn die schlussendlichen Auswirkungen gross sein können (Iansiti & Lakhani, 2017). Deshalb sei es für Unternehmen wichtig, sich einerseits mit der Technologie auseinanderzusetzen, um Erfahrungen zu sammeln, und andererseits erste Versuche mit der Blockchain zu wagen, indem initiale Prototypen und Applikationen umgesetzt werden, um die Möglichkeiten und Einsatzgebiete der Technologie erkennen zu können (Iansiti & Lakhani, 2017). Laut Iansiti und Lakhani (2017) kann es für Unternehmen ausserdem von Vorteil sein, sich bereits jetzt das notwendige Know-how anzueignen, damit der Einstieg in die Technologie nicht verpasst wird und um sich «First-Mover-Advantages» gegenüber der Konkurrenz zu verschaffen.

Ziel dieses Kapitels ist es nun, die für diese Arbeit benötigten Konzepte der Blockchain-Technologie aufzuzeigen und ein gemeinsames Verständnis zu schaffen, welches das Lesen dieser Arbeit erleichtern soll.

2.2.1 Einführung in die Blockchain-Technologie

Grundsätzlich ist die Blockchain eine dezentralisierte Datenbank (Almeida, Albuquerque & Silva, 2019). Dezentralisiert heisst, dass die Daten nicht an einer zentralen Stelle gespeichert werden und eine einzelne Entität die alleinige Datenhoheit besitzt, sondern dass die Daten über ein Peer-to-Peer-Netzwerk an alle Entitäten (Nodes) verteilt werden (Nguyen & Dang, 2018). Im Zusammenhang mit Blockchain wird jedoch oft von Konten (engl. Ledger) anstatt von Datenbanken gesprochen (Meier, 2018). Die Benennung ist auf die transaktionsorientierte Speicherung der Daten, wie bei der initialen Verwendung der Blockchain in Bitcoin, zurückzuführen (Meier, 2018).

In einer Blockchain werden die Transaktion zur Persistenz einem Block zugewiesen, der über kryptographische Hashes mit dem jeweils vor- und nachgelagerten Block verbunden wird, sodass eine Kettenstruktur entsteht (Kwak, Kong, Cho, Phuong & Gim, 2019). Die Blocks (mit Transaktionen) werden von anderen Teilnehmern im Netzwerk verifiziert, weshalb die Verifikation nicht von einer vertrauenswürdigen Drittpartei (z. B. einer Bank) durchgeführt werden muss (Kwak et al., 2019). Somit reguliert und überwacht sich das Netzwerk selbst und keine zentrale Entität, welche die Kontrolle oder Governance des Netzwerkes übernimmt, wird benötigt (Almeida et al., 2019; Abeyratne & Monfared, 2016). Transaktionen können einen Übertrag von monetären Geldbeträgen oder jedem beliebigen materiellen oder immateriellen Vermögenswert sein (Wüst & Gervais, 2018). Unter welchen Konditionen diese Transaktionen stattfinden, kann je nach Plattform durch einen automatisch ausführbaren Programmcode festgelegt werden (Luu, Chu, Olickel, Saxena & Hobor, 2016). Solche Programme, die von jedem Netzwerkteilnehmer in die Blockchain geschrieben werden können, sind meist unter dem Namen Smart Contracts bekannt (Luu et al., 2016). Mithilfe von Smart Contracts könnten ganze dezentralisierte und autonome Organisationen (DAOs) entstehen, die aufgrund der festgelegten Konditionen automatisch Transaktionen vornehmen (Luu et al., 2016). Gleich wie bei Transaktionen können in der Blockchain gespeicherte Smart Contracts nicht mehr verändert werden (Luu et al., 2016). Ein Beispiel für ein Smart Contract ist, dass einem Kind beim Erreichen des 18. Lebensjahrs automatisch CHF 1000.- überwiesen werden (Swan, 2015). Solche Smart Contracts werden aber nicht von jeder Blockchain-Plattform unterstützt und auch deren Aufbau und Funktionsweise ist anbieterabhängig (Swan, 2015).

Um nun ein einheitliches Verständnis der Blockchain-Technologie zu schaffen, soll für diese Arbeit die folgende Definition von Seebacher & Schüritz (2017) gelten: Eine Blockchain ist eine verteilte Datenbank, die zwischen einem Peer-to-Peer-Netzwerk geteilt und vereinbart wird. Die Blockchain besteht aus einer verknüpften Folge von Blöcken, die zeitgestempelte Transaktionen enthalten, durch Public-Key-Kryptographie gesichert und von der Netzwerk-Community überprüft werden (Seebacher & Schüritz, 2017). Sobald ein Element an die Blockchain angehängt wurde, kann es nicht mehr geändert werden, wodurch eine Blockchain zu einem unveränderlichen Datensatz vergangener Aktivitäten wird (Seebacher & Schüritz, 2017).

Zusammenfassend können aus der Einleitung und der Definition die nachfolgend aufgelisteten Schlüsselelemente entnommen werden (Abeyratne & Monfared, 2016; Seebacher & Schüritz, 2017).

- **Dezentralisierte Datenbank:** Daten werden nicht an einem zentralen Ort gespeichert, sondern über die Entitäten verteilt (konsistente, redundante Datenhaltung).
- **Peer-to-Peer-Übermittlung:** Die Kommunikation (z.B. eine Transaktion) geschieht direkt zwischen zwei Entitäten, anstatt über eine zentrale Instanz.
- **Transparenz:** Alle Transaktionen und die jeweils zugehörigen Werte sind für alle Teilnehmer des Netzwerks sichtbar (nur in öffentlichen Blockchains).
- **Unveränderbarkeit:** Ist eine Transaktion einmal in die Blockchain eingetragen, kann sie nicht mehr verändert werden.
- **Automation (programmierte Logik):** Transaktionen können an bestimmte, von Teilnehmern programmierte Konditionen gebunden werden, die bei der Erfüllung automatisch von der Blockchain ausgeführt werden.

2.2.2 Öffentliche, private und Konsortium-Blockchains

Die grösste Differenzierung innerhalb der Blockchain-Technologien wird zwischen öffentlichen (permissionless), privaten und Konsortium- (permissioned) Blockchains vorgenommen (Kwak et al., 2019). Mit Blockchain-Anwendungen befinden sich Unternehmen in einem Zielkonflikt zwischen Datenschutz (Privacy) und Transparenz (Wüst & Gervais, 2018). In einer vollkommen transparenten Blockchain ist es jedem Teilnehmer erlaubt, sämtliche Information in der Blockchain zu lesen (Wüst & Gervais, 2018). Mit einer solch hohen Transparenz existiert jedoch keine Privacy und umgekehrt. Öffentliche Blockchains verfolgen das Konzept der Transparenz und somit sind alle Transaktionen für die Teilnehmer sichtbar (Zheng, Xie, Dai, Chen & Wang, 2018). Jeder Teilnehmer hat ausserdem zumindest theoretisch die Möglichkeit am Übereinstimmungsprozess (Konsensus), der im System stattfindenden Transaktionen teilzunehmen (Zheng et al, 2018). Transaktionen zu validieren bedeutet Aufwand, weshalb in öffentlichen Blockchains Anreizsysteme geschaffen werden müssen, damit Entitäten (in Bitcoin Miner) diese Aufgabe erledigen (Kwak et al., 2019). Dadurch entstehen erhebliche Transaktionsgebühren, und je nach Zeitaufwand der Validierung wird die Skalierbarkeit der Applikation eingeschränkt (Kwak, et al., 2019). Bei einer privaten Blockchain hingegen entscheidet eine zentrale Instanz, wer im Blockchain-Netzwerk teilnehmen darf und führt

den Übereinstimmungsprozess durch (Xu et al., 2016). Das heisst, nur ausgewählte Teilnehmer erhalten Lese- und Schreibrechte für gewisse Informationen im Netzwerk, sodass die Privacy gewahrt werden kann (Zheng et al., 2018). Eine spezifische Zuweisung von Rechten bedeutet gleichzeitig, dass die Teilnehmer der verwaltenden Instanz bekannt sein müssen, was eine Limitierung der Teilnehmeranzahl bedeuten kann (Zheng et al., 2018). Die letzte Ausprägung von Blockchains, die Konsortium-Blockchain, ist ein Kompromiss. In einer Konsortium Blockchain wird der Übereinstimmungsprozess von einer ausgewählten Gruppe an Entitäten (Nodes) durchgeführt, die von den restlichen Teilnehmern als vertrauenswürdig angesehen werden (Zheng et al., 2018). Dies bietet den Vorteil gegenüber einer privaten Blockchain, dass die Entscheidungsgewalt nicht bei einer einzelnen Entität liegt. Einen Überblick der Eigenschaften der verschiedenen Blockchain-Typen bietet die Tabelle von Zheng et al. (2018), gezeigt in Tabelle 1.

Zwischen den Typen kann jedoch nicht von besser oder schlechter gesprochen werden (Massessi, 2018). Es gibt nur einen geeigneten oder weniger geeigneten Typ für einen gewissen Use Case und oft werden hybride Modelle zwischen den Stereotypen verwendet (Massessi, 2018).

2.3 Blockchain im Business-Ökosystem

Die Kombination aus Blockchain-Technologie und Business-Ökosystem stellt laut Heumüller und Richter (2018) einen vielversprechenden Ansatz dar. Jedoch befindet sich die Blockchain-Technologie zum einen immer noch in einem frühen Entwicklungsstadium und zum anderen findet der Business-Ökosystem-Ansatz in der Wirtschaft noch keine breite Adaption und beschränkt sich auf wenige bekannte Beispiele wie Apple, Google oder Amazon (Mäntymäki, Salmela & Turunen, 2018; Abeyratne & Monfared, 2016). Wird sich auf Plattformen beschränkt, die nicht alle nach Moore (1996) benannten Stakeholder miteinbeziehen, so existieren bereits mehrere blockchain-basierte Anwendungsfälle oder befinden sich in der Entwicklung (Heumüller & Richter 2018).

Eine Vereinigung von Grossbanken namens We.trade, darunter auch die UBS, hat eine Plattform für den internationalen Handel entwickelt (Ade, 2018). Ziel ist es, internationale Transaktionen zu vereinfachen, transparenter zu gestalten und die langwierigen, papierintensiven Prozesse zu digitalisieren (Ade, 2018). Die Plattform soll für alle Interessenten geöffnet werden (Ade, 2018). Weitere Banken können sich anschliessen,

Benutzer (Transportunternehmen, Zwischenhändler, Endkonsumenten) sollen möglichst einfach eingebunden und auch Konkurrenzprojekte sollen integriert werden (Ade, 2018).

Im Gegensatz zu der grundlegenden Handelsplattform des vorherigen Beispiels, spezialisierten sich andere Start-ups stärker auf einzelne Plattformnischen (Fischer, Fiedler & Babenko, 2019). Loyyal bietet eine blockchain-basierte Plattform für Loyalitätspunkte an (Fischer et al., 2019). Das bisherig grösste Einsatzgebiet der Plattform ist der Flughafen Dubai, auf dem Kunden und Geschäfte der Plattform beitreten können (Fischer et al., 2019). Endkonsumenten mit gesammelten Loyalitätspunkten können diese in anderen Geschäften einlösen oder auf einem Marktplatz mit anderen Konsumenten handeln (Fischer et al., 2019). Das Geschäft von Loyyal beschränkt sich aber nicht nur auf B2C, sondern bietet auch eine Lösung für den B2B2C-Markt (Fischer et al., 2019).

Ein weiteres Beispiel ist die Plattform des Start-ups GUTS. GUTS entwickelt eine Plattform für den Ticketverkauf, mit dem Ziel, den nachfolgenden Schwarzmarkt für die verkauften Tickets zu verhindern (GUTS, o. J.). Dazu integriert die Plattform Artisten, Ticket-Konsumenten, Organisatoren und Veranstalter (GUTS, o. J.). Mithilfe der Blockchain-Technologie und Smart Contracts wird der Verkaufsweg des Tickets überwacht, der ursprüngliche Verkaufspreis beibehalten, automatische Transaktionen an die Mitglieder der Plattform ermöglicht und die Integrität der Tickets gewahrt (GUTS, o. J.).

Das letzte vorgestellte Beispiel für die Umsetzung eines blockchain-gestützten Business-Ökosystems ist das in Kooperation von Industrie und Hochschulen entwickelte Projekt Car Dossier (Zavolokina, Spychiger, Tessone & Schwabe, 2018). Ziel des Car-Dossier-Projekts ist ein transparentes, digitales Dossier über die gesamte Laufzeit eines Automobils zu führen (Zavolokina et al., 2018). Ein solches Dossier beinhaltet Aufzeichnungen über Schäden, Wartungen, Kilometerstände und weitere wichtige Attribute eines Fahrzeugs, die zur Bewertung des Zustands notwendig sind (Zavolokina et al., 2018). Durch die sichere Aufzeichnung mittels Blockchain will man dem Misstrauen im Automobilmarkt entgegenwirken und bindet dazu alle Stakeholder ein, die über den Lebenszyklus mit dem Fahrzeug in Berührung kommen (Zavolokina et al., 2018). Darunter befinden sich Hersteller, Importeure, Händler, Garagen, Versicherungen und auch die privaten Käufer und Verkäufer (Zavolokina et al., 2018). Im Projekt wird dabei die gemeinsame Wertgenerierung für die Mitglieder des Ökosystems fokussiert (Zavolokina et al., 2018).

Die vorgestellten Praxisbeispiele zeigen, dass Einsatzgebiete der Blockchain-Technologie in einem Business-Ökosystem existieren und sowohl Unternehmen als auch Forschungsinstitute gewillt sind, marktfähige Systeme zu entwickeln. Vorteile gegenüber vorhandenen Technologien bestehen immer dann, wenn ein Misstrauensverhältnis zwischen den Parteien besteht, die Transparenz im System erhöht werden soll, Daten zu Vermögensgegenständen unverändert bleiben und zurückverfolgt werden müssen oder Transaktionen aufgrund von Bedingungen automatisiert werden sollen. Zusätzlich wird durch die Dezentralität der Plattform, der Gedanke der gemeinsamen Wertschöpfung und kooperativen Weiterentwicklung der Plattform gestärkt, was anhand des Beispiels des Car Dossiers eindrücklich gezeigt wird. Nachteile hingegen ergeben sich aus derzeitigen technischen Limitierungen der Technologie, wie den hohen Transaktionskosten bei öffentlichen Blockchains, die tiefere Transaktionsrate gegenüber bisherigen Systemen oder das geringe Volumen der möglichen dezentralisierten Datenspeicherung. Auch erwähnen Heumüller und Richter (2018) das Problem, dass es schwierig sei, neue Stakeholder zu integrieren, weil oft das Verständnis und Vertrauen in die Technologie fehlen. Wichtig sei deshalb, dass bei der Evaluation des Business-Ökosystems und der Gestaltung der Plattform alle Teilnehmer identifiziert und in die Planung miteinbezogen werden (Heumüller & Richter, 2018). Nur wenn alle Ziele, Motivationen und Nutzen der Teilnehmer bekannt sind, kann eine gemeinsame Lösung umgesetzt werden, bei der für jedes Mitglied Wert entsteht (Heumüller & Richter, 2018).

3 Methodik

Nach Balzert, Schröder und Schäfer (2011, S. 71) unterliegt diese Arbeit dem Typ der konstruktiven Arbeiten. Eine konstruktive Arbeit hat zum Ziel, durch ein systematisches Vorgehen, ein (technisches) Erzeugnis zu erstellen, das selbst gestellten oder vorgegeben Anforderungen genügt (Balzert et al., 2011, S. 77). Aus dieser Typisierung resultiert laut Balzert et al. (2017) eine wissenschaftsbezogene Methodik, die folgend vorgestellt wird.

In der Wirtschaftsinformatik und auch in der Softwareentwicklung existieren zahlreiche Vorgehensmodelle für eine strukturierte, ergebnisorientierte Zielerreichung, die entweder sequenzieller oder iterativer Natur sind (Sandhaus, Berg & Knott, 2014, S. 27). Schwierigkeiten für die Wahl eines einzelnen standardisierten Vorgehensmodells in dieser Arbeit entstehen dadurch, dass zum einen nicht nur eine theoretische Analyse, sondern auch ein Entwicklungsprozess stattfinden soll und zum anderen das Projektteam aus einer einzelnen Person besteht. Aus diesen Gründen wird für diese Arbeit ein hybrides Modell der Softwareentwicklung angewandt (Sandhaus, Berg & Knott, 2015, S. 307). Der Kerngedanke von hybriden Modellen ist die Kombination von klassischen Phasenmodellen mit agilen Methoden in der Umsetzung (Sandhaus et al., 2015, S. 307).

Grundsätzlich wird die Arbeit in Analyse-, Planungs-, Implementation- und Finalisierungsphase unterteilt. Somit ist nach Habermann (2013) das dominierende Grundmodell sequenziell aufgebaut. Für die arbeitsbezogene Adaption der Methodik sollen nun eine gezielte Vorgehensweise erarbeitet und die konzeptionellen Grundlagen dazu geschaffen werden, wie die Supply Chain von Feldschlösschen zu einem Business-Ökosystem transformiert werden kann.

In der Literatur existieren verschiedenste Methoden und Prozesse, wie die Transformation einer Supply Chain in ein Business-Ökosystem anzugehen ist (Kandiah & Gossain, 1998; Yue, 2013; Mäntymäki et al., 2018; Moore, 1993). In dieser Arbeit wurde sich für eine Kombination aus den Prozessen von Yue (2013) und Lavikka et al. (2017) entschieden. Aufgrund des Rahmens der Arbeit wird der Prozess jedoch vereinfacht und situativ angepasst. Der daraus entstandene Prozess wird in Abbildung 1 gezeigt. Im Gegensatz zum Originalprozess nach Yue (2013) und Lavikka et al. (2017) folgen nach Prozessschritt zwei nicht die Evaluation, Planung und Umsetzung einer Plattform für sämtliche Stakeholder, sondern nur für die, des abgesteckten Kontexts des Use Cases.

Analyse: In der Analysephase wird die theoretische Grundlage durch eine fundierte Literaturrecherche deduktiv erarbeitet. Anhand des gewonnenen theoretischen Basiswissens soll das Business-Ökosystem der Feldschlösschen Getränke AG analysiert, daraus ein Use Case für den Prototyp abgeleitet und die adäquate Blockchain-Plattform zur Lösung der Problemstellung ausgewählt werden. Die eruierte Blockchain-Plattform soll den anschliessend zu definierenden Anforderungen als Grundlage dienen. Weiter sind in der Analysephase die Ist-Prozesse des Rückvergütungs- und Darlehensvertragsmanagement von Feldschlösschen zu ermitteln.

Bei der Identifikation der Teilnehmer und ihrer Rollen soll nach dem Modell von Yue (2013) vorgegangen werden. Um die Komplexität eines Business-Ökosystems zu vereinfachen und bei der Identifikation systematisch vorzugehen, teilt Yue (2013) das System in unterschiedliche Ringe ein, die in Abbildung 2 gezeigt werden.

Der «Core Business Circle» stellt den innersten Ring dar. Dieser enthält den Keystone, seine direkten Kunden und Lieferanten. In der zweiten Ebene des «Extended Business Circle» befinden sich hauptsächlich Kunden sowie Lieferanten aller Stufen (Tiers). Die ersten beiden Ringe des Modells sind somit äquivalent zur Supply Chain des Unternehmens. Der dritte Ring, der «Support and Maintenance Circle», umfasst alle staatlichen Organisationen und unterstützenden Stakeholder, wie Finanzinstitute, Versicherungen, Beratungsunternehmen, Agenturen, Investoren und Eigentümer, Gewerkschaften, Zertifizierungs- und Nominierungsstellen etc. Alle Organisationen bis zum dritten Ring tragen direkt oder indirekt zur Wertschöpfung innerhalb des Systems bei. Im äussersten Ring, dem «External Environment Circle», befinden sich alle übrigen Anspruchsgruppen aus dem politischen, ökonomischen, ökologischen, sozialen und technologischen Umfeld. Zu diesen Stakeholdern zählen unter anderem NGOs.

Ein weiteres Modell, welches für die Identifikation der Beziehungen zwischen den Teilnehmern benötigt wird, ist die Matrix von Matthes, Faber & Hernandez-Mendez (o. J.). Diese Matrix erlaubt es, die Beziehungen zwischen den Teilnehmern übersichtlich darzustellen, indem alle Stakeholder sowohl auf der X- als auch auf der Y-Achse abgebildet werden. Falls vorhanden, wird nun jedem X-Y-Schnittpunkt (ausser in der Diagonalen) einer der folgenden Beziehungs-Typen zugewiesen: befindet sich im Informationsaustausch mit; ist in einer Partnerschaft mit; ist Kunde von; ist Dienstleister von; ist Zulieferer von; ist interessiert an; ist Anteilseigner / Investor von. Durch die Zuweisung der

Beziehungen und das Einteilen der Stakeholder in Gruppen wird das Verwalten der Stakeholder vereinfacht (Matthes et al., o. J.). Diese Vorgehensweise kann mit einer Segmentierung im Marketing verglichen werden, anhand welcher die Profile der Plattform ausgerichtet werden können (Matthes et al., o. J.).

Konzeption / Planung: In der Konzeptions- und Planungsphase werden zu Beginn die Anforderungen an den Prototyp definiert. Dazu wird der von Hull, Jackson und Dick (2011, S. 25) beschriebene Requirements-Engineering-Prozess angewendet. Über das Requirements Engineering soll definiert werden, *was* das System erfüllen soll (Hull et al., 2011, S. 6). Diese Definitionen sollen in einer engen Zusammenarbeit mit Feldschlösschen qualitativ ausgearbeitet werden. Weiter verlangt der Prozess des Requirements Engineering, dass die definierten Anforderungen gemanagt (Traceability) und kommuniziert werden (Hull et al., 2011, S. 13). Schlussendlich werden die Anforderungen an den Prototyp für die Abnahme in der Finalisierungsphase benötigt.

Aufbauend auf den Anforderungen soll in einem frühen Schritt eine Visualisierung in einem iterativen Prozess erstellt werden. Diese Methode lehnt sich an die Vorgehensweise des Rapid-Prototyping oder des Minimum Viable Products (MVP) an und differenziert sich daher vom klassischen modellgetriebenen Web Engineering (MDWE) (Rivero et al., 2012, S. 152). Mock-ups sind ein wesentlicher Bestandteil dieser Vorgehensweise (Koren & Klamma, 2018, S.103). Damit sollen die Interaktionen mit den Stakeholdern gestärkt werden, mit dem Ziel, laufend Feedback zu erhalten und das Risiko für aufwendige Änderungen in einem späten Stadium des Projekts zu minimieren (Falzone & Bernaschina, 2018, S. 496). Mock-ups bilden ausserdem eine Kommunikationsgrundlage zur Überprüfung der definierten Anforderungen und fördern das Engagement des Projektteams durch die Veranschaulichung eines konkreten Ziels (Rivero et al., 2010, S. 40).

Implementation: Die Implementationsphase soll iterativ aufgebaut werden. Jedoch ist dies durch die Verfügbarkeit von lediglich einem einzigen Projektmitarbeiter nur teilweise möglich und agile Methoden wie Scrum oder Extreme Programming erübrigen sich (Sandhaus et al., 2014). Der iterative Aspekt soll dahingehend erreicht werden, dass die Anwendung in fertige Arbeitspakete zerlegt wird und diese bei ihrer Fertigstellung mit dem Praxispartner überprüft werden (Sandhaus et al., 2015). Für die Programmierung werden die übergeordneten Arbeitspakete weiter zerlegt und nach dem Kanban-Prinzip abgearbeitet (Al-Tahat, Dalalah & Barghash, 2012).

Finalisierung: In der letzten Phase wird die theoretische und praktische Arbeit vereinigt und kritisch gewürdigt. Die gestellten Anforderungen an den Prototyp werden überprüft und allfällige Abweichungen dokumentiert. Anschliessend sollen mittels einer Diskussion die Voraussetzungen zur Beantwortung der zu Beginn gestellten Forschungsfragen geschaffen werden. Zuletzt sollen Handlungsempfehlungen an Feldschlösschen abgegeben werden, ob ein allfälliges Folgeprojekt aufgrund der Ergebnisse empfohlen wird und welche kritischen Erfolgsfaktoren dabei zu beachten sind (Balzert et al., 2011, S. 78).

Wird abschliessend die beschriebene Methodik mit den explorativ analysierten Methoden der Wirtschaftsinformatik von Balzert et al. (2011, S. 285) verglichen, so besteht die höchste Übereinstimmung mit den Methoden Aktionsforschung und Prototyping. Mit der ausgeführten und festgelegten Methodik kann folgend mit der initialen Projektphase – der Analyse begonnen werden.

4 Analyse

Mit diesem Kapitel beginnt die Analysephase dieser Arbeit. Zu Beginn wird sich auf das Business-Ökosystem der Feldschlösschen Getränke AG konzentriert und daraus ein Use Case für den Prototyp abgeleitet. Basierend auf dem eruierten Use Case wird die geeignete Blockchain Plattform evaluiert, woraufhin die Ist-Prozesse und Ziele der Akteure analysiert werden. Die Informationen zur Analyse wurden im direkten Gespräch[1] mit Feldschlösschen entnommen.

4.1 Analyse des Business-Ökosystems der Feldschlösschen Getränke AG

Feldschlösschen besitzt kein verwaltetes Business-Ökosystem. Die aktuellen Aufwände beschränken sich auf die Customer-Supply-Chain, die alle nachgelagerten Akteure von Feldschlösschen umfasst. Laut Thomas Stadler, dem Leiter der Customer-Supply-Chain, ist sein Team darum bemüht, dass die Produkte jederzeit in der richtigen Menge, zur richtigen Zeit und am richtigen Ort beim Kunden verfügbar sind (Stadler, o. J.). Grund für diese eingeschränkte Supply-Chain-Betrachtung ist, dass die Orchestrierung der vorgelagerten Wertschöpfungsschritte zentral von Carlsberg für die gesamte Gruppe übernommen wird. Ziel der Analyse in dieser Arbeit ist jedoch, eine holistische Sicht auf die Stakeholder von Feldschlösschen zu ermöglichen, sodass potenzielle Teilnehmer eines Business-Ökosystems identifiziert werden können.

4.1.1 Identifizierung der Teilnehmer des Business-Ökosystems

Die Identifizierung der Teilnehmer für ein Business-Ökosystem wird nach dem Modell von Yue (2013) durchgeführt, welches in der Methodik in Kapitel 3 eingeführt wurde. Zur Komplexitätsreduktion des Modells werden jedoch nicht die einzelnen Firmen aufgeführt, sondern lediglich eine Repräsentation in Form ihrer Branche / Tätigkeit. Die Ergebnisse der Analyse werden im Netzwerkdiagramm in Abbildung 3 gezeigt.

Im «Core Business Circle» befinden sich die direkten Kunden. Da Feldschlösschen vor allem im B2B-Bereich tätig ist, sind dies regionale Depositäre (Getränkehändler), Detailhändler und Gastrobetriebe (Restaurants, Bars, Stadien und öffentliche Anlagen). Allerdings verfügt das Unternehmen auch über einen B2C-Kanal, den Webshop beer4you.ch,

[1] S. Schorno, Video-Konferenz, 11. März 2019

über den Endkonsumenten direkt beliefert werden. Auf der anderen Seite befinden sich alle Lieferanten für die Bierproduktion und die Lieferanten der Handelswaren.

Der «Extended Business Circle» umfasst hauptsächlich die Lieferanten weiterer Stufen. Dazu zählen Hopfen- und Getreidebauern sowie die Partnerbrauereien der Gruppe, welche Feldschlösschen mit Bier für den Handel versorgen. Die Partnerbrauereien können jedoch nicht nur isoliert in diesen Ring eingeteilt werden, da sie eine partnerschaftliche Unterstützungsfunktion übernehmen und deshalb auch dem «Support and Maintenance Circle» angehören.

Der «Support and Maintenance Circle» beinhaltet in ihrer Funktion sehr unterschiedliche Akteure. Im Falle von Feldschlösschen sind dies Banken, die für die Zahlungsabwicklung im System zuständig sind und Versicherungen, die Risiken während der Wertschöpfung mindern. Die beiden Akteure sind deshalb nicht mit anderen Knoten verbunden, weil diese Dienste von jedem Teilnehmer in Anspruch genommen werden und deren Verbindung zu einer Unübersichtlichkeit des Diagramms führen würde. Weitere wichtige Akteure in diesem Ring sind Mitarbeiter sowie deren Gewerkschaften und Verbände, Technologiepartner (darunter Swisscom und Microsoft), mit denen zukünftige Geschäftsmodelle und Plattformen entwickelt werden und staatliche Organisationen (Zölle, Wirtschaftskontrolleure, WEKO, Steuerämter etc.), welche die Compliance in der Wertschöpfung überwachen. Feldschlösschen besitzt mehrere Zertifizierungen, die dem Kunden gewisse Produkteigenschaften garantieren und dadurch Wert schaffen. Somit gehören auch Zertifizierungsstellen (z. B. das Bureau Veritas) diesem Ring an.

Im äussersten Ring, dem External Environment Circle, befinden sich die übrigen Anspruchsgruppen von Feldschlösschen, die aus allen Umweltsphären des PESTEL-Frameworks stammen. Dazu gehören beispielsweise Anwohner, die den Durchfahrtsverkehr hinnehmen müssen, in der entgegengesetzten Richtung jedoch Heizenergie durch die bei der Bierproduktion entstehende Abwärme erhalten. Weiter umfasst der Ring NGOs wie Swissaid oder Public Eye, die fordern, keine Patente auf pflanzliche Rohstoffe zu beantragen (Pfander, 2016). Akteure aus der ökonomischen Umweltsphäre stellen die Konkurrenten dar. Feldschlösschen konkurriert mit nationalen (Schützengarten, Locher etc.) und internationalen (Heineken, Anheuser Bush etc.) Mitbewerbern.

4.1.2 Analyse der Beziehungen zwischen den Teilnehmern

Nachdem die möglichen Teilnehmer des Business-Ökosystem von Feldschlösschen identifiziert wurden, können nun die Beziehungen zwischen den Teilnehmern kategorisiert werden. Die Kategorisierung soll zu einem späteren Zeitpunkt für die Rollenverteilung im Prototyp verwendet werden. Die Ergebnisse der Kategorisierung werden in Tabelle 2 aufgezeigt. Aus Gründen der praktischen Relevanz für den Prototyp und der Übersichtlichkeit der Matrix wurde sich bei der Kategorisierung auf die Beziehungen, dargestellt im Netzwerkdiagramm in Abbildung 3, beschränkt.

4.2 Anwendungsfälle der Blockchain-Technologie im Business-Ökosystem der Feldschlösschen Getränke AG

Die Entstehung eines verwalteten Business-Ökosystem benötigt Zeit und kann anhand eines Lebenszyklus aufgezeigt werden (Rong et al., 2015). Zu Beginn des Lebenszyklus können nicht alle Stakeholder auf einmal eingebunden werden (Rong et al., 2015). Auch sollen nur die Stakeholder einbezogen werden, die weiteren Wert generieren und das Ökosystem so in seiner Gesamtheit weiterentwickeln lassen (Co-Evolution) (Rong et al., 2015). In Bezug auf Feldschlösschen wäre es kaum möglich, innerhalb einer nützlichen Frist und mit den vorhandenen Ressourcen, eine Plattform mit lose gekoppelten Verbindungen zu entwickeln, die alle Lieferanten, Partnerbrauereien, die Konkurrenz und unterstützende Stakeholder wie Banken und Versicherungen in einem Marktplatz verbindet. Solch grosse Ökosysteme müssten aus einem Konsortium der grössten Brauereien (Anheuser Bush, Heineken, Carlsberg) entstehen und wären ein vieljähriges Unterfangen. Deshalb wird der Aufbau eines Business-Ökosystems im Umfeld von Feldschlösschen nicht über eine zentrale Plattform, die alle Stakeholder miteinander verbindet möglich sein, vielmehr durch kleinere Domänen-Plattformen. Diese sollen ein Set aus Stakeholdern einbinden, welche vom gegenseitigen Informations-, Daten- und Leistungsaustausch profitieren. Anwendungsfälle für solche Domänen-Plattformen innerhalb des Ökosystems von Feldschlösschen werden nachfolgend aufgezeigt.

Nachhaltigkeits-Tracking: In der Strategie «Together Towards ZERO» hat sich Feldschlösschen zum Ziel gesetzt, den Co2-Ausstoss während der Bierproduktion bis 2022 um 50 Prozent zu reduzieren und bis 2030 soll die Produktion Co2-neutral sein (Feldschlösschen, o. J.). Neben dem Co2-Ausstoss belastet Feldschlösschen die Umwelt durch den Wasserverbrauch. Deshalb soll dieser bis 2022 gegenüber 2018 um 25 Prozent und

bis 2030 um 50 Prozent gesenkt werden (Feldschlösschen, o. J.). Feldschlösschen optimiert jedoch ausschliesslich die eigene Produktion und betrachtet nicht die gesamte Wertschöpfungskette. Ein dadurch entstehender Anwendungsfall ist, mithilfe einer blockchain-gestützten Plattform den Co2-Ausstoss und den Wasserverbrauch entlang des gesamten Wertschöpfungsprozesses zu erfassen. Der Ressourcenverbrauch könnte von der Charge bis auf die Flasche Bier heruntergerechnet und über eine Zertifizierungsstelle beglaubigt werden. Der Kunde und NGOs wären dadurch in der Lage, die Anstrengungen der Produzenten zu verfolgen und die Transparenz zum Ressourcenverbrauch in der Bierherstellung könnte gesteigert werden. Ausserdem würde aufgrund der Transparenz eine Art «Wettbewerb» zwischen den Supply-Chain-Teilnehmern entstehen, welcher Betrieb den Co2-Ausstoss am besten optimieren kann. Auch staatliche Organisationen könnten über den Zugriff auf die Plattform die benötigten Informationen zu Besteuerung des Co2-Ausstosses beziehen. Über eine Blockchain-Lösung wäre die Integrität und Rückverfolgbarkeit der Daten gesichert.

- **beteiligte Stakeholder:** *Supply-Chain-Mitglieder, Endkonsumenten, NGOs, Zertifizierungsstellen und staatliche Organisationen*

Loyalitätsprogramm: Loyalitätsprogramme sind in verschiedensten Industrien beliebte Werkzeuge, um Kunden zu binden (Schmitt, 2001, S. 86). Eine Plattform könnte es Kunden ermöglichen, Loyalitätspunkte zu sammeln und beim Einlösen Rabatte zu erhalten. In einem weiteren Rahmen könnten die Punkte über die gesamte Carlsberg-Gruppe gesammelt und verwendet werden. Feldschlösschen würde bei dieser Lösung durch die erhaltenen Daten zum Konsumverhalten der Kunden profitieren. Die Blockchain würde versichern, dass die Punkte nur einmal vergeben und eingelöst werden, die Kundendaten durch Kryptographie geschützt sind und die Ausgleich-Zahlvorgänge über die involvierten Parteien automatisieren werden. Dem Kunden könnte durch die Plattform eine höhere Transparenz über seinen Bierkonsum und seine bereits realisierten Loyalitätsvorteile geboten werden.

- **beteiligte Stakeholder:** *Endkonsumenten, Feldschlösschen, Detailhändler, Gastrobetriebe, Getränkehändler, Banken, Brauereien der Carlsberg-Gruppe*

Beer-Station: Einen neuen Verkaufskanal von Feldschlösschen stellt die Beer-Station dar. Die Beer-Station ist ein Getränkeautomat (Zapfanlage), an dem sich der Kunde mit einer Karaffe sein Bier selbst zapfen kann. Bisher sind nur Biere der Carlsberg-Gruppe

verfügbar – zukünftig könnte die Bierstation jedoch für weitere Anbieter geöffnet werden. Auf einer neuen Plattform würden Verkaufsrechte für eine Station gehandelt und mittels Smart Contracts könnte die automatische Nachbestellung und Zahlungsabwicklung realisiert werden. Ausserdem wäre die Anlage bei einem Defekt in der Lage, selbst Service-Mitarbeiter anzufordern und zu bezahlen. Durch die Blockchain-Technologie und die implementierten Smart Contracts könnte die Verkaufsstation völlig autonom agieren (einfaches Beispiel einer Dezentralisierten Autonomen Organisation – DAO).

- **beteiligte Stakeholder:** *Endkonsumenten, Feldschlösschen, Konkurrenten, Banken, Logistik- und technische Service-Dienstleister*

Die Anwendungsfälle zeigen, welche Chancen die Blockchain-Technologie in Business-Ökosystemen mit sich bringt und wie durch den Einbezug weiterer Stakeholder neue Geschäftsmodelle für die Teilnehmer entstehen können. Natürlich existieren weitere Anwendungsfälle wie der bekannte Supply-Chain-Use-Case der Rückverfolgung des Bieres von der Detailhandelsfiliale bis zum Hopfenbauer oder eine Dokumente-Austausch-Plattform für die Logistiklieferkette unter Einbezug der Zollstellen. Zuletzt wäre auch die Organisation der Logistik-Dienstleister über eine Auktionsplattform für Lieferaufträge möglich, bei der Verträge und Dokumente automatisch ausgetauscht werden könnten.

4.3 Vorstellung Use Case und Abgrenzung Prototyp

Die Vertragsverwaltung in der Blockchain verspricht, die statischen und papierbasierten Verträge abzulösen (Aitken, 2017). Smart Contracts sind in der Lage, sich aufgrund von sich ändernden Umwelteinflüssen dynamisch anzupassen und den Vertragsinhalt automatisch und ohne Kompromisse zu forcieren (Aitken, 2017). Laut Boshell (2018) ist die Blockchain zum heutigen Zeitpunkt vor allem für einfache und in ihrer Grundstruktur repetitive Verträge geeignet, weil ein Template des Vertrag-Gerüsts programmiert und in der Blockchain gespeichert werden muss (Boshell, 2018). Als Vorteile gegenüber bisherigen Systemen nennt Boshell (2018), dass nur eine einzige Blockchain (Ort der Datenspeicherung) für alle beteiligten Vertragsparteien existiert und dadurch eine einzige gültige Version des Vertrags sichergestellt werden kann. Die hinterlegten Verträge seien zudem von jedem internetzugänglichen Punkt der Welt und zu jeder Zeit ausführbar (Boshell, 2018). Noch mehr verspricht sich Boshell (2018) von den parallelen Revisions- und Signierungsprozessen und die damit verbundene Zeiteinsparung während der Vertragsentstehung. Dorschel und Manz (2017) heben die unveränderbare Rückverfolgbarkeit und

die nahtlose Dokumentation der Ereignisse hervor, weshalb das Vertragsmanagement ein geeigneter Use Case für eine Blockchain sei. Auch erwähnen Dorschel und Manz (2017) die Eignung von Smart Contract für einfache mathematisch auflösbare Regeln zur automatischen Vertragserfüllung. Jedoch sei zu beachten, dass in der Finanzindustrie Verträge existieren, die noch nicht in Programmcode abgebildet werden können und menschliche Bearbeitung erfordern (Dorschel & Manz, 2017). Weiter geben sie zu bedenken, dass die Sicherheit in vielen Blockchains von der Aufbewahrung des Private Keys abhängt (Dorschel & Manz, 2017). Kommt der zum Signieren verwendete Private Key abhanden, nützt die sichere Aufbewahrung des Vertrags in einer Blockchain wenig. Auch rechtliche Grundlagen für die digitale Vertragsspeicherung müssen geprüft werden und unterscheiden sich von Land zu Land.

Der Einsatz der Blockchain-Technologie im Vertragsmanagement bildet eine interessante Grundlage für einen Use Case und zur Untersuchung der derzeitigen Eignung sowie des Grads der möglichen Umsetzbarkeit. Als Vertragsmanagement Use Case für diese Arbeit soll das Rückvergütungs- und Darlehensgeschäft von Feldschlösschen dienen.

Mit dem Ziel einer langfristigen Zusammenarbeit versucht Feldschlösschen Partnerschaften mit Gastrobetrieben einzugehen, die auf einem Rückvergütungsvertrag basieren. Bezieht der Kunde von der Brauerei eine vereinbarte Warenmenge (festgehalten in Umsatz), erhält er einen entlastenden nachträglichen Rabatt in der Form einer Rückvergütung. Rückvergütungen werden basierend auf dem Sortimentsbezug vereinbart. In dieser Arbeit wird sich auf das Bier- und Mineralwasser-Sortiment von Feldschlösschen beschränkt. In anderen Worten wird in einem Rückvergütungsvertrag festgelegt, dass der Kunde bei einem jährlich erreichten Bier-Umsatz von beispielsweise CHF 100‘000.-, 10 Prozent des Einkaufspreises zurückerstattet erhält.

Zusätzlich zum Rückvergütungsvertrag hat der Kunde die Wahl, ein Darlehen zu beziehen. Feldschlösschen schliesst mit den Kunden somit Rückvergütungsverträge mit oder ohne Darlehen ab. Die Idee hinter der Verbindung von Rückvergütungs- und Darlehensgeschäften von Brauereien ist, dass Gastrobetriebe bei der Finanzierung von Investitionen unterstützt werden sollen, sodass beide Parteien vom folgenden wirtschaftlichen Erfolg des Gastrobetriebs profitieren können. Der Bezug eines Darlehens ist an mehrere Rahmenbedingungen gebunden. Beispielsweise muss der Gastrobetrieb ein gewisses Eigenkapital für die Investition vorlegen oder ein minimales Umsatzziel gegenüber

Feldschlösschen erfüllen. Innerhalb des Darlehensvertrags werden Bedingungen zu Tilgung und Zinssätze festgelegt. Die Höhe des Darlehens misst sich am prognostizierten Umsatz von Feldschlösschen mit dem Gastrobetrieb. Für die Erstellung des Prototyps soll sich ausschliesslich auf den Rückvergütungsvertrag mit einem Darlehensvertrag fokussiert werden.

Hintergrund für die Entstehung solcher Verträge ist, dass Banken oft nicht bereit sind, das Risiko mit neuen Gastrobetrieben einzugehen, da die Wahrscheinlichkeit eines Ausfalles der Rückzahlung in der Startphase eines Unternehmens gross ist (Sulc, 2014). Die Verträge werden meist über eine Laufzeit von ein bis fünf Jahren unterzeichnet und erhalten für den Gastrobetrieb dadurch eine bindende Wirkung an die Brauerei, die von Wirte-Verbänden oft kritisiert wird (Sulc, 2014). Je nach Vertrag werden Klauseln eingeführt, die den Wirt dazu verpflichten, ein festgelegtes Sortiment von der Brauerei zu beziehen (Sulc, 2014). Von Seiten der Brauereien heisst es, dass diese Massnahmen erforderlich seien, wenn ein solch grosses Risiko eingegangen wird (Sulc, 2014). Der detaillierte Prozess und die Entstehung des Darlehensgeschäfts wird in der Ist-Analyse unter Punkt 4.5 weiter erläutert. In Bezug auf das Business-Ökosystem sind die in Abbildung 4 aufgeführten Akteure in das Darlehensgeschäft involviert. In der Abbildung werden zudem die internen Stellen von Feldschlösschen aufgeführt, deren Rollen in der Ist-Analyse erklärt werden.

Ausser Feldschlösschen und den Gastrobetrieben sind Getränkehändler und Banken als Akteure aufgeführt. Die Banken vollziehen die Zahlungsflüsse zwischen den Teilnehmern und bilden eine Unterstützungsfunktion. Die Getränkehändler fungieren als Partner von Feldschlösschen. Es existiert die Möglichkeit, dass ein Gastrobetrieb die Waren als indirekter Kunde über einen Getränkehändler bezieht. In diesem Fall teilen sich Feldschlösschen und der Getränkehändler den Prozentsatz der Rückvergütung in einem festgelegten Verhältnis. Mit dem Einbezug von Banken und Getränkehändlern in Abbildung 4, soll die Chance aufgezeigt werden, neue Akteure in das Ökosystem einzubeziehen, welche dieses zeitweilig unterstützen. Für die Abgrenzung des Use Cases sollen im Prototyp jedoch nur die Parteien Gastrobetrieb und Feldschlösschen umgesetzt werden. Dem Prototyp müssen deshalb keine weiteren Kunden dynamisch hinzugefügt werden können. Auch im weiteren Verlauf der Analyse sollen ausschliesslich diese beiden Akteure fokussiert werden.

4.4 Evaluation der Blockchain-Plattform anhand des Use Cases

Die Auswahl der Blockchain-Plattform wird in zwei Schritte unterteilt. Im ersten soll evaluiert werden, ob eine Blockchain benötigt wird und falls ja, welcher Typ von Blockchain-Technologie am geeignetsten für den Prototyp des Rückvergütungs- und Darlehensvertragsmanagements ist. Mit der gewonnenen Erkenntnis soll nachfolgend die passende Plattform eruiert werden, indem die Vor- und Nachteile der verschiedenen Anbieter einander gegenübergestellt werden.

4.4.1 Auswahl des Blockchain-Typs

Die auf den Use Case bezogene Beurteilung des am geeignetsten Blockchain-Typs bedingt ein Evaluationsprozess. Zur Strukturierung der Evaluation existieren in der Literatur verschiedene Vorgehensmodelle. Aus den eruierten Modellen wurde der Entscheidungsbaum von Wüst und Gervais (2018), das auf dem World Economic Forum vorgestellte Modell von Mulligan, Rangaswami, Warren und Zhu-Scott (2018), das aus der Hyperledger-Community stammende Flussdiagramm von Zubko und Bohner (2018) und das vom National Institute of Standards and Technology (US) standardisierte Modell von Yaga, Mell, Roby und Scarfone (2018) ausgewählt. Dabei wurden die für den Prototyp kritischen Entscheidungsfragen der Modelle extrahiert und in einem neuen Modell zusammengefügt. Das entstandene Modell ist in Abbildung 5 visualisiert. Mithilfe des Modells und der Beantwortung der Fragen soll nun der geeignete Typ für den Proof-of-Concept-Prototyp gefunden werden.

Die Auswertung des Modells und die Beantwortung der gestellten Fragen, abgebildet in Tabelle 3, zeigen, dass eine Private- oder Konsortium- (permissioned) Blockchain den geeigneten Blockchain-Typ für den Prototyp darstellt. Unter der Verwendung dieser Erkenntnis kann laut Maple und Jackson (2019) mit der Auswahl der Plattform begonnen werden.

4.4.2 Auswahl der Blockhain Plattform

Die ersten Blockchain-Plattformen, angeführt von Bitcoin, spezialisierten sich auf Protokolle für den Austausch von Kryptowährungen (Meier, 2018). Nach und nach entstanden immer mehr Plattformen, die auch auf andere Domänenbereiche, wie die Supply Chain, abzielten oder versuchten, eine modulare Plattform für industrieübergreifende Anwendungsbereiche bereitzustellen (Meier, 2018). Das wohl bekannteste Beispiel einer solchen Plattform ist Ethereum. Allerdings ist Ethereum eine öffentliche Blockchain und

wird deshalb nicht in die Evaluation miteinbezogen. Die in der Literatur meistgenannten privaten Blockchains sind laut Valenta und Sandner (2017) Hyperledger Fabric und R3 Corda. Eine in der Fachliteratur weniger behandelte, doch in Blogs oft referenzierte Plattform, ist Quorum von J.P. Morgan, die auf dem Ethereum-Netzwerk basiert. Natürlich existieren weitere private oder hybride Plattformen, wie MultiChain, Nextledger oder Loopchain, jedoch wird sich in dieser Arbeit aufgrund der Informationsverfügbarkeit und der öffentlich zugänglichen Dokumentation, auf die vorher genannten drei Plattformen beschränkt (Maple & Jackson, 2019).

Das grundsätzliche Ziel von privaten Plattformen ist, die Probleme der Privacy und der geringen Transaktionsraten in öffentlichen Blockchains zu lösen (Kwak et al., 2019). Die Umsetzung unterscheidet sich zwischen den Plattformen jedoch markant. Deshalb sind die grundlegenden Eigenschaften der Plattformen sowie die für diese Arbeit relevanten Ergebnisse aus den Vergleichen von Höfelmann und Sander (2019) und von Maple und Jackson (2019) in Tabelle 4 ersichtlich. In Bezug auf den Use Case sind dabei folgende Differenzierungsmerkmale hervorzuheben:

- Hyperledger Fabric sichert die Privacy über die Zugriffsverwaltung der Teilnehmer auf das Netzwerk oder Sub-Netzwerk (Channels). In Corda sind nicht nur das Netzwerk, sondern auch die einzelnen Transaktionen privat. Der Konsens wird daher nicht auf Netzwerk-, sondern auf Transaktionsebene erzielt und eine Transaktion kann nur von den beteiligten Parteien eingesehen werden. Quorum legt einen weiteren Layer über Ethereum, welcher der Plattform erlaubt, private Smart Contracts zu implementieren, die nur von ausgewählten Teilnehmern ausgeführt werden können. In den Smart Contracts können Attribute und Rollen der Teilnehmer abgefragt werden, wodurch der Zugriff auf Informationen gesteuert wird.
- Quorum weist den geringsten Installationsaufwand vor Corda und Fabric auf. Zwar muss auch ein Node über ein vorgefertigtes Git-Hub-Projekt konfiguriert werden, doch wird die Netzwerksteuerung durch das vorhandene Ethereum-Netzwerk vereinfacht. Mit der Verwendung von Docker-Containern ist das Set-up von Hyperledger Fabric das zeitintensivste. Durch leichtgewichtigere und eigenständige JVM-Maschinen wird der Konfigurationsaufwand für Corda als geringer eingeschätzt. Ausserdem verfügt Corda über die umfangreichste Dokumentation.

- Keine der drei Plattformen besitzt eine eingebundene (native) Kryptowährung. In Hyperledger Fabric und Corda können aber mithilfe von Smart Contracts Tokens erstellt werden, die einen digitalen Asset für das Netzwerk repräsentieren. In Quorum ist es durch das zugrundeliegende Ethereum möglich, Transaktionen mit Ether (native Kryptowährung von Ethereum) durchzuführen.
- Corda ermöglicht über Anhänge an eine Transaktion, rechtlich bindende Vereinbarungen zu hinterlegen (Legal Pros) und ist deswegen unter anderem für den Finanzmarkt konzipiert. Hyperledger Fabric wurde nicht für einen speziellen Anwendungsbereich entwickelt und weist die höchste Modularität auf. Quorum entstand ebenfalls dem Finanzsektor, ist jedoch wie Corda auch für andere Anwendungsfälle adaptierbar.

Auf die technischen Unterschiede, wie die unterschiedlichen Konsensus-Verfahren, wird in diesem Abschnitt nicht eingegangen. Die technischen Konzepte werden in der Implementation lediglich für die ausgewählte Plattform detaillierter expliziert. Für die Auswahl der Blockchain-Plattform sollen ausserdem nicht nur technische Eigenschaften berücksichtigt werden, sondern auch Faktoren in Bezug auf die Entwicklung des Prototyps. In Tabelle 5 werden deshalb Vor- und Nachteile der Plattformen unter der Berücksichtigung des Use Cases aufgelistet, die auf den Nachforschungen von Sandner (2018) basieren.

Unter der Betrachtung der Ergebnisse aus Tabelle 5 wird Corda als die geeignetste Plattform für den Prototyp des Rückvergütungs- und Darlehensvertragsmanagement von Feldschlösschen angesehen. Ausschlaggebend ist die Spezialisierung von Corda auf Finanzdienstleistungen, die Möglichkeit, den digitalen Verträgen rechtliche Vereinbarungen anzufügen, die einfachere Implementierung für einen Proof-of-Concep-Prototyp und die grosse Community, was zu einer höheren Verfügbarkeit an Dokumentation führt. Abschliessend kann aber nicht gesagt werden, welche Plattform langfristig für welchen Anwendungsfall die geeignetste ist, denn die Plattformen entwickeln sich in einer hohen Geschwindigkeit weiter (Kwak et al., 2019). Ausserdem existieren grosse Überlappungen zwischen den Communities (unterstützende Unternehmen), was die Unsicherheit im Entscheid der zukunftsträchtigsten Plattform zeigt (Kwak et al., 2019). Zuletzt muss beachtet werden, dass bei dieser sich schnell weiterentwickelnden Technologie, sich ständig neue Anbieter auf den Markt drängen und die bestehenden Plattformen durch Verbesserungen herausfordern. Daher muss sich in der Praxis erst noch zeigen, welche Plattformen sich in Zukunft durchsetzen werden (Kwak et al., 2019).

4.5 Ist-Analyse Rückvergütungs- und Darlehensgeschäft

In der Ist-Analyse werden die bestehenden Prozesse von Feldschlösschen im Rückvergütungs- und Darlehensgeschäft sowie die Ziele der Akteure untersucht. Ausserdem sollen alle für die Umsetzung des Prototyps benötigten geschäftsbezogenen Informationen von Feldschlösschen beschaffen werden.

4.5.1 Analyse Ist-Prozess und Informationsflüsse

Laut Aagesen und Krogstie (2010) hat das Modellieren von Prozessen eine lange Tradition in der Analyse, Steuerung und Verbesserung von Geschäftsabläufen. Mit der Zunahme der Geschäftsprozess-Automatisierung durch Workflow-Engines nimmt die Bedeutung der Modellierung weiter zu (Aagesen & Krogstie, 2010). Prozessmodelle ermöglichen Simulationen und Berechnungen, mit denen Verbesserungen in der Durchlaufgeschwindigkeit und in der Qualität der Prozesse erreicht werden können (Aagesen & Krogstie, 2010). Ausserdem bilden sie die Grundlage für verschiedenste Change-Management-Vorgehensmodelle (Milton & Johnson, 2012). Aus diesen Gründen werden nun der vorhandene Ist-Prozess und die Informationsflüsse des Rückvergütungsgeschäfts von Feldschlösschen sowie die beteiligten Teilnehmer und deren Ziele analysiert.

Zur Visualisierung des bestehenden Prozesses- und der Informationsflüsse wurde ein BPMN-Diagramm erstellt, welches in Abbildung 7 gezeigt wird. BPMN wurde dem UML-Activity-Diagramm vorgezogen, da die Notation dem Praxispartner bereits bekannt war. Das Diagramm weist für jeden im Prozess involvierten Akteur eine Lane (Bahn) auf. Das heisst, für die feldschlösschen-internen Funktionen Sales Manager, Backoffice und Vorgesetzter sowie für den externen Gastrobetrieb. Bei der Betrachtung des Diagramms sind verschiedene Probleme des aktuellen Prozesses identifizierbar. Der Sales Manager tritt mit dem Kunden in Kontakt, sobald er an einer Partnerschaft interessiert ist. Er besitzt aber keine Möglichkeit den Kunden zu Beginn auf vergangene Fälle oder die Kreditwürdigkeit zu prüfen, womit ein Vertragsschluss zu einem frühen Zeitpunkt verhindert werden könnte. Bei einer ersten Besprechung werden grundlegende Informationen wie Umsatz, Sortimentswünsche oder Darlehenshöhe ausgetauscht, auf welchen der Sales Manager eine Offerte erstellt. Überarbeitungen der Offerte können mehrere Iterationen in Anspruch nehmen. Die darauffolgende Nachweiserbringung und Beschaffung von Dokumenten erfolgt via E-Mail. Dies erschwert die Rückverfolgbarkeit der Vertragsentstehung und führt bei fehlenden Dokumenten zu aufwendigen Nachbeschaffungen zwischen den

Parteien. Für den Prototyp wurden zusammen mit Feldschlösschen repräsentative Nachweise und Vertrags-Informationen definiert, welche in Tabelle 6 aufgelistet werden. Sind alle Informationen vorhanden, wird der Kunde von einem Backoffice-Mitarbeiter manuell beurteilt. Einerseits werden die erhaltenen Informationen, andererseits die Kreditwürdigkeit über die Portale Moneyhouse und Crif überprüft. Aufgrund der erhaltenen Informationen erstellt das Backoffice Entwürfe der benötigten Verträge. Die eingesetzten Konditionen werden in der Verhandlung zwischen Kunden und Sales Manager diskutiert und wenn nötig angepasst.

Der Sales Manager besitzt in den Verhandlungen einen gewissen Handlungsspielraum, der von Feldschlösschen als «Goldene Regeln» bezeichnet wird. Bleibt der Sales Manager innerhalb dieser Regeln, kann der Signierungsprozess nach den Vertragsanpassungen durch das Backoffice beginnen. Ansonsten müssen die Verträge einem Vorgesetzten vorgelegt werden und weitere Iterationen aus Verhandlungen und Genehmigungen können entstehen. Die Rückverfolgbarkeit der Vertragsentstehung ist durch persönliche Kontakte und E-Mail-Korrespondenz nicht sichergestellt und der sequenzielle Signierungsprozess ist aufgrund des Postversandes oder der persönlichen Übermittlung zeitintensiv.

Die im Vertrag enthaltenen Konditionen werden vom Backoffice zusätzlich in das ERP-System eingetragen. Anschliessend werden die signierten Verträge von Feldschlösschen physisch abgelegt und eine Kopie davon je an Gastrobetrieb und Depositär versandt. Mit der Ablage endet der für den Prototyp betrachtete Prozess der Vertragsentstehung.

4.5.2 Analyse der Ziele und Motive der Teilnehmer

Die Untersuchungen von Lavikka et al. (2017) haben ergeben, dass die Suche nach gemeinsamen Zielen ein entscheidender Erfolgsfaktor für die Gestaltung einer Business-Ökosystem-Plattform darstellt. Aus diesem Grund wurden die Ziele der einzelnen Teilnehmer zusammengetragen und die Ergebnisse in Tabelle 7 notiert. Die Analyse zeigt, dass die funktionalen Ziele der Teilnehmer durch die Plattform erreichbar sind. Hingegen kann die Plattform bei den meisten konträren wirtschaftlichen Zielen keine Verbesserung bewirken. Personalaufwände aller Beteiligten können lediglich gesenkt, wodurch Kosten gespart werden. Nach der Analyse der Ziele muss laut Lavikka et al. (2017) sichergestellt werden, dass diese in der Anforderungsdefinition berücksichtigt werden. Nur durch die Erfüllung der Teilnehmer-Ziele kann der zukünftige Erfolg und die Mitwirkung aller Beteiligten an der Plattform erreicht werden (Lavikka et al., 2017).

5 Planung und Konzeption

In der Planung und Konzeption soll der Prototyp designt werden. Mit dem Begriff Design werden nicht nur ästhetische, sondern auch funktionale und technische Aspekte beschrieben. Daher werden zuerst die für den Prototyp erforderlichen Anforderungen definiert, die in einem direkten Gespräch[2] mit Feldschlösschen identifiziert wurden. Basierend auf den Anforderungen soll darauffolgend der Soll-Prozesse ausgearbeitet und modelliert werden. Zuletzt können mit dieser Grundlage die Mock-ups für den Prototyp ausgestaltet werden.

5.1 Definition Anforderungen

Die Anforderungen an den Prototyp sollen innerhalb von User Stories definiert und anschliessend durch technische Systemanforderungen ergänzt werden.

5.1.1 User Stories

User Stories sind eine bekannte Methode, um Funktionalitäten einer Software zu beschreiben (Lucassen, Dalpiaz, Werf & Brinkkemper, 2016). Mit ihnen soll der Nutzen für den Kunden oder den Endbenutzer erfasst werden (Cohn, 2004). Aus diesem Grund werden User Stories aus der Perspektive des Benutzers verfasst und beinhalten, *wie* und zu *was* dieser die Software gerne verwenden möchte, ohne auf die technische Umsetzung einzugehen (Nguyen, Gallagher, Read & de Vreede, 2009). Durch die Vermeidung spezifischer Fachsprache soll die Kommunikation zwischen Benutzern und Entwicklern verbessert sowie Diskussionen im Projektteam angeregt werden (Nguyen et al., 2009). User Stories sind ein integraler Bestandteil verschiedener agiler Software Entwicklungs-Methoden und besitzen gegenüber klassisch ausformulierten Anforderungen den Vorteil, dass sie leichtgewichtiger und dadurch schneller anpassbar sind (Nguyen et al., 2009). Weil die technische Umsetzung nicht von Beginn an definiert wird, soll diese in den Iterationsbesprechungen (in Scrum → Daily Scrum) thematisiert und so die Expertise aller Projektmitarbeiter genutzt werden (Nguyen et al., 2009). Probleme mit User Stories entstehen, wenn der Endbenutzer Domänen-Wissen nicht ausspricht oder die kurzen Statements ungenau ausformuliert werden (auftreten von Spracheffekten), wodurch unterschiedliche Interpretationen entstehen können (Nguyen et al., 2009). Aus diesen Gründen müssen User Stories folgende Merkmale aufweisen (Cohn, 2004):

[2] S. Schorno, Video-Konferenz, 19. März 2019

- User Stories müssen Nutzen für den Kunden oder Benutzer beinhalten.
- User Stories dürfen keine unterschiedlichen Interpretationen zulassen. Dies soll durch Gespräche im Projektteam und durch eine genaue Formulierung (Templates) sichergestellt werden.
- User Stories sollen keine technischen Details enthalten.
- User Stories sollen eine adäquate Länge aufweisen. Sie müssen nicht atomar sein, sollen aber auch nicht in grosser Anzahl kombiniert werden.
- User Stories müssen verhandelbar, überprüfbar und ihr Aufwand messbar sein.

Für die Erstellung von User Stories werden oft Templates verwendet, damit die Lesbarkeit für den Kunden vereinfacht und die Genauigkeit der Formulierung sichergestellt wird (Lucassen et al., 2016). Ein oft verwendetes Template ist *«Als ⟨Rolle⟩ möchte ich ⟨Ziel/Wunsch⟩, [sodass / um ⟨Nutzen⟩]»* (Lucassen et al., 2016). Der Nebensatz in der eckigen Klammer (Nutzen) ist bei diesem Template optional und dient dem besseren Verständnis bei Unklarheiten.

Trotz der Nachteile bezüglich des Detaillierungsgrads gegenüber klassischen Requirements sind User Stories aufgrund ihrer Leichtgewichtigkeit und Verständlichkeit die geeignete Methode für die Anforderungsdefinition des Prototyps. Die in Zusammenarbeit mit Feldschlösschen festgelegten User Stories, sind in Tabelle 8 festgehalten.

5.1.2 System-Anforderungen

Nachdem die von den Stakeholdern benötigten Funktionalitäten innerhalb von User Stories definiert wurden, können im Requirements-Engineering-Prozess nach Hull et al. (2011, S. 29) zusätzlich Anforderungen an das System (Prototyp) definiert werden. Die System-Anforderungen müssen allerdings an den User Stories ausgerichtet sein und dürfen diese nicht beeinflussen (Hull et al., 2011, S.30). Sie beinhalten oft einen technischen Aspekt und definieren, *was* das System können muss (Hull et al., 2011, S.30). Die Definition von Systemanforderungen ist kein Bestandteil der im vorherigen Absatz beschriebenen iterativen Vorgehensweise und bezieht sich auf das klassische Requirements Engineering (Nguyen et al., 2009). Jedoch entfällt mit der Einschränkung eines einzelnen Projektmitarbeiters die Möglichkeit von technischen Vorgehensbesprechungen. Ausserdem soll dem Leser aufgezeigt werden, welche technischen Funktionen der Prototyp umfasst. Aus diesen Gründen wurde sich für die Definition von System-Anforderungen entschieden, die in Tabelle 9 aufgeführt sind.

5.2 Definition Soll-Prozess und Informationsflüsse des Prototyps

Die Definition des Soll-Prozesses vereint die definierten Anforderungen mit den analysierten Problemstellungen aus der Ist-Analyse (Funk, Gómez, Niemeyer & Teuteberg, 2010, S. 21). Auch der für den Prototyp entwickelte Soll-Prozess wurde in Zusammenarbeit mit der Feldschlösschen ausgearbeitet und berücksichtigt die in Kapitel 1.6 und 4.3 definierten Abgrenzungen.

Weil im Ist-Prozess kein prozessunterstützendes System verwendet wird (mit Ausnahme der Eingabe der Rückvergütungskonditionen in das ERP-System) und der Ablauf von manuellen Tasks getrieben ist, wurde auf die Typisierung der Aktivitäten verzichtet. Mit dem Prototyp entsteht ein solches System, weshalb eine Typisierung mit den Elementen in Abbildung 6 vorgenommen wurde.

Der Ablauf des resultierten Prozesses, veranschaulicht in Abbildung 8, unterscheidet sich wenig zu jenem des Ist-Prozesses. Änderungen ergeben sich weniger in den Aufgaben an sich, sondern vielmehr durch die technische Unterstützung des Systems. Die einflussreichsten Veränderungen werden nachfolgend vorgestellt.

- Durch die von Beginn an verfügbare Identität des Kunden im System (vorherig definierter Node), soll die Kreditwürdigkeit automatisch zu einem frühen Zeitpunkt geprüft und so Fehlaufwände vermieden werden. Ausserdem soll durch die neue Rückverfolgbarkeit der Vertragsverhältnisse, der Vertragsschluss mit früheren Problemkunden verhindert werden.
- Sämtliche Informationen werden in einem System erfasst und müssen nicht mehr ausgetauscht werden. Dies spart Zeit und ermöglicht eine automatische Validierung der Dokumente und Informationen, sodass der Nachbeschaffungsprozess minimiert werden kann.
- Anträge, Offerten und Vertrags-Entwürfe werden aufgrund der eingegebenen Informationen automatisch generiert und müssen nicht mehr manuell von einem Backoffice-Mitarbeiter erstellt werden. Mit dem optimierten Prozess kann sich der Backoffice-Mitarbeiter ausschliesslich auf das Prüfen der Kunden konzentrieren, statt viel Zeit für administrative Arbeiten aufzuwenden.

- Nach der persönlichen Verhandlung müssen die Vertrags-Entwürfe nicht mehr ausgetauscht werden. Die Verhandlung soll aber weiterhin persönlich stattfinden, anstatt durch parallele Vorschläge von Konditionen im System. Zudem soll die Verifizierung des Vertrags durch den Vorgesetzten weiterhin sequenziell angeordnet bleiben, damit Doppelarbeiten verhindert werden. Trotzdem kann der Prozess durch den digitalen Austausch der Dokumente und die digitale Signierung beschleunigt werden.
- Der Eintrag des endgültigen Vertrags in die Blockchain und in das ERP-System soll von Services automatisiert werden.

Zusammengefasst zeigen die Anpassungen am Vertragsentstehungsprozess, dass der Arbeitsaufwand durch eine zentrale Erfassung und digitale Verträge reduziert werden kann. Insbesondere im Bereich Informations- und Datenfluss können Effizienzsteigerungen erzielt werden. Darüber hinaus wird durch einen standardisierten Prozess und einer spezifisch für die Aufgabe erstellten Applikation, die Arbeit der Mitarbeitenden erleichtert.

5.3 Mock-ups

Mock-ups können in unterschiedlichen Detaillierungsgraden erstellt werden. Weisen sie einen geringen Detaillierungsgrad auf, ähnlich einer Skizze, werden sie Wireframe genannt (Rivero et al., 2010). Mock-ups mit einem hohen Detaillierungsgrad beinhalten bereits das ausgestaltete Design samt Buttons, Icons, Farben sowie Schriftarten und können oft auf dem gewünschten Endgerät angezeigt und durchgeklickt werden (Rivero et al., 2010). Vorteile einfacher Mock-ups sind die kürzere Erstellungszeit und der Fokus auf die Erfüllung der Anforderungen, statt des ästhetischen Designs (Koren & Klamma, 2018, S.103). Hingegen sind die Vorteile von detaillierten Mock-ups die Nähe zum Endprodukt und die Wiederverwendbarkeit der Designelemente während der Programmierung (Rivero et al., 2010).

In dieser Arbeit wurde sich für einen detaillierten Prototyp entschieden. Gründe dafür waren erstens, dass der Aufwand gegenüber dem Nutzen für einen Prototyp zu gross wäre, um zuerst Wireframes und dann Mock-ups zu entwickeln. Zweitens unterstützen detaillierte Mock-ups die Kommunikation mit dem Praxispartner dahingehend, dass sie Verständnisprobleme und den Interpretationsspielraum während der Diskussion minimieren. Drittens überwogen die Vorteile der Wiederverwendbarkeit der bereits ausgestalteten Elemente und die Vereinfachung der Umsetzung mit einem vorhandenen Design. Durch

den Entscheid für detaillierte Mock-ups sind einfache Grafik-Tools wie Balsamiq oder Lucidchart nicht geeignet. Die Wahl fiel auf Sketch, ein für Mock-ups optimiertes Vektorgrafikprogramm. Mit Sketch war es möglich in angemessener Zeit Mock-ups zu erstellen, welche über die Sketch Cloud [3] geteilt werden konnten, sodass ein stetiger Austausch mit dem Praxispartner stattgefunden hat. Die ausgestalteten Mock-ups sind in Abbildung 9 bis 14 aufgeführt. Aufgrund der hohen Überschneidungsrate der Erklärungen zu den Mock-ups sowie der fertiggestellten Applikation wurde sich gegen eine Erläuterung in diesem Kapitel entschieden. Die ausgestalteten Ansichten der Web-Applikation werden in Kapitel 7.4 vorgestellt.

[3] https://sketch.cloud/s/75JEE/a/Gp940v/play

6 Implementation

Mit der Implementation beginnt die praktische Umsetzung des Prototyps. In diesem Kapitel werden dazu die verwendeten Technologien und deren Zusammensetzung näher vorgestellt. Für die Erläuterungen der Technologien soll die Architektur des Prototyps als Leitfaden dienen und in einem Bottom-up-Ansatz expliziert werden. Unter der Berücksichtigung der Anforderungen für den Prototyp ist die in Abbildung 15 visualisierte Architektur entstanden.

6.1 Corda

Corda ist eine Open-Source-Blockchain-Plattform, gegründet und weiterentwickelt vom R3-Konsortium und der Corda-Community (R3-Consortium, o. J.). Das R3-Konsortium verbindet führende Finanzinstitute (HSBC, Deutsche Bank, UBS etc.), Technologiekonzerne (Microsoft, Intel, Amazon etc.) und zahlreiche Start-ups (R3-Consortium, o. J.). Auch wenn Corda als Blockchain-Technologie vermarktet wird, so wird Corda in der Literatur nicht als klassische Blockchain wie Bitcoin angesehen, sondern vielmehr als Distributed Ledger Technology (DLT) mit integrierten Blockchain Konzepten (Sandner, 2018; Newton, 2018). Die Unterschiede werden an entsprechenden Stellen dieses Kapitels genauer aufgezeigt. In der Literatur sowie in der Corda Dokumentation werden die Technologien zur Lösung der gleichen Problemstellungen mit überschneidenden Konzepten angesehen und zur Vereinfachung für den Leser oft als Synonyme verwendet (Höfelmann & Sander, 2019). Diese Praktik soll auch in dieser Arbeit verwendet werden, auch wenn einem bewusst sein muss, dass technische Unterschiede zwischen den Begriffen bestehen. Eine DLT wird immer dann benötigt, wenn in verschiedenen Organisationen unterschiedliche Datenbanken existieren, die synchronisiert werden müssen und kein vollständiges Vertrauensverhältnis besteht (Gendal, Carlyle, Grigg & Hearn, 2016). Dieses Problem tritt beispielsweise zwischen Banken auf. In einer Banktransaktion muss sichergestellt werden, dass ein Kundenkonto bei Bank A abnimmt und das Konto der begünstigten Partei bei Bank B zunimmt. Zusätzlich müssen die Banken versichern können, dass ein Betrag von einem Kontoinhaber nicht zweimal ausgegeben werden kann (Double-Spending-Problem) (Newton, 2018). Für eine internationale Banktransaktion werden dazu mehrere Synchronisationsprozesse und Parteien benötigt (Newton, 2018). Viel einfacher wäre es laut Newton (2018), wenn die Banken auf eine gemeinsame Datenquelle zugreifen würden. Die entstehenden Probleme einer zentralen Datenhaltung

sind jedoch, dass zum einen eine einzelne Organisation die Datenhoheit besitzt und zum andern ein sogenannter «Single-Point-of-Failure» entsteht (Newton, 2018). Das Ziel einer DLT ist deshalb, eine einzig gültige, aktuelle, konsistente und integre Version einer Datenbank (Ledger) über ein Netzwerk zu verteilen (Moothart & Escoto, 2018). Die Integrität der Datenbanken und der sich darin befindenden Transaktionen wird in Corda mittels eines Blockchain-Protokolls und Kryptografie gewährleistet (Moothart & Escoto, 2018). Folgend werden nun die Schlüsselkonzepte von Corda vorgestellt, mit welchen die angesprochenen Probleme von mehreren Datenquellen gelöst werden sollen.

6.1.1 Netzwerk

Bevor auf die Datenebene eingegangen werden kann, muss zuerst das Netzwerk vorgestellt werden. Corda basiert auf einem Peer-to-Peer-Netzwerk bestehend aus Nodes (Netzwerkknoten), veranschaulicht in Abbildung 16. Nodes besitzen die Möglichkeit, direkt mit anderen Nodes zu kommunizieren, ohne dass weitere Nodes oder ein zentraler Server miteinbezogen werden muss (Moothart & Escoto, 2018). Der Zugang zum Netzwerk wird über Zertifikate (Doorman-Service) geregelt (Hearn, 2016). Hier unterscheidet sich Corda von öffentlichen Blockchains, in denen keine Restriktionen für den Beitritt bestehen (Newton, 2018). Mit dieser Massnahme zur Gewährleistung der Privacy innerhalb des Netzwerks steigt jedoch auch der benötigte Grad an Vertrauen gegenüber den Gegenparteien der Transaktion (Newton, 2018). Einerseits muss die Identität einer Partei bekannt sein, damit sie einem Corda-Netzwerk beitreten kann, andererseits muss darauf vertraut werden, dass digitale Vermögenswerte anderer Netzwerkteilnehmer auch physisch in der Realität existieren (Newton, 2018).

Corda basiert nicht wie klassische Blockchains auf einem Gossip-Netzwerk (Moothart & Escoto, 2018). Gossip bedeutet, dass Änderungen in der geteilten Datenbank an alle Netzwerkteilnehmer übermittelt werden (Hearn, 2016). Ein Corda-Netzwerk hingegen ist wie ein in Abbildung 16 visualisiertes Venn-Diagramm aufgebaut (Moothart & Escoto, 2018). Zugriff auf eine Transaktion und die darin enthaltenen Daten besitzen nur die Parteien, die an der Transaktion beteiligt sind. Im Venn-Diagramm wird dieses Verhalten durch die Schnittmenge der Parteien dargestellt. Existieren in Abbildung 16 also Transaktionen zwischen Alice und Bob, so repräsentieren die abgebildeten Tabellen die Datenbanken auf den jeweiligen Nodes. Ein Node kann zudem Daten in die Blockchain speichern, auf die er nur selbst Zugriff hat, wie bei Punkt 8 des Diagramms (Moothart & Escoto, 2018).

Möglich wird die Peer-to-Peer Kommunikation, da dem Initiator-Node die Netzwerk-Identitäten der Parteien einer Transaktion bekannt sind und eine direkte Verbindung aufgebaut werden kann (Newton, 2018). Als Vergleich: In einem Gossip-Netzwerk, in dem die Identitäten der Nodes unbekannt sind, muss ein neuer Zustand an alle Teilnehmer gesendet werden, damit sichergestellt werden kann, dass der gewünschte Node den Zustand erhält (Rilee, 2018). Dieses Konzept bringt Vor- und Nachteile gegenüber klassischen Blockchains (Newton, 2018). Ein Vorteil ist, dass die Privacy erhöht wird, weil die Leseberechtigung auf Transaktions- und nicht auf Netzwerkebene stattfindet. Weitere Vorteile entstehen dadurch, dass nicht alle Nodes die gesamte Blockchain speichern müssen und diese Methode sowohl weniger Speicherplatz benötigt als auch die Geschwindigkeit von Transaktionen erhöht (Newton, 2018). Nachteile ergeben sich bei der Datensicherheit und der Datenintegrität. Die reduzierte Datensicherheit entsteht aufgrund der geringeren Redundanz der Daten durch die transaktionsbasierte Speicherung (Newton, 2018). Die Datenintegrität verschlechtert sich, weil weniger Nodes angegriffen werden müssen, um eine Transaktion in der Datenbank zu ändern (Newton, 2018). Zudem sinkt mit einer hohen Privacy die Transparenz in einem Netzwerk (Newton, 2018).

6.1.2 CorDapp

CorDapps sind auf einem Node installierte Applikationen, welche die Funktionalitäten der verteilten Datenbank erweitern und festlegen, wie Transaktionen von digitalen Wertgegenständen (Assets) oder Nachrichten ausgeführt werden sollen (Gendal et al., 2016). Auf anderen Blockchain-Plattformen können CorDapps mit Smart Contracts verglichen werden. CorDapps verhalten sich wie Apps auf Smartphones. Ihre Funktionalitäten können auf den Nodes benutzt werden, auf denen sie installiert sind, sie müssen aber nicht auf jedem Node des Netzwerks installiert sein (Gendal et al., 2016). Hier zeigt sich ein weiterer Unterschied zu klassischen Blockchains, bei denen sich in der Blockchain befindende Smart Contracts, von allen Netzwerkteilnehmern ausgeführt werden können (Newton, 2018). Hauptsächlich bestehen CordApps aus States (Zustand), Contracs (Vertrag) und Flows (Fluss), die in den folgenden Absätzen weiter erläutert werden.

6.1.3 States

Ein State ist ein unveränderbares Objekt (Entität) von Daten, das mit einer anderen Partei geteilt werden soll (Moothart & Escoto, 2018). Das Objekt beschreibt den digitalen Wertgegenstand (Asset), mit dem über Transaktionen beliebige Operationen wie das Erstellen,

Transferieren oder Terminieren des Objekts ausgeführt werden können (Hearn, 2016). Für die Vorstellung von Corda soll ein Darlehensgeschäft zwischen Alice und Bob als Beispiel dienen, wobei der Darlehensvertrag den State darstellt. Vereinfacht umfasst der Vertrag einen Kreditgeber (Alice), einen Kreditnehmer (Bob), eine Darlehenssumme, ein Datum und einen Zinssatz. Zusätzlich soll im State gespeichert werden, wie viel vom Darlehen bereits zurückbezahlt wurde. Der eingegangene Vertrag ist in State 1 der Abbildung 17 dargestellt und wird in die Datenbank von Alice und Bob eingetragen.

Ein State kann nach der Erstellung nicht mehr verändert werden (Hearn, 2016). In unserem Beispiel muss jedoch, falls Bob einen Betrag des Darlehens zurückbezahlt, der Rückzahlungswert angepasst werden. Corda löst dies, indem ein neuer State erstellt, mit dem alten über eine Identifikationsnummer (Hash) verbunden und so eine Rückverfolgung gewährleistet wird (Hearn, 2016). Gültig ist immer nur der aktuelle State. Die bisherigen States werden als «historic» markiert, gezeigt in Abbildung 17. Die Entwicklung zu einem neuen State findet innerhalb einer Transaktion statt (Moothart & Escoto, 2018). Bob schlägt Alice also einen neuen aktualisierten State in Form einer Rückzahlung von CHF 20.- vor, der bei erfolgreicher Verifizierung durch den referenzierten Contract sowohl von Bob als auch von Alice signiert wird. Der neue State 2 wird daraufhin von allen beteiligten Nodes gespeichert (Moothart & Escoto, 2018). Wie ein Contract die Verifizierung durchführt, wird im nächsten Abschnitt behandelt.

6.1.4 Contracts

Ein Contract verifiziert einen State über den Programmcode (Gendal et al., 2016). Dabei werden beliebige Bedingungen geschrieben, auf die ein State innerhalb der Transaktion geprüft wird (Gendal et al., 2016). Beispiele sind, dass ein Zinssatz nicht höher 15% sein darf oder der Kreditgeber nicht gleichzeitig Kreditnehmer sein kann. Auch hier unterscheidet sich Corda zu anderen Blockchains, in denen ein Smart Contract festlegt, welche Aktionen für eine Transaktion wie durchgeführt werden (Moothart & Escoto, 2018). Zum Vergleich: In Corda wird in einer Transaktion von einer Partei ein neuer State der Datenbank vorgeschlagen und falls dieser den hinterlegten Bedingungen im Contract entspricht, werden die Datenbanken aktualisiert (Moothart & Escoto, 2018). Ein State muss daher immer auf einen Contract referenzieren, der ihn verifiziert (Gendal et al., 2016). Weiter kann ein Contract Legal Pros enthalten. Legal Pros sind klassische, rechtlich bindende Dokumente, die einer Transaktion angehängt werden können. Diese sollen in einem Rechtsstreit nachweisen, unter welchen rechtlichen Grundlagen eine Transaktion

vorgenommen wurde. Wie ein State und ein Contract zusammenhängen wird in Abbildung 18 aufgezeigt.

6.1.5 Transaktionen (Transactions)

Transaktionen beschreiben in Corda die Entwicklung von States (Hearn, 2016). Sie bilden einen Container, der keinen oder mehrere Input-States und keinen oder mehrere Output-States beinhaltet (Moothart & Escoto, 2018). In Bezug auf das Beispiel bedeutet dies, eine Transaktion kann eine Entstehung eines Darlehensvertrags (keinen Input-, einen Output-State), eine Tilgung (einen Input-, einen Output-State) oder eine Terminierung des Vertrags (einen Input-, keinen Output-State) darstellen. Transaktionen müssen einen oder mehrere Commands beinhalten, die beschreiben, um was für eine Transaktion es sich handelt. Über diese Kategorisierung können für die Transaktion spezifische Bedingungen programmiert werden, die gewährleisten, dass eine Transaktion wie gewünscht durchgeführt wird. Zusätzlich beinhaltet ein Command eine Liste der Parteien (Public Keys), welche diese Transaktion zu unterzeichnen haben. Neben Commands erlaubt Corda in Transaktionen Zeitfenster (Timeframes) einzubinden, die bestimmen, wann eine Transaktion stattfinden darf, und Anhänge (Attachements) in Form einer Hash-Referenz beizufügen. Eine schematische Transaktion ist in Abbildung 19 zu finden.

6.1.6 Flows

Mithilfe von Flows soll die Geschäftslogik in einem CorDapp integriert werden (Moothart & Escoto, 2018). Flows koordinieren, wie in einem Multischritt-Mehrparteienprozess Konsensus über einen neuen State erreicht wird, oder in anderen Worten, wie eine Transaktion durchgeführt wird (Hearn, 2016). Sie definieren, *wie*, mit *was*, mit *wem* und in *welcher Abfolge* die Parteien miteinander kommunizieren (Hearn, 2016). Der Primäre Anwendungsfall von Flows ist das Einholen der Signaturen sämtlicher in einer Transaktion beteiligten Parteien (Nodes) sowie dem Notary Node, wodurch eine Transaktion verifiziert wird. Flows können als Vorlage (SubFlow) gespeichert und wiederverwendet werden, sodass ein modularer Baukasten entsteht (Hearn, 2016). Ein Flow für das Beispiel wird in Abbildung 20 visualisiert. Mit der Behandlung von Flows wurden die wichtigsten Bestandteile und Konzepte eines CorDapps expliziert. Das ausprogrammierte CorDapp wird kompiliert und auf den ausgewählten Nodes installiert (Moothart & Escoto, 2018). Ein Konzept, welches noch aussteht, ist, wie die in einer Transaktion be-

teiligten Nodes autonom Konsensus über den neuen Zustand der Datenbank (Ledger) finden. In diesem Zusammenhang wird auch der bis jetzt unbekannte und in Abbildung 20 aufgeführte Notary Node vorgestellt.

6.1.7 Konsensus und Notary

Standardmässig wird Konsensus in Corda über zwei Konsensus-Verfahren erreicht (Gendal et al., 2016). Die beiden Verfahren werden in Corda Validity Consensus (Validitäts-Konsensus) und Uniqueness Consensus (Einzigartigkeits-Konsensus) genannt (Gendal et al., 2016). Validity Consensus wird über die bereits in Abschnitt 6.1.4 explizierten Contracts erreicht. Erfüllt die Transaktion die im Programmcode festgelegten Bedingungen, so ist sie valide (Gendal et al., 2016). Ausserdem wird geprüft, ob die notwendigen Parteien (Liste in Command) den neuen State signiert haben (Moothart & Escoto, 2018). Diese Verifizierung wird von allen Parteien durchgeführt und falls gewünscht auch vom Notary Node. Wichtig im Validity Consensus-Prozess ist, dass bei der Verifizierung alle Parteien das gleiche Ergebnis erhalten (Moothart & Escoto, 2018).

Notary Nodes werden primär für die Erstellung von Uniqueness Consensus benötigt (Gendal et al., 2016). Sie können aus einem einzelnen oder aus einem Set aus Nodes gebildet werden (Moothart & Escoto, 2018). Ausserdem können mehrere Notaries in einem Corda Netzwerk existieren (Gendal et al., 2016). Die Verteilung des Rechts zur Herstellung des Konsensus auf mehrere Parteien ist der Grund, weshalb Corda als Konsortium-Blockchain bezeichnet wird (Sandner, 2018). Ein Notary erstellt Uniqueness Consensus, indem er versichert, dass ein State nur einmal als Input-State einer Transaktion verwendet werden kann (Moothart & Escoto, 2018). Dazu speichert der Notary die Identifikationsnummer jedes Input-States einer zu signierenden Transaktion. Besitzt der Notary die Identifikationsnummer bereits, heisst dies, dass der State bereits verwendet wurde und die Transaktion wird untersagt (Gendal et al., 2016). Besteht die Transaktion beide Konsensus-Verfahren, übermittelt der Initiator Node der Transaktion den neuen State an alle beteiligten Parteien, welche ihr Abbild der Datenbank mit dem neuen State ergänzen (Gendal et al., 2016).

6.1.8 Nodes

Im letzten Abschnitt über Corda sollen die wichtigsten Bestandteile eines Nodes näher betrachtet werden. Die Architektur eines Corda-Nodes wird in Abbildung 21 gezeigt, in der die in dieser Arbeit behandelten Elemente rot markiert wurden. Das CorDapp wird über den Plugin-Service mit dem Node verbunden (installiert) und beinhaltet die programmierten Contracts, Flows und States. Damit lose gekoppelte Anwendungen mit einem Node kommunizieren können, unterstützen Nodes Remote Procedure Calls (RPC), die vom Corda-RPC-Ops Service entgegengenommen werden (Hearn, 2016). Dies ermöglicht im Darlehensbeispiel einer authentifizierten Web-Applikation, über ein Corda-Node einen Darlehensvertrag zu eröffnen und im Corda-Netzwerk zu speichern. Der Service-Hub bildet die Verbindung zu den Corda-Node-Services (Hearn, 2016). Auf den Service-Hub kann entweder über ein CorDapp oder mittels RPC Call zugegriffen werden (Hearn, 2016). Die wichtigsten Node-Services sind folgende:

- **Vault:** Im Vault werden die für den Node relevanten Output-States gespeichert. Über den Vault-Service können somit aktuelle Output-States ausgelesen werden, die für eine folgende Transaktion als Input-States genutzt werden können.
- **Identity-Mgmt:** Über den Identity-Management-Service wird die Identität des Nodes zugänglich gemacht. Die Identität eines Nodes in Corda kann für das Netzwerk sichtbar oder anonym sein. Ist sie anonym besteht sie lediglich aus einem Public Key (0A5B7…). Ist sie nicht anonym, besteht sie aus dem Public Key und einem Corda-X500-Name (z. B. O=Feldschlösschen, L=Rheinfelden, C=CH).
- **Network-Map:** Ein Node kann als Network-Map Service definiert werden. Ein Network-Map-Service stellt die Namen und die Public Keys der sich im Netzwerk befindlichen Nodes zur Verfügung.
- **Notary:** Dieser Service wird nur benötigt, wenn der Node als Notary dienen soll.
- **Messaging:** Über den Messaging-Service kann ein Node mit einem anderen kommunizieren.

Mit der Behandlung von Nodes wurden alle Schlüsselkonzepte von Corda thematisiert. In einem nächsten Schritt soll nun das Framework für die Node Clients und das Backend der Web-Applikation ausgeführt werden.

6.2 Spring-Boot

Das Spring-Framework und das Spring-Boot-Projekt werden in dieser Arbeit gleich mehrfach verwendet. Zum einen werden die Node Clients auf Basis einer Spring-Boot-Applikation erstellt, zum anderen das Backend der Web-Applikation. Die Einzelheiten und der Zweck der Artefakte werden in Kapitel 7 «Resultate» vorgestellt. Aus technologischer Sicht bildet das Spring-Framework einen wichtigen Bestandteil dieses Projekts und soll nun in diesem Kapitel näher erläutert werden.

Im Projekt werden Spring-Boot-Applikationen verwendet, um den Datenaustausch zwischen dem Benutzer und den Corda-Nodes über REST-APIs und RPC zu gewährleisten. Neben Spring existieren weitere Frameworks, mit welchen die benötigten Funktionalitäten umgesetzt werden können. Die wohl bekannteste Alternative für die API-Entwicklung ist Node.js (Hardy, 2016). Angesichts der folgenden Überlegungen wurde sich in diesem Projekt jedoch für Spring entschieden. Spring unterstützt alle JVM-basierten Programmiersprachen (Walls, 2016, S. 93). Da CorDapps in Kotlin oder Java geschrieben werden, besitzt Corda eine gute Spring-Integration. Ausserdem kann in einem Spring-Bean mit geringem Aufwand eine RPC-Verbindung zu einem Node aufgebaut werden. Auf dem Gedanken der Wiederverwendbarkeit bietet es sich folglich an, die API des Backends mit der gleichen Technologie zu entwickeln.

Spring bietet ein umfassendes Framework zum vereinfachten Bilden von JVM-basierten Applikationen und hat sich zu einem etablierten Standard entwickelt (Walls, 2016, S. 1). Eine der wichtigsten Eigenschaften von Spring ist «Dependency Injection», zu Deutsch Abhängigkeits-Injektion (Walls, 2016, S. 2). Dies bedeutet, dass eine Spring-Applikation nur die Komponenten (Libraries, Treiber etc.) besitzt, die im aktuellen Projekt benötigt werden (Johnson et al., 2004). Spring gilt deshalb als leichtgewichtige Alternative zu den Java-Enterprise-Edition-Applikationen, welche alle Funktionalitäten standardmässig mit sich bringen (Walls, 2016, S. 2).

Spring-Boot erleichtert das Bilden der Applikationen weiter (Gutierrez, 2017, S. 7). Mit Konzepten wie «opinionated technology» und «convention over configuration» kann sich auf das Programmieren der Funktionalitäten konzentriert werden, anstatt viel Zeit für Konfigurationen und repetitive Arbeiten aufzuwenden (Gutierrez, 2017, S. 7). Über Annotationen, Namensgebung und durch das Lesen der Applikationspfade versucht Spring-Boot die Intention hinter der Applikation zu «verstehen». Folgend werden Dienste wie der Appache-Tomcat-Server (Webserver) automatisch konfiguriert (Gutierrez, 2017, S.

7). Diese automatische Konfiguration ermöglicht es dem Programmierer, die Java-Applikation vorwiegend über POJOs (Plain-Old-Java-Objects) aufzubauen (Walls, 2016, S. 2).

Für den Prototyp wird einer der häufigsten Anwendungsfälle von Spring verwendet, dem Aufbau einer REST-API (Walls, 2016, S. 3). Eine REST-API wird für das Bilden einer losgekoppelten (stateless) Kommunikation mit einem Server benötigt (Zhou, Li, Luo & Chou, 2014). Losgekoppelt heisst, es besteht keine dauerhafte Verbindung zwischen Server und Client (Zhou et al., 2014). Der Client stellt eine Anfrage an den Server, welcher diese beantwortet und woraufhin die Verbindung endet. Ein Client kann jedoch beliebig viele Anfragen senden. Innerhalb einer «Rest-Controller-Klasse», dem zentralen Element der Spring-Web-Ebene, werden Endpunkte mittels Funktionen (Methoden) definiert und jeweils eine HTTP-Methode zugewiesen (Gutierrez, 2017, S. 20). Mögliche Methoden sind unter anderem die CRUD-Operationen (Create → POST, Read → GET, Update → PUT, Delete → DELETE). Ein Beispiel für einen Endpunkt ist «http://meine-url.ch/**users**». Ruft ein Benutzer den Endpunkt mit der entsprechenden HTTP-Methode auf, wird die zugehörige Funktion ausgeführt und eine JSON-Datei mit allen Benutzern zurückgegeben. Mit dieser Funktionalität soll im Prototyp die Kommunikation zwischen Frontend und Backend sowie zwischen Backend und Node Clients hergestellt werden.

Die zweite für den Prototyp benötigte Funktionalität, ist das Definieren einer Datenbank. Spring ermöglicht unter der Verwendung von Hibernate / JPA (einer Dependency) ein Object-Relational-Mapping (ORM) (Johnson et al., 2004, S. 400). Vereinfacht bedeutet dies, dass die Daten speziell annotierter Java-Objekte bidirektional in das Format für relationale Datenbanken umgewandelt werden können. Weil relationale Datenbanken immer noch eine der weit verbreitetsten Datenspeicherungsformen darstellen ist eine Transformation zum objektorientierten Aufbau von Java unerlässlich (Johnson et al., 2004, S. 400).

Resümiert wird Spring in diesem Projekt dazu verwendet, die Kommunikation zwischen den Applikationen (Komponenten des Prototyps) herzustellen und um zusätzliche Vertragsdaten in einer Datenbank zu verwalten. Die Datenbank selbst wird allerdings nicht von Spring zur Verfügung gestellt. Aus diesem Grund wird im nächsten Abschnitt die verwendete H2-Datenbank vorgestellt.

6.3 H2-Datenbank

H2 ist ein relationales Datenbank-Management-System geschrieben in Java (H2, o. J.). Die Datenbank wird in diesem Projekt verwendet, um erweiterte Daten (Meta-Daten) und Dokumente zu einem Vertrag zu persistieren. Das System ist ein Open-Source-Projekt und weist eine breite Kompatibilität auf (Kostrzewa, Bach, Brzeski & Werner, 2016). Zusätzlich verfügt die Datenbank über eine visuelle Webkonsole, in der SQL-Anweisungen ausgeführt werden können (Kostrzewa et al., 2016). Die H2-Datenbank unterstützt einen In-Memory- (existiert nur solange der Rechner in Betrieb ist) einen Disk- und einen Read-Only-Modus (Kostrzewa, Bach, Brzeski & Werner, 2016). In diesem Projekt soll der Disk-Modus verwendet werden. Dazu legt H2 eine Datei an, in der die Datenbank gespeichert wird und wodurch die Daten über einen Neustart hinaus für die Applikation bestehen bleiben. Durch ihre Leitgewichtigkeit, Geschwindigkeit und Unterstützung der JDBC-Schnittstelle (Standard für einen Datenbankzugriff via Java) weist H2 eine gute Kompatibilität zu Java als auch zu Spring auf (Kostrzewa et al., 2016). Die Engine der Datenbank kann mittels Spring-Dependency dem Projekt hinzugefügt werden und ist durch diese einfache Konfiguration, eine der meist verwendeten Datenbanken in Spring-Projekten (In 28 Minutes, 2017).

6.4 Apiary

Apiary[4] ist eine Web-Plattform von Oracle zum Designen, Prototypen, Dokumentieren und Testen von APIs (Apiary, o. J.). Die Plattform soll für die Mock-API von Moneyhouse verwendet werden, welche die Kreditwürdigkeit des Kunden als Resultat zurückgibt. Eine API auf Apiary wird in der Scriptsprache API Blueprint geschrieben, die an ein Gemisch aus Javascript und Markdown erinnert. Die Plattform ermöglicht in kurzer Zeit eine Mock-API mit Beispielswerten zu erstellen. Zusätzlich bietet sie den Service, die erstellte API auf einem Mock-Server zu veröffentlichen und so die Daten für das Backend des Prototyps mittels HTTP-Requests zugänglich zu machen.

[4] https://apiary.io/

6.5 Angular

Angular ist ein JavaScript-Framework zum Erstellen von reaktiven, einseitigen Web-Applikationen (Single-Page-Application – SPA) (Schwarzmüller, 2019). In Bezug auf das Projekt bedeutet dies, dass Angular zum Bilden des Frontends des Prototyps verwendet wird. Neben Angular existieren zahlreiche weitere Bibliotheken und Frameworks, wie React oder Vue, die den Programmierer mit Strukturvorgaben, vorgefertigten Funktionalitäten und der Möglichkeit zur Integration weiterer Bibliotheken unterstützen (Hamedani, 2018). Aufgrund der hohen Verfügbarkeit an Dokumentation, der weiten Verbreitung in der Praxis und den Funktionalitäten wie Routing oder Formular-Handling, fiel die Entscheidung auf Angular. Ein oft zitierter Nachteil von Angular ist der aufwendigere Lernprozess gegenüber einer Bibliothek wie React (Hamedani, 2018; Jarnjak, 2010). Vom Autor dieser Arbeit wurde es trotzdem als angemessen angesehen, die benötigten Fähigkeiten im Zuge dieser Arbeit zu erlernen.

In den folgenden Abschnitten sollen nun ausgewählte Konzepte von Angular vorgestellt werden. In Angular wird nicht jede Seite einzeln von einem Server gerendert und anschliessend an den Client (Browser) gesendet, sondern die gesamte Applikation besteht aus einer einzelnen Seite (Single-Page), deren Struktur (DOM) vom Browser mittels JavaScript angepasst wird (Schwarzmüller, 2019). Die in der Applikation benötigten Daten werden asynchron im Hintergrund geladen, sodass für den Benutzer keine Unterbrechungen erkennbar sind (Schwarzmüller, 2019).

Angular ist am ehesten nach dem Model-View-ViewModel-Muster (MVVM) von John Gossmann aufgebaut (Marx, 2016). In den Models des Musters werden die Daten bzw. die Datenstrukturen der Applikation festgelegt und können mit Daten-Klassen in Java verglichen werden (Marx, 2016; Jarnjak, 2010). Models besitzen keine direkte Verbindung zu den anderen beiden Bestandteilen des Musters und dürfen keine Logik enthalten (Marx, 2016). Angular-Applikationen werden in TypeScript, einer Erweiterung von JavaScript programmiert, wodurch Models über TypScirpt-Klassen definiert werden (Schwarzmüller, 2019). TypeScript ermöglicht, im Gegensatz zu JavaScript, Klassen, Interfaces und statische Typen anzulegen, wie es aus objektorientierten Programmiersprachen bekannt ist (Bierman, Abadi & Torgersen, 2014). Dies hat den Vorteil, dass Fehler im Scriptcode bereits während des Kompilierens und nicht erst zur Laufzeit erkannt werden (Bierman, et al., 2014). Durch das Kompilieren wird TypeScript-Code in JavaScript umgewandelt, welches vom Browser interpretiert werden kann (Schwarzmüller, 2019).

Komponenten stellen einen wichtigen Bestandteil ein Angular-Applikation dar (Schwarzmüller, 2019). Sie beinhalten sowohl die View, als auch das ViewModel des MVVM-Musters (Marx, 2016). Kern einer Komponente ist eine TypeScript-Datei mit einer `@Component`-annotierten Klasse (Schwarzmüller, 2019). In der Annotation wird die View der Komponente durch eine Referenz auf ein externes HTML-Template oder eine direkte HTML-Einbindung festgelegt (Schwarzmüller, 2019). Zusammen mit der zugehörigen CSS-Datei bildet das Template die sichtbaren Elemente der Applikation (Jarnjak, 2010). Die TypeScript-Klasse selbst repräsentiert das verbindende ViewModel des Musters (Jarnjak, 2010; Marx, 2016). In der Klasse wird die Logik der Komponente über Funktionen und Life-Cycle-Hooks (Funktionen die zu bestimmten Lebenszyklen der Komponente ausgeführt werden) integriert (Schwarzmüller, 2019). Als verbindend gilt das ViewModel deshalb, weil es die benötigten Daten bevorzugt als Model erhält, wenn nötig in den Funktionen transformiert und diese durch Data-Binding in der View dem Benutzer ausgibt (Jarnjak, 2010; Marx, 2016). Data-Binding bedeutet, dass aus der View (HTML-Template) direkt auf in dem ViewModel angelegte Variablen zugegriffen werden kann (Hamedani, 2018). Zusammengefasst bestehen Angular-Komponenten meistens aus drei Dateien: einem HTML-, einem CSS- und einem Typscript-Dokument (Schwarzmüller, 2019). Entscheidend ist, dass in einer Angular-Web-Applikation beliebig viele Komponenten existieren, die ausgehend von einer Root-Komponente, ineinander verschachtelt werden (Schwarzmüller, 2019). Zusätzlich können Daten-Modelle und Services ausgelagert werden, damit Funktionalitäten, Daten und Datenstrukturen über die Komponenten hinweg wiederverwendet werden können (Schwarzmüller, 2019).

Zuletzt bietet Angular verschiedene vorgefertigte Module, wie für die Implementation eines Seiten-Routings, für die Generierung von bidirektionalen und validierten Formularen und für das Handhaben von HTTP-Anfragen. Die Module werden dabei über ein zentrales Konfigurations-Dokument importiert (Hamedani, 2018). Bereits mit diesen Modulen können die meisten Use Cases von Applikationen abgedeckt werden. Angular bietet jedoch, wie Spring, die Möglichkeit von «Dependency Injections», über welche Module von externen Drittanbietern eingebunden werden können (Neuhaus, 2017).

7 Resultate

Dieses Kapitel beinhaltet die Vorstellung der erarbeiteten Artefakte. Die Ergebnisse werden wie im Kapitel Implementation in einem Bottom-up-Ansatz vorgestellt, beginnend mit der Corda-Blockchain. Zentrale Code-Ausschnitte der einzelnen Themenbereiche werden zusätzlich im Anhang aufgeführt und anhand von Kommentaren erklärt.

7.1 Corda-Blockchain

Die unterste Ebene des Prototyps bildet die Corda-Blockchain. Sie wird dazu benötigt, um die Verträge dezentral zu speichern, damit jede Vertragspartei eine Single-Version-of-Truth auf ihrem persönlichen Node besitzt. Ziel dieses Abschnitts ist es aufzugeigen, wie ein Corda-Netzwerk im Entwicklungs-Modus aufgebaut und wie ein Prototyp-CorDapp für das Management von Rückvergütungs- mit Darlehensverträgen implementiert werden kann. Der Prototyp trägt den Namen «Blocklaon» und ein Rückvergütungs-mit Darlehensvertrag wird im Projekt mit «ReimLoan» abgekürzt.

7.1.1 Set-up

Für die Entwicklung mit Corda werden nur wenige Installationen benötigt. Die wichtigste Voraussetzung ist das Java Development Kit (JDK), welches die Java Virtual Machine (JVM) zum Ausführen des Java-Bytecodes der Applikation zur Verfügung stellt (Corda, o. J.-f). Für das Build-Management des Projekts setzt Corda auf Gradle. Mit Gradle können die Projektstruktur, die Konfiguration und die benötigten Abhängigkeiten des Projekts festgelegt werden (Corda, o. J.-f). Wird das CorDapp in der Programmiersprache Kotlin geschrieben, empfiehlt Corda die Nutzung der Entwicklungsumgebung (IDE) Intellij. Sowohl Kotlin als auch Intellij stammen von den Entwicklern von JetBrains, weshalb die IDE eine gute Unterstützung bietet (Corda, o. J.-f). Mit diesen Installationen sind bereits alle Notwendigkeiten für einen Corda-Prototyp vorhanden.

Für das Aufsetzen des Projekts stellt Corda verschiedene Beispielsprojekte und Templates zu Verfügung, die über das Corda-GitHub-Profil[5] heruntergeladen werden können. Indem ein Template nach den persönlichen Bedürfnissen angepasst wird, kann ein neues, eigenes Corda-Projekt aufgesetzt werden. In dieser Arbeit wurde das Example-CorDapp-

[5] https://github.com/corda/samples

Projekt[6] von Dudley (2019) verwendet, weil es zum einen eine gute Dokumentation besitzt und zum anderen laufend mit den neusten Standards von Corda aktualisiert wird. Das Example-CorDapp-Projekt ist in drei Module eingeteilt: das Template für die Clients, die Contracts und die Workflows. Die Aufteilung von Contracts und States zu den restlichen Modulen wird vorgenommen, weil die JAR-Datei der Contracts an jede Transaktion angehängt wird, damit eine Verifizierung des States zu einem späteren Zeitpunkt wiederholt werden kann (Corda, o. J.-h). Deshalb soll das Modul so klein wie möglich gehalten werden. Die Inhalte dieser Module werden in den fortfolgenden Abschnitten ausgeführt.

7.1.2 Aufbau Netzwerk

Um ein lokales Corda-Netzwerk aufzubauen, wie es für den Prototyp verwendet wird, existieren verschiedene Möglichkeiten (Corda, o. J.-d). Die bereits im CorDapp-Example-Projekt vorkonfigurierte Methode verwendet das CordForm-Gradle-Plugin (Corda, o. J.-d). Mithilfe dieses Plugins können Nodes innerhalb eines Gradle-Tasks definiert und gebildet werden (Corda, o. J.-d). Ein Beispiel zu einer Node-Konfiguration befindet sich im Anhang unter Code-Block 1. Der Prototyp umfasst drei Nodes, einen für Feldschlösschen, einen für den Gastrobetrieb sowie den Notary Node. Das Plugin bildet die Nodes im Entwickler-Modus, in welchem für jeden Node die JAR-Datei, die Entwickler-Zertifikate, die Datenbank und weitere Node-Informationen generiert werden (Corda, o. J.-d).

Ein Node besitzt mehrere Zertifikate mit unterschiedlichen Funktionen und einem hierarchischen Aufbau, wodurch Zertifikate auf Netzwerk-, sowie auf Node-Ebene existieren (Corda, o. J.-d). Zertifikate werden unter anderem für das Signieren von JAR-Dateien und Transaktionen, für die Netzwerk-Zugangskontrolle und zur Sicherstellung einer eindeutigen Node-Entität (Corda-X-500-Name / Public Key) benötigt. Durch die Signaturen können weitere Bedingungen in den Contracts hinterlegt werden (z. B. gültige Versionen) oder es kann verhindert werden, dass auf Nodes unberechtigte CorDapps installiert werden (Corda, o. J.-d). Die Zertifikate bestehen aus Hash-Schlüsseln (Public und Private Keys), die in einem «Keystore» auf dem Node gespeichert werden (Corda, o. J.-d). In einem produktiven Netzwerk können die Zertifikate von einem Doorman-Service überprüft oder neue Zertifikate für Nodes ausgegeben und so die Integrität des Netzwerks gewahrt werden (Corda, o. J.-g). Zertifikate und Signaturen sichern das Netzwerk, bedeuten jedoch viele Umstände während der Entwicklung. Aus diesem Grund existiert in

[6] https://github.com/corda/samples/tree/release-V4/cordapp-example

Corda der Entwicklungs-Modus, in welchem jedem Node ein Standard-Keystore zugewiesen wird (Corda, o. J.-d). Dies bedeutet, dass bei Änderungen im Code (Contracts, State, Flow) lediglich der Gradle-Task neu ausgeführt werden kann und das aktualisierte CorDapp samt den Nodes neu gebildet wird (Corda, o. J.-d). Mit diesem Vorgehen muss sich während der Entwicklung nicht um Signaturen gekümmert werden. Die Nodes akzeptieren die Installation des CorDapps und sind dazu berechtigt, miteinander zu kommunizieren (Corda, o. J.-d).

Der Gradle-Task erstellt für alle Nodes ein Verzeichnis, welches unter anderem eine Node-Konfigurations-Datei enthält. In dieser Datei können Konfigurationen vom JDBC-Datenbankzugang über die erlaubte maximale Anhang-Dateigrösse bis zur zu verwendenden Konsensus-Methode festgelegt werden (Corda, o. J.-d). Damit diese vom Standard abweichenden Konfigurationen nicht nach jedem Bilden der Applikation ergänzt werden müssen, erlaubt der Gradle-Task das Hinzufügen einer Referenz auf eine Template-Konfigurations-Datei. Eine solche wurde für die Nodes von Feldschlösschen und des Gastrobetriebs verwendet, um den Datenbankzugang via H2-Webkonsole zu definieren. Zuletzt erstellt der Gradle-Task eine Batch-Datei, `runnodes.bat`, mit welcher die Nodes über ein Terminal gestartet werden können. Mit den lauffähigen Nodes ist das Netzwerk aufgesetzt und es kann sich folgend um das Erstellen des CorDapps gekümmert werden.

7.1.3 State und Schema

Der einzige State im Prototyp, `ReimLoanState.kt`, repräsentiert einen Rückvergütungs mit Darlehensvertrag und beinhaltet die notwendigen Informationen für den Abschluss eines solchen Vertrages. Die im State festgelegten Informationen sind in Tabelle 10 abgebildet. Im State wurden die Interfaces `LinearState`, `QueryableState` und `SchedulableState` implementiert. Das LinearState-Interface schreibt vor, dem State eine Linear-ID hinzuzufügen, über welche alle States während der Laufzeit eines Vertrages miteinander verbunden werden. Aufgrund des QueryableState-Interface muss ein Schema inkludiert werden, das es ermöglicht, eine spezifische Tabelle in der Node-Datenbank anzulegen (Corda, o. J.-a). Das Schema wird in einer separaten Kotlin-Klasse definiert und mit den aus der Java-Persistence-API (JPA) bekannten Annotationen wie `@Entity` ergänzt. Die Definition der Produkte-Entität wird in Code-Block 2 gezeigt. In Bezug auf den Prototyp konnten durch die Verwendung eines Schemas die gesamten Informationen aus Tabelle 10 zusammen mit den Beziehungen zu den Produkte- und Rückvergütungs-Entitäten in den Node-Datenbanken gespeichert werden. Die Struktur des Schemas wurde

im ERD in Abbildung 22 visualisiert. Der grösste Vorteil eines spezifischen State-Schemas in Verbindung mit einem `LinearState` entsteht durch die Möglichkeit, die in der Datenbank abgelegten States über die Linear-ID auszulesen. Ohne ein Schema würden lediglich die Transaktions-IDs und die Daten als Bytecode in der Node-Datenbank persistiert werden. Diese Funktionalitäten werden benötigt, um die aktuellen Verträge des gewünschten Nodes auszulesen oder um eine Historie eines Vertrages darlegen zu können. Das letzte Interface, `SchedulableState`, ordnet an, eine `nextScheduledActivity`-Funktion zu implementieren (Corda, o. J.-e). Die Funktion gibt eine `ScheduledActivity` zurück, die zur Instanziierung einen Flow und einen Zeitpunkt (Instant) benötigt. Unter der Verwendung des Terminierungs-Flows und des Enddatums kann im Prototyp somit die Beendigung des Vertrages automatisiert werden. Damit ein Flow automatisiert werden kann, muss er zusätzlich mit `@SchedulableFlow` annotiert sein (Corda, o. J.-e). Der Zeitpunkt und die Aktivität werden schlussendlich in die Datenbank der Nodes eingetragen.

Ein State wird ausschliesslich aus Read-only-Variablen aufgebaut. Dies bedeutet, dass ein State nach der Instanziierung nicht mehr verändert werden kann. Einige Anforderungen an den Prototyp, wie das Aktualisieren der Vertragswerte, verlangen jedoch eine «Entwicklung» des States. Dazu mussten im State Funktionen eingebunden werden, die eine Kopie des bisherigen State, zusammen mit den neuen Werten, erstellen. Ein Beispiel einer solchen Funktion, die zur Erfassung einer Tilgung dient, wird in Code-Block 3 abgebildet. Mit den Kopierfunktionen ist der State für den Rückvergütungs- mit Darlehensvertrag komplettiert und der Contract zur Validierung der States kann erstellt werden.

7.1.4 Contract und Commands

Im `ReimLoanContract.kt` werden die Transaktionen zur State-Änderung verifiziert. Ein Rückvergütungs- mit Darlehensvertrag kann über seine Laufzeit die folgenden Status annehmen: Offerte, Vertrags-Entwurf, Vertrag, Terminiert oder Gekündigt. Für jeden Übergang zu einem neuen Status und für jede Vertrag-Aktion innerhalb eines Status müssen eindeutige Bedingungen erfüllt werden, die im Contract festgelegt sind. Ein Contract kann beliebig viele Bedingungen enthalten, mit denen sichergestellt wird, dass der Vertragsinhalt (State) dem gewünschten Regelwerk entspricht sowie in Transaktionen ausschliesslich erlaubte Aktionen ausgeführt werden. In Corda besitzen alle Transaktionen einen Command, der beschreibt, um was für eine Transaktion es sich handelt. Die Commands werden als Interfaces innerhalb des Contracts festgelegt. Für den Prototyp wurden

die in Tabelle 11 aufgelisteten Commands erstellt. Mithilfe der Commands können die im Contract definierten Bedingungen kategorisiert werden. Das heisst, wird einer Transaktion der Command CreateOffer zugewiesen, werden nur die Bedingungen geprüft, die für das Erstellen einer Offerte erfüllt sein müssen. Ein Auszug an Bedingungen für den CreateOffer-Command wird in Code-Block 4 dargestellt. Nach der Fertigstellung des Contracts, ist nur noch ein Bestandteil des CorDapps ausstehend – die Flows.

7.1.5 Flows

Flows werden im Prototyp dazu verwendet, die Transaktionen über die Nodes hinweg durchzuführen. Dazu musste ein Flow für jeden in Tabelle 11 aufgeführten Command erstellt werden. Ein Flow kann in Kotlin als Singelton-Objekt definiert werden, welches eine Flow-Klasse für den Initiating- und den Accepting-Flow enthält. Dies bedeutet, dass ein Flow in der Regel zwei Seiten besitzt. Die Erste, die Initiator-Seite, nimmt ein Node ein, um eine Transaktion zu starten. Im Prototyp initialisiert beispielsweise der Feldschlösschen-Node eine Transaktion zum Erstellen einer Offerte. Die Annehmer-Seite (Acceptor) nehmen alle anderen Nodes ein, die in die Transaktion involviert sind. Dies ist der Gastrobetrieb, der die Offerte bestätigt. Zur Erhöhung der Übersichtlichkeit wurden die beiden Seiten in einem Kotlin-Objekt vereint, könnten jedoch auch getrennt voneinander bestehen. Die durch diese zweiseitige Flow-Architektur entstehende Flexibilität wird benötigt, um einem Flow zusätzliche Bedingungen hinzuzufügen, die über die allgemeinen Vertragsbedingungen des Contracts hinausgehen. Das heisst, auf jedem Node könnte ein individueller Flow installiert werden, welcher spezifische Bedingungen einer Partei beinhaltet. Für den Gastrobetrieb, der auf der Annehmer-Seite agiert, wurde innerhalb des Flow festgelegt, dass ein Vertrag nicht weniger als 3 Prozent Rückvergütung enthalten darf. Somit könnte zukünftig jeder Kunde seine Toleranzgrenzen im Programmcode festlegen, wodurch die Automation von Flows ermöglicht wird. Damit der richtige Acceptor-Flow ausgeführt wird, muss ihm die Initiating-Flow-Klasse innerhalb der Annotation `` `@InitiatedBy()` `` übergeben werden.

Die Initiator-Seite des Flows ist zuständig für den Aufbau der Transaktion und muss mit der Annotation `` `@InitiatingFlow` `` versehen werden. Soll der Flow mittels Client über RPC gestartet werden, muss zusätzlich die Annotation `` `@StartableByRPC` `` hinzugefügt werden. Die Flows im Prototyp beinhalten einen Tracker, mit welchem der Fortschritt des Flows sowohl auf der Node-Konsole ausgegeben als auch geloggt wird. In den meisten Flows des Prototyps besteht folgender Ablauf: Zu Beginn wird der aktuelle State des

Vertrags über den Vault-Service des Feldschlösschen-Nodes ausgelesen. Dieser dient als Input-State der Transaktion. Anschliessend wird über die in Abschnitt 7.1.3 vorgestellten Kopierfunktionen ein neuer State generiert, der als Ouptut-State der Transaktion dient. Durch die Verwendung eines `TransactionBuilder`-Objekts, wird die neue Transaktion, bestehend aus Input-State, Output-State sowie dem entsprechenden Command, zusammengesetzt und folgend verifiziert. Danach wird eine Session-Liste erstellt, welche die Identität (Public Key) des Gastrobetrieb-Nodes beinhaltet. Mit dieser Liste kann ein vordefinierter Sub-Flow aufgerufen werden, der die Signatur des Gastrobetriebs einholt. Dabei wird auf dem Gastrobetrieb-Node der entsprechende Acceptor-Flow ausgeführt. Wurde die Transaktion erfolgreich verifiziert und signiert, kann der letzte Schritt, das Einholen der Notary-Node-Signatur, gestartet werden. Auch dazu existiert ein vordefinierter Sub-Flow, da diese Funktionalität bei allen Transaktionen benötigt wird. Besteht die Transaktion die Konsensus-Verfahren des Notary, so wird die Transaktion erfolgreich in den Datenbanken der beteiligten Nodes eingetragen.

Als Zusatzfunktionalität wurden bei den Verträgen zum Erstellen oder Kündigen eines definitiven Vertrags zusätzliche HTTP-Requests (Anfragen) an das Backend implementiert, die überprüfen, ob beide Parteien den Vertrag oder die Kündigung im Frontend manuell signiert haben. Dadurch konnte bewiesen werden, dass aus Flows externe Services aufgerufen werden können, deren Antwort in einer Bedingung zur Verifikation einer Transaktion benutzt werden kann. Für einen Prototyp wurde das Backend allerdings nicht als Corda-Oracle-Service aufgebaut, sondern als übliche REST-API.

Mit den programmierten Flows wurde das CorDapp für den Prototyp fertiggestellt. Das erstellte CorDapp wird beim Ausführen des Gradle-Tasks gebildet und auf die Nodes geladen. Bis anhin können Flows nur über die jeweiligen Node-Konsolen gestartet werden. Allerdings sollen diese zu einem späteren Zeitpunkt mittels HTTP-Requests ausführbar sein, weshalb Node Clients benötigt werden.

7.2 Spring-Boot Node Clients

Die Node Clients stellen das letzte Modul des `blockloan`-Projekts dar. Die Clients basieren auf einer Spring-Boot-Applikation und können über die sich im Client-Modul befindende Gradle-Build-Datei gestartet werden. Eine Anforderung an den Prototyp ist, dass sowohl die Verträge des Feldschlösschen- als auch die des Gastrobetrieb-Nodes ausgelesen werden können. Aus diesem Grund benötigen beide Nodes einen Client. Damit für den Prototyp nicht zwei Clients programmiert werden müssen, werden die Clients auf unterschiedlichen Ports und mit unterschiedlichen RPC-Zugängen, jedoch unter der Verwendung derselben Spring-Boot-Applikation gestartet. Die Clients beider Nodes besitzen daher die gleichen Funktionalitäten. In einem produktiven Projekt müsste ein separater Client für den Ausgeber Feldschlösschen und die Kunden erstellt werden. Unter der im Prototyp verwendeten Lösung wäre es möglich, dass über den Gastrobetrieb-Client versucht wird, einen neuen Vertrag mit dem Gastrobetrieb als Aussteller zu erstellen. Der Vertrag würde jedoch aufgrund von Bedingungen im Contract des CorDapps nicht verifiziert werden, weshalb diese Methode kein Risiko darstellt.

Eine zentrale Aufgabe der Node Clients ist das Herstellen einer RPC-Verbindung zum entsprechenden Node. Die Instanziierung der Verbindung wird in Code-Block 5 dargestellt. Eine zweite wichtige Funktionalität des Clients ist das Bereitstellen einer API, über welche Transaktionen auf den Nodes gestartet werden können. Zudem muss es über die API möglich sein, die sich auf dem Node befindenden Verträge auszulesen. Um diese Anforderungen zu erfüllen wurde ein Spring `RestController` mit den zugehörigen `RequestMapping`-Methoden implementiert. Die resultierten API-Endpunkte mit ihren HTTP-Methoden und Beschreibungen sind in Tabelle 12 aufgelistet.

Ein weiterer Bestandteil der Clients sind der `ControllerService.kt` und die Helfer-Klassen. Diese werden zur bidirektionalen Umwandlung der aus den HTTP-Requests erhaltenen JSON-Objekte in die korrekten Corda-Objekte benötigt. Die Transformationen sind notwendig, weil die im `ReimLoanState` enthaltenen Java-Datums-Typen nicht korrekt aus den nach deutschem Datumsformat übergebenen Strings erstellt werden können.

Mit der vollendeten Programmierung der Clients ist der Corda-Blockchain-Anteil des Prototyps abgeschlossen. Für die Nutzung der Blockchain über eine grafisch unterstützte und benutzerfreundliche Oberfläche wird zusätzlich eine Web-Applikation benötigt, die in den nachfolgenden Abschnitten erläutert wird.

7.3 Spring-Boot Backend

Das Backend bildet die Integrationsebene des Prototyps. Über dieses werden die Node Clients und somit die Blockchain, die Backend-Datenbank für zusätzliche Vertragsinformationen sowie Drittservices, in diesem Fall die Mock-Moneyhouse-API, angesteuert. Sowohl die über das Backend zugänglich gemachten Daten als auch die Funktionalitäten der Node Clients werden vom Backend via REST-API zur Verfügung gestellt. Für das Backend wurde mittels Maven ein separates Java-Spring-Boot-Projekt angelegt. Zusätzlich wurden im Projekt die Abhängigkeiten zur Spring-JPA-Unterstützung und der Treiber für die H2-Datenbank inkludiert.

Die Datenbank des Backends wird dazu benötigt, zusätzliche Vertragsinformationen, die nicht in der Blockchain gespeichert werden sollen, zu persistieren. Zum einen sind dies Meta-Vertragsdaten, zum anderen Dokumente, die der Kunde als Nachweis zu erbringen hat. Wird im Frontend ein neuer Vertrag erstellt, so muss das übergebene Vertragsobjekt vom Backend aufgeteilt werden. Dokumente und Meta-Vertragsdaten werden in der Backend-Datenbank, die Vertragsinformationen in der Blockchain gespeichert. Beim Auslesen der Verträge werden beide Teile über die in der Blockchain generierte Linear-ID zusammengeführt und als Gesamtobjekt dem Frontend zurückgegeben. Die Attribute der Vertrag-Meta-Entität der Backend-Datenbank sind in Tabelle 13 abgebildet.

Der Bytecode der Dokumente wird nicht direkt in der Blockchain abgespeichert, da basierend auf dem dezentralen Grundkonzept einer Blockchain, die Daten auf jedem Node redundant gehalten werden. Corda besitzt in dieser Hinsicht den Vorteil, dass Transaktionsdaten nur auf den in der Transaktion beteiligten Nodes gespeichert werden. Trotzdem ist es laut Hearn (2016) nicht empfohlen, grosse Datenmengen in der Blockchain zu speichern. Attachments, wie sie in Corda genannt werden, sind dazu gedacht, wiederverwendbare Dateien auf einem Node zu speichern, die einer Transaktion über den Hash-Wert der Datei angehängt werden (Hearn, 2016, S. 18). Beispiele sind gleichbleibende Vertragsbedingungen oder ein Ferienkalender (Hearn, 2016, S. 18). Im Prototyp, in welchem nur die beiden Dokumente Ausweiskopie und Steuererklärung implementiert wurden, wäre es technologisch möglich gewesen, die Dokumente als ZIP-Datei auf die Blockchain hochzuladen (Hearn, 2016, S. 18). Für das Produktiv-System mit sehr vielen Dokumenten zu einem Vertrag ist dies jedoch nicht die angemessene Lösung. Aus diesen Gründen wurde ein anderer Ansatz gewählt. Der Bytecode der Dokumente soll in der Backend-

Datenbank und nicht in der Blockchain gespeichert werden. Dem Vertrag auf der Blockchain werden dennoch die Hash-Werte der Dokumente hinzugefügt. Mit diesen kann zu einem späteren Zeitpunkt überprüft werden, ob die Dokumente in der Datenbank verändert wurden und somit die Integrität des Vertrages gewahrt werden.

Die API des Backends wurde, wie für die Node Clients, durch einen Spring `RestController` erstellt. Auch weist der Controller die fast gleichen Endpoints wie die Clients auf. Unterschiede bestehen für das Auslesen oder Eintragen der Meta-Boolean-Werte (eine Variable, die nur wahr oder falsch annehmen kann). Beispielsweise kann über die Werte ausgelesen werden, ob der Vertrag vom Kunden bereits signiert wurde. Die verfügbaren Endpoints des Backends und deren Beschreibung sind in Tabelle 14 aufgeführt. Obwohl für einige Funktionalitäten die auf der Blockchain ausgeführt werden, keine weiteren Daten aus dem Backend benötigt werden und somit direkt aus dem Frontend auf die Node Clients zugegriffen werden könnte, wurde immer der Weg über das Backend gewählt. Diese Architektur einer zentralen Integrationsstelle hat den Vorteil, dass die Anbindung von übergelagerten Anwendungen zentral gesteuert werden kann. Bei Änderungen in der Ansteuerung von Drittservices müssen diese folglich nur einmal angepasst werden. Zudem wird das Backend durch die Bereitstellung der Dienste via API komplett vom Frontend losgelöst, wodurch es als Service vielseitig einsetzbar ist.

Um in den `RequestMapping`-Methoden gemeinsam benötigte Funktionalitäten auszulagern, wurde ein `ReimLoanService.kt` erstellt. Mit dem Service können die Transformationen zwischen den verschiedenen Vertragsobjekten des Frontends, des Backends und der Blockchain bidirektional vorgenommen werden. Ausserdem wird der Service dazu verwendet, die auf den Nodes vorhandenen Verträge auszulesen. Die Entscheidung, von welchem Node die Verträge ausgelesen werden sollen, wird anhand des hinterlegten TCP-Ports (zukünftig Host-Adresse) des im Frontend angemeldeten Benutzers sowie seiner Rolle getroffen. Hiermit verfügt das Backend über alle definierten Anforderungen und das letzte erstellte Artefakt, das Frontend, kann vorgestellt werden.

7.4 Angular Frontend

Das Frontend bildet den visuell sichtbaren Anteil des Prototyps. Ziel des Frontends ist es, dem Benutzer eine grafische Oberfläche in Form einer Web-Applikation zu bieten, über die der Benutzer mit der Blockchain interagieren kann. Als Grundlage der Entwicklung dienten die in Kapitel 5.3 vorgestellten Mock-ups, die bis auf marginale Designanpassungen in der Applikation umgesetzt werden konnten. Die ausgestalteten Ansichten werden in den Abbildungen 23 bis 30 illustriert.

Den Benutzern des Prototyps wurden verschiedene Rollen zugewiesen, welche die zukünftigen Akteure im System repräsentieren sollen. Demzufolge wurden Benutzer für die in Abbildung 4 dargestellten internen Beteiligten von Feldschlösschen und den Gastrobetrieb mit gleichnamigen Rollen definiert. Mit dem persönlichen Login können sich die Nutzer in der Applikation anmelden (Abbildung 23) und werden auf das ihrer Rolle entsprechende Dashboard weitergeleitet (Abbildungen 24, 25). Der Sales Manager und der Vorgesetzte besitzen auf dem Dashboard die Aktion zum Erfassen eines neuen Vertrags. Über ein Eingabeformular können die Nutzer alle relevanten Daten für einen Rückvergütungs- mit Darlehensvertrag aufnehmen und anschliessend eine Offerte in der Blockchain generieren lassen. Sobald die Offerte in der Blockchain verfügbar ist, werden die Akteure Sales Manager und Gastrobetrieb vom System gebeten, der Offerte die Nachweisdokumente beizufügen. Die übrigen Vertragsinformationen können weiterhin aktualisiert werden (Abbildung 26). Wurden alle Nachweise korrekt erfasst, kann der Sales Manager aus der Offerte einen Vertrags-Entwurf erstellen. Bevor nun ein definitiver Vertrag von den beteiligten Parteien unterzeichnet werden kann, muss der Kunde vom Backoffice freigegeben werden. Zur Überprüfung ist es dem Backoffice möglich, die gesamte Vertragsentstehung über die Historie des Vertrags zurückzuverfolgen (Abbildung 27).

Nach der Freigabe des Kunden ist es beiden Parteien erlaubt, den Vertrag zu signieren oder weitere Verhandlungen zu führen. Werden währenddessen Nachweise verändert, muss der Kunde vom Backoffice erneut geprüft werden. Unterzeichnet eine Partei den Vertrag, kann er nicht mehr geändert werden. Der definitive Vertrag wird jedoch erst in der Blockchain persistiert, wenn beide Parteien signiert haben. Diese Vorgehensweise hat den Grund, dass während der Verifizierung eines definitiven Vertrags, aus der Blockchain beide Signaturen mittels HTTP-Request zum Backend überprüft werden. Würde versucht werden, bereits nach der ersten Signatur einen Vertrag zu generieren, schlägt die Verifizierung in der Blockchain fehl. Zuletzt wird überprüft, ob sich der zu unterzeichnende

Vertrag ausserhalb der Goldenen Regeln befindet. Falls ja, besitzt der Sales Managers nicht die ausreichenden Rechte und der Vorgesetzte hat den Vertrag zu signieren.

Nach erfolgreicher Signatur besteht im System ein definitiver Vertrag, der nicht mehr verändert werden kann und allen Benutzern in der Vertrags-Übersicht angezeigt wird (Abbildung 29). Sowohl Sales Manager als auch der Gastrobetrieb verfügen zu diesem Zeitpunkt ausschliesslich über die Funktion den Vertrag zu kündigen. Der Kündigungsprozess weist exakt dasselbe Vorgehen wie die Signierung des Vertrags auf. Das heisst, beide Parteien müssen die Vertragskündigung signieren, erst dann wird die Änderung in der Blockchain vorgenommen. Das Backoffice hingegen ist berechtigt, Tilgungen oder Rückvergütungen zu erfassen, die einen festen Bestandteil des Vertrags darstellen und in die Blockchain eingetragen werden (Abbildung 28).

Grundsätzlich wird ein Vertrag automatisch mit dem Erreichen des Enddatums terminiert. Da sowohl der Rückvergütungs- als auch der Darlehensvertrag ein Enddatum besitzen, wird das Enddatum gewählt, welches weiter in der Zukunft liegt. Damit wird sichergestellt, dass entweder Rückvergütungen oder Tilgungen weiterhin erfasst werden können. Für Demonstrationszwecke wurde im Prototyp dennoch eine Terminierungsfunktion in der Vertragsübersicht des Backoffice eingebaut. Nach der Terminierung können keine Funktionen mit dem Vertrag mehr ausgeführt werden und der Vertrag kann ausschliesslich über die Kunden-Übersicht aufgerufen und konsultiert werden (Abbildung 30).

Zuletzt finden die Akteure auf ihren spezifischen Dashboards Shortcuts für ausstehende Arbeiten. Sales Manager und Vorgesetzter können sich alle Verträge anzeigen lassen, die sie zu signieren haben und das Backoffice die, bei welchen eine Kundenüberprüfung vorgenommen werden muss. Die Funktion hat den Zweck, den Mitarbeitern von Feldschlösschen als Aufgabenliste zu dienen.

Mit der Vorstellung der Aufgabenliste wurden die wichtigsten Funktionalitäten des Frontends präsentiert. Auch wurden mit dem Abschluss des Frontends sämtliche Bestandteile des Prototyps umgesetzt und vorgestellt. Unter der Betrachtung des in Abbildung 1 dargestellten Vorgehensprozesses endet nach der Implementation und dem Testen des Prototyps der produktive Teil dieser Arbeit. Im nachfolgenden Kapitel, der Diskussion, soll nun der Prototyp validiert werden. Zudem sollen in der Diskussion die Voraussetzungen für die Beantwortung der Forschungsfragen sowie die während der Projektumsetzung gemachten Erfahrungen angesprochen und reflektiert werden.

8 Diskussion

In diesem Kapitel wird die Arbeit kritisch gewürdigt. Es soll aufgezeigt werden, in welchem Umfang die im Prototyp verwendeten Technologien, zusammen mit den erstellten Artefakten, die Problemstellungen aus Kapitel 1.2 lösen oder verbessern. Darüber hinaus werden die Umsetzbarkeit des Prototyps zu einem Produktiv-System bewertet und die Voraussetzungen zur Beantwortung der Forschungsfragen geschaffen. Aufgrund der Neuartigkeit der Blockchain-Technologie sollen ausserdem Erkenntnisse während der Programmierung und zur Vorgehensweise geteilt werden.

8.1 Vorgehensweise während der Programmierung

In der Umsetzung wurde versucht, sich so nahe wie möglich an die in Kapitel 3 beschriebene Methodik zu halten. Allerdings war die Planung während der Programmierung nicht immer umsetzbar. Wie in Kapitel 3 beschrieben, sollten die Funktionalitäten des Prototyps in fertige Arbeitspakete zerlegt und iterativ mittels Kanban-System abgearbeitet werden. Bereits zu Beginn der Programmierarbeiten wurde jedoch festgestellt, dass es aufgrund eines einzelnen Projektmitarbeiters wenige Vorteile hat, mit dieser Zerlegung zu arbeiten, bevor nicht gewisse Grundstrukturen des Projekts erstellt wurden. Aus diesem Grund wurde in einem ersten Schritt das gesamte Corda-Projekt vollständig aufgebaut, bevor mit den Node Clients weitergearbeitet wurde. Auch das Backend- und Frontend-Projekt wurden bis zur Implementation der Vertrags-Funktionalitäten aufgesetzt. Erst danach konnte zur iterativen Methodik übergegangen werden.

In Iterationen zerlegt und durchgeführt wurden primär die von den Corda-Flows bereitgestellten Funktionalitäten. Diese konnten Schritt für Schritt über Clients, Backend und Frontend implementiert werden. Das heisst, eine neue Funktion, wie das Erstellen einer Offerte, wurde zuerst der Node Client API hinzugefügt, dann im Backend eingebettet und zuletzt das dazugehörige Formular im Frontend erstellt. Auf diese Weise konnte der iterative Charakter der Implementation doch noch beibehalten werden.

Eine der wichtigsten Erkenntnisse bezüglich der Umsetzung des Prototyps ist, dass die Vertrags-Datenmodelle bereits zu Beginn akkurat geplant werden müssen. Da die Rückvergütungs- und Darlehensverträge das zentrale Element der Applikation darstellen, müssen bei Änderungen im Datenmodell aufwendige Nachbesserungen in sämtlichen Bestandteilen des Prototyps unternommen werden. Aus diesem Grund tut man gut daran,

die Datenmodelle ausgiebig zu planen und detailliert mit dem Praxispartner zu besprechen, um diesen Aufwand zu vermeiden.

8.2 Erkenntnisse aus der Entwicklung mit Corda

Ein Ziel dieser Arbeit ist, erste Erfahrungen mit einer Blockchain-Technologie zu sammeln und mit der Umsetzung eines Prototyps die Möglichkeiten und Limitierungen der Technologie bewerten zu können. Aus diesem Grund wird es vom Autor dieser Arbeit als sinnvoll angesehen, gemachte Erfahrungen mit Corda zu teilen.

Zu Beginn der Blockchain-Evaluation musste festgestellt werden, dass Corda keine reine Blockchain darstellt, sondern eine Distributed-Ledger-Technology (DLT), die Konzepte und Funktionalitäten einer Blockchain beinhaltet. Die Unterschiede wurden in Abschnitt 4.4 eingeführt. Dies hat zu Folge, dass sich die Konzepte von Corda stärker von anderen Blockchains unterscheiden, wie die von öffentlichen Blockchains untereinander. Um mit Corda einen Prototyp zu entwickeln, müssen deshalb spezifische Konzepte für den Aufbau eines Corda-Netzwerks und das Entwickeln von CorDapps erlernt werden. Dennoch greift Corda auf viele bekannte Konzepte und Technologien, wie relationale Datenbanken zurück, welche die Arbeit erleichtern. So besitzt der Programmierer durch den direkten Zugriff auf die Node-Datenbank via Konsole eine hohe Transparenz darüber, *was* und *wie* etwas auf einem Node gespeichert wird.

Der markanteste Unterschied zu öffentlichen Blockchain-Technologien entsteht daher, dass einem Node die Teilnehmer des Netzwerks (Peers) bekannt sind. Der Benutzer oder die Web-Applikation müssen demnach die Adresse (Public Key) der Gegenparteien nicht aufbewahren, um eine Transaktion mit diesen auszuführen. Es reicht aus, die Peers des Netzwerks über den eigenen Node-Map-Service auszulesen und die in der Transaktion beteiligten Parteien auszuwählen. Die gesamte Verwaltung von Public und Private Key einer Organisation wird vom Node selbst übernommen. Dem Programmierer ist es somit möglich, eine Applikation zu schreiben, die sich in der Bedienung nicht von heutigen unterscheidet. Zudem mussten im Backend des Prototyps weder Public noch Private Keys zum Auslesen und Erstellen von Transaktionen gespeichert werden. Diese Möglichkeit zur Gestaltung üblicher Software könnte ein wesentlicher Beitrag zur Adaption von Corda leisten.

Obwohl Corda erst seit 2016 auf dem Markt ist, besitzt das Projekt bereits eine gut ausgestaltete Dokumentation[7]. Zusätzlich stellt Corda Lernvideos auf den bekannten Plattformen YouTube, Vimeo und Udemy zur Verfügung, die einem beim Erlernen der Konzepte unterstützen. Für das Erstellen einer ersten Anwendung, welche nahe an den Basiskonzepten von Corda liegt, bestehen ausserdem diverse Beispielsprojekte und Prototypen, bei denen einfache Implementationen der Konzepte konsultiert werden können. Beachtet werden müssen jedoch die unterschiedlichen Corda-Versionen, da viele Funktionalitäten der Version 3 zur aktuellen Version 4 überholt wurden. Schwierigkeiten entstehen weiter, wenn spezifische Fragestellungen während der Programmierung auftreten. Gerade weil das Projekt noch nicht lange auf dem Markt ist, existieren nur wenige Blogbeiträge zu Problembehandlungen und alternativen Lösungswegen. Zudem behandeln Blogbeiträge meist Implementationen, die sich im Entwicklungsmodus von Corda befinden. Für ein zukünftiges Live-System, indem produktive Zertifikate vergeben und kontrolliert sowie Aktualisierungen von CorDapps signiert werden müssen, besteht zurzeit nur wenig Hilfsmaterial. Auch in Slack[8], dem Chat-Tool der Corda-Community, existieren grosse Know-how-Differenzen zwischen denen, welche die Plattform weiterentwickeln und denen, welche die Technologie ausprobieren. Im Zwischenraum, dem produktiven Bereich, fehlen noch viele Unterstützer. Zuletzt konnten nur wenige aktuelle Live-Applikationen, wie das Darlehens-CorDapp von Finastra, gefunden werden und ausserdem unterliegt die Plattform einer schnellen Weiterentwicklung, wodurch grössere Veränderungen nicht ausgeschlossen werden können (Morris, 2018). Die Erkenntnisse zeigen, dass sich Corda noch zu Beginn des Entwicklungsstadiums befindet und man sich dieser Situation bei der Evaluation eines produktiven Projekts bewusst sein muss.

Die letzte Erfahrung mit Corda ist, dass ein Netzwerk im Entwicklermodus, im Vergleich zu anderen privaten Blockchains, relativ schnell aufgebaut werden kann, das Deployment einer neuen Version jedoch zeitintensiv ist. Zwar kann als Abhilfe die Test-API[9] von Corda zum Schreiben von JUnits-Tests verwendet werden, allerdings ist der Aufwand für das Erstellen dieser Tests für sämtliche Contract-Bedingungen und Flows auch nicht zu unterschätzen. In diesem Projekt waren eine genaue Arbeitsweise und Code-Überprüfungen vor dem Deployment deshalb unerlässlich. Ausserdem war es hilfreich, sich zuerst

[7] https://docs.corda.net/
[8] https://cordaledger.slack.com
[9] https://docs.corda.net/api-testing.html

ausgiebig mit dem Entwickeln von CorDapps zu befassen, bevor ausprobiert wurde. Dazu wurde der von Corda empfohlene Kurs «Introduction to Corda Development[10]» auf Udemy durchgearbeitet.

8.3 Verifizierung Prototyp

Die Verifizierung des Prototyps wird über zwei Stufen vorgenommen. Auf der ersten Stufe werden die Ergebnisse des Prototyps mit den Anforderungen aus Kapitel 5.1 verglichen. Auf der zweiten Stufe wird der Prototyp bezüglich der in Kapitel 1.2 aufgelisteten Problemstellungen zusammen mit einem Vertreter von Feldschlösschen bewertet[11]. Weil der Prototyp in diesem Stadium der Entwicklung nicht als Live-System eingesetzt werden konnte, wurde sich für eine Verifizierung mittels Expertenbefragung entschieden. Die Ergebnisse werden folgend vorgestellt.

Im Prototyp konnten alle in Kapitel 5.1 vorgegebenen Anforderungen umgesetzt werden. Die Funktionalitäten wurden getestet und in einer Demonstration dem Praxispartner vorgestellt. Darüber hinaus bildet der Prototyp auch den in Absatz 5.2 definierten Soll-Prozess ab. Demzufolge konnte der Prototyp aus technischer Sicht gemäss Planung realisiert und kann durch das Aufweisen der definierten Funktionalitäten auf erster Stufe verifiziert werden.

Für die Verifizierung der zweiten Stufe sollen die Problemstellungen aus Kapitel 1.2 hinzugezogen und unter Verwendung des Prototyps neu bewertet werden. Die in der Problemstellung ausgeführten Herausforderungen bezüglich Erweiterbarkeit und Sicherheit werden aufgrund ihrer Wichtigkeit in den separaten Abschnitten 8.4 und 8.5 diskutiert.

Den grössten Vorteil des im Prototyp umgesetzten Rückvergütung- und Darlehensvertrags-Management-Systems sieht Feldschlösschen in der Reduktion der administrativen Arbeit sowie in einem systemunterstützten, standardisierten Prozess. Mit der Integration des Soll-Prozesses ist das Backoffice ausschliesslich für die Genehmigung des Kunden zuständig. Im Gegensatz zum Ist-Zustand, in dem das Backoffice die Nachweisdokumente beschaffen, die Vertragsdokumente erstellen und die physischen Verträge verwalten muss, bedeutet die neue Methode eine erhebliche Aufwandsreduktion.

[10] https://www.udemy.com/corda-development/
[11] S. Schorno & M. Weiss, Präsentation des Prototyps, 8. Mai 2019

Die Handhabung der Vertragsinformationen und Dokumente wird durch den Prototyp für alle Beteiligten vereinfacht, indem alle Informationen einem bestimmten Vertrag zugewiesen werden und an einem Ort zugänglich sind. Da alle Mitwirkenden, das heisst, Kunde, Sales Manager, Vorgesetzter und Backoffice, den Zugriff auf sämtliche Informationen des Vertrags besitzen, wird die Transparenz in der Vertragsentstehung erhöht und der Mailverkehr zwischen den Parteien beseitigt. In einem Folgeprojekt könnte der Prototyp durch eine Benachrichtigungsfunktion ergänzt werden, welche die Kommunikation innerhalb der Applikation ermöglicht und zusätzlich zu einem Vertrag speichert. Doch bereits die im Prototyp implementierte Ansicht der Historie eines Vertrags, erfüllt die Forderungen von Feldschlösschen nach mehr Transparenz und Rückverfolgbarkeit. Welche Vertragsinformationen und Fortschritte in einem produktiven System in der Blockchain oder in der Backend-Datenbank gespeichert werden, muss allerdings weiter evaluiert werden. Trotzdem zeigt die Architektur des Vertrages, wie die Rückverfolgbarkeit für ein Rückvergütungs- und Darlehensvertrag in Corda implementiert werden kann.

Die bisher diskutierten Funktionalitäten des Prototyps verbessern den Vertragsentstehungsprozess von Feldschlösschen, doch wären diese auch ohne die Nutzung einer Blockchain umsetzbar. Nun sollen deshalb die Vorteile der Corda-Blockchain gegenüber der in der praktischen Relevanz in Kapitel 1.3 vorgestellten Lösungen aufgezeigt werden.

Der wohl wichtigste Vorzug entsteht durch die Speicherung des immer exakt gleichen Abbilds des Vertragsobjekts auf allen beteiligten Nodes, sodass eine einzig gültige Version des Vertrags besteht. Mit dem Besitz eines eigenen Nodes, auf dem der unveränderbare Vertrag gespeichert wird, verfügt jede Partei über eine sichere Version des Vertrags. Für einen Vertragsabschluss muss ausserdem kein höherer Grad an Vertrauen vorhanden sein, als dass die Identität der Gegenpartei bekannt sein muss. Anders als bei anderen digitalen Vermögenswerten in einer privaten Blockchain, wie Währungen oder materielle Güter, sind die Parteien nicht auf die Wiederverwendbarkeit des Werts angewiesen, sondern müssen den Vertrag lediglich aufbewahren. Das Misstrauensverhältnis in der Geschäftsbeziehung zwischen den Parteien wird folglich nicht von der Blockchain, sondern vom Vertrag gemildert, welcher vor rechtlicher Instanz geltend gemacht werden kann. Aufgrund dieser Funktionalität löst der Prototyp die ausstehende Problemstellung, die von den in Abschnitt 1.3 aufgeführten bisherigen Systemen nicht erfüllt werden konnte. Ausstehend ist jedoch, ob Verträge in einer Blockchain auch in der Schweiz rechtskräftig sind. Diese Thematik wird unter Punkt 8.6 dieses Kapitels diskutiert.

Dank einer einfachen Umsetzbarkeit hat die Verwendung von Corda weitere Vorteile: In Corda-Contracts kann ein zentrales, gemeinsam anerkanntes Basisregelwerk für Rückvergütungs- mit Darlehensvertrag festgelegt werden. Unabhängig mit welchem System (ausgeberseitige oder kundenseitige Applikationen) ein Vertrag erstellt wird – die festgelegten Bedingungen können nicht umgangen werden. Nebst in Contracts können Bedingungen innerhalb von Flows bestimmt werden, die parteispezifische Kriterien zur Annahme eines Vertrags ermöglichen. Die Festlegung von Bedingungen wurde in Corda einfach gelöst und die Funktionalitäten zur Überprüfung werden vorgefertigt mitgeliefert. In üblichen Backend-Systemen müssten diese selbst eingebunden werden und ausserdem wäre eine Überprüfung der Bedingungen aller Parteien ohne ein dezentrales System in diesem Umfang nicht möglich.

Corda besitzt Mechanismen, mit welchen Transaktionen automatisiert werden können. Um diese Funktionalität zu verifizieren, wird im Prototyp der Flow für das Terminieren eines Vertrags zeitbasiert beim Vertragsende ausgeführt. Die zeitlichen Auslöser werden in der Node-Datenbank abgelegt und können über die H2-Konsole überwacht werden. Darüber hinaus ist das Konzept der Flows ein Differenzierungsmerkmal gegenüber öffentlichen Blockchains, in denen die Business-Logik in einer eigens erstellten Workflow-Engine aufgebaut werden muss. Dies bedeutet, dass die Abfolge der Aufrufe von Smart Contracts nicht auf der Blockchain-Ebene festgelegt und damit die korrekte Prozessabfolge nicht sichergestellt werden kann. In Corda hingegen lassen sich über Flows zum einen standardisierte Geschäftsprozesse in der Blockchain hinterlegen, zum anderen bietet Corda ein mitgeliefertes Framework, mit welchem die Prozesse automatisiert werden können. Zudem lassen sich durch die Verschachtelung der wiederverwendbaren Flows selbst komplexe Geschäftsprozesse im System abbilden.

Mit den Erkenntnissen aus dem Terminierungs-Flow wurde zusammen mit Feldschlösschen der Rückschluss gezogen, dass ein Kunde mithilfe von akkurat festgelegten Bedingungen in Contracts und automatisierten Flows selbstständig einen Vertrag eröffnen könnte. Zuerst werden die Vertragsbedingungen aufgrund der vom Kunden eingegebenen Werte berechnet. Bestätigt der Kunde über die Signatur sein Einverständnis, wird der Vertrag von der Blockchain geprüft und von Seiten Feldschlösschen automatisch signiert. Zurzeit müsste der Kunde lediglich vom Backoffice aufgrund der hochgeladenen Dokumente geprüft werden. Durch immer fortschrittlichere Bilderkennungssysteme und Machine Learning, könnte jedoch auch der Prüfvorgang immer weiter automatisiert werden.

Zusammenfassend hat der Prototyp gezeigt, dass die Single-Version-of-Truth eines Vertrags in einem Blockchain-System umsetzbar ist. Ausserdem konnten mit dem Prototyp Verbesserungsmöglichkeiten in Bezug auf Transparenz, Effizienz und Automatisierung aufgezeigt werden. Der Prototyp kann deshalb auch auf zweiter Stufe zusammen mit Feldschlösschen verifiziert werden. Für die Transformation des Corda-Prototyps zu einem produktiven System besteht bezüglich Erweiterbarkeit, Sicherheit und Rechtsfragen allerdings weiterer Abklärungsbedarf, der in den folgenden Abschnitten diskutiert werden soll.

8.4 Erweiterbarkeit des Systems

Die Erweiterbarkeit wird aus technischer Hinsicht als grösste Herausforderung für ein produktives System bewertet. Feldschlösschen besitzt mehrere tausend Rückvergütungs- und Darlehensverträge mit Gastrokunden. Aus diesem Grund muss in einem Folgeprojekt in Zusammenarbeit mit den Gastrobetrieben eruiert werden, ob die Betreibung eines Nodes für jeden Kunden machbar ist. Eine weitere Herausforderung besteht darin, dass Gastrobetriebe meist Kleinstunternehmen darstellen, die keine eigene Informatik bzw. keinen Webserver besitzen, auf dem ein Node gestartet werden könnte. Dies bedeutet, dass, falls Feldschlösschen an der Anforderung einer privaten Blockchain festhält, einen Dienst für die Gastrobetriebe zur Verfügung stellen muss, auf dem ein Node gestartet werden kann. Ob sich das für meist nur einen Vertrag mit einem Gastrobetrieb finanziell auszahlt, muss in Gegenüberstellung zu den aktuellen Mehrkosten der Administration abgewogen werden. Corda unterstützt den Aufbau von Netzwerken auf den Amazon-AWS- und Microsoft Azure-Cloud-Plattformen (Corda, o. J.-b; Corda, o. J.-c). Die Nutzung von Cloud-Servern wäre somit eine Möglichkeit, die Nodes für die Gastrobetriebe bereitzustellen. In einem Forumsbeitrag stellt Mike Hearn, Lead Platform Engineer von R3, ausserdem in Aussicht, dass die Community an einer Lösung arbeitet, die mehrere Identitäten auf einem Node zulässt (Hearn, 2017). Dies würde bedeuten, dass nicht jeder Gastrobetrieb einen eigenen Node benötigt, sondern dass sich Gastrobetriebe oder Verbände einen Node teilen könnten. Die Notwendigkeit eines Nodes für jeden Kunden ist im Use Case von Feldschlösschen sicherlich ein Nachteil einer privaten Blockchain-Lösung und limitiert die Erweiterbarkeit des Systems.

8.5 Sicherheit

Die Sicherheit von Corda ist schwierig zu beurteilen, da kaum Erfahrungsberichte ausserhalb des Testnets und in produktiven Systemen vorliegen (Höfelmann & Sander, 2019). Corda verfügt über aktuelle Sicherheitsmechanismen und unterstützt die Regulatorien der Finanzbranche. Aufgrund der hohen verfügbaren Privacy in Corda wird der Datenschutz verglichen mit anderen Blockchain-Plattformen als positiv eingestuft (Höfelmann & Sander, 2019). Aus diesem Grund entschied sich auch die Schweizer Börse SIX für den Aufbau ihrer digitalen Börse die Corda-Plattform zu verwenden (Jochum, 2019). Die Untersuchungen von Höfelmann und Sander (2019) ergaben, dass sich private Blockchains in punkto Sicherheit kaum voneinander unterscheiden. Alessandro De Carli, Gründer des Blockchain-Start-up Papers, gab in einem Interview (A. De Carli, persönliche Kommunikation, 23. März 2019) jedoch zu bedenken, dass private gegenüber öffentlichen Blockchains einer viel geringeren Belastung ausgesetzt sind. Öffentlich zugängliche Blockchains wie Ethereum würden durch den offenen Quellcode und Hackerangriffe täglich auf ihre Sicherheit geprüft und daher stetig weiterentwickelt. Damit habe Ethereum bewiesen, dass die Plattform eine hohe Sicherheit aufweise und sei deshalb tendenziell als sicherer einzustufen als private oder Konsortium-Blockchains wie Corda. Diese Aussage deckt sich auch mit den Untersuchungen von Höfelmann und Sander (2019). Aus diesen geht hervor, dass keine der in dieser Arbeit miteinander verglichenen privaten Blockchains eine vergleichbare Belastbarkeit der Sicherheitspakete wie Ethereum vorweisen kann (Höfelmann & Sander, 2019). Entscheidet sich Feldschlösschen für einen Einsatz von Corda, sollten deshalb erste Erfahrungsberichte aufmerksam verfolgt werden.

8.6 Rechtliche Fragestellungen

Die letzte in dieser Arbeit angesprochene Limitierung der Blockchain-Technologie stellen ungeklärte Rechtsfragen dar. Folgend werden die unternommenen Recherchen bezüglich der rechtlichen Situation des Prototyps vorgestellt. Diese können jedoch nicht abschliessend qualifiziert werden. Aus diesem Grund möchte Feldschlösschen in einem allfälligen Folgeprojekt einen Rechtsdienst beauftragen, der die rechtlichen Fragestellungen für einen blockchain-basierten Rückvergütungs- und Darlehensvertrag aufklärt.

In der Strategie «Digitale Schweiz» des Bundes wurden Massnahmen definiert, welche die rechtlichen Rahmenbedingungen der Blockchain-Technologie klären sollen (EFD, 2018). Zusätzlich hat der Bundesrat zum Thema Blockchain / DLT einen Bericht veröffentlicht, welcher eine Auslegeordnung relevanter rechtlicher Rahmenbedingungen bieten und den Handlungsbedarf klären soll (Eidgenössischer Bundesrat, 2018, S. 8). In diesem Bericht proklamiert der Bundesrat, dass er die Voraussetzungen verbessern und die Rahmenverbindungen dazu schaffen möchte, damit sich die Schweiz zu einem führenden Standort für Fintech- und Blockchain-Unternehmen etablieren kann (Eidgenössischer Bundesrat, 2018, S. 8). Weiter führt der Bundesrat aus, dass die Schweiz offen gegenüber blockchain-basierten Geschäftsmodellen sei und der Schweizer Rechtsrahmen bereits heute dazu geeignet ist (Eidgenössischer Bundesrat, 2018, S. 8).

Laut Bericht würde ein Rückvergütungs- oder Darlehensvertrag unter die zweite Kategorie von Token fallen, welche Forderungen, Mitgliedschaften und dingliches Recht abbilden (Eidgenössischer Bundesrat, 2018, S. 67). Dem Fazit dieser Kategorie ist zu entnehmen, dass die meisten rechtlichen Grundlagen aus den bisherigen Gesetzen übernommen werden können (Eidgenössischer Bundesrat, 2018, S. 67). Als Ausgangspunkt für einen Vertrag in einer Blockchain gilt daher auch die Form- und Inhaltsfreiheit. Laut dem Konsumkreditgesetz (KKG) benötigt es für ein Darlehen einen Vertrag der schriftlichen Form und einem minimalen obligaten Inhalt. Unter der Verwendung von Legal Pros bzw. Attachments, könnte dieser Anforderung in Corda nachgekommen werden. Dem Bericht können weitere Diskussionen zu den rechtlichen Rahmenbedingungen für die Signatur, den Übertrag eines Vertrags und die Gültigkeit der Blockchain zur Aufbewahrung eines Wertpapiers entnommen werden. Diese lassen jedoch keine endgültige Schlussfolgerung zwischen Lehre und Praxis zu, da noch keine Gerichtsentscheide zu Rechtsfällen vorliegen (Eidgenössischer Bundesrat, 2018, S. 50, 63, 67, 77).

Im Rechtssystem bestehen somit Gesetzeslücken, die vor dem Einsatz eines solchen Systems abgeklärt werden müssen. Dies würde aber über den Umfang dieser Arbeit hinausgehen. Der Autor dieser Arbeit ist jedoch der Ansicht, dass mit der Intention des Bundes zur Schaffung der gesetzlichen Rahmenbedingungen, zusammen mit der Entscheidung von SIX, eine digitale Börse basierend auf Corda zu entwickeln, eine Umsetzung eines Corda-Vertragsmanagementsystems in der Schweiz aus rechtlicher Sicht in naher Zukunft möglich wäre.

9 Schlussfolgerung

In dieser Arbeit wurde untersucht, wie aktuelle Herausforderungen im Rückvergütungs- und Darlehensvertragsmanagement der grössten Schweizer Brauerei unter dem Einsatz der Blockchain gelöst werden können. In der Schlussfolgerung werden nun die erarbeiteten Erkenntnisse dazu verwendet, die zu Beginn gestellten Forschungsfragen zu beantworten und um Handlungsempfehlungen an die Feldschlösschen Getränke AG abgeben zu können, welche kritischen Faktoren in einem Folgeprojekt beachtet werden müssen.

In einer ersten basislegenden Literaturrecherche wurde analysiert, welche Differenzierungsmerkmale zwischen einer Supply Chain und einem weiter gefassten Business-Ökosystem bestehen und mit welchen Herausforderungen Unternehmen sich in einem solchen Umfeld konfrontiert sehen. Zudem wurde die Eignung der Blockchain-Technologie zur Bildung der verbindenden Plattform eines Business-Ökosystems evaluiert, indem erste Praxisbeispiele aufgezeigt und bewertet wurden. Die Recherche hat gezeigt, dass die Blockchain-Technologie einen vielversprechenden Ansatz für die zukünftige Implementation von Business-Ökosystemen darstellt und neue digitale Geschäftsmodelle ermöglicht. Mit der Opportunität zur Verbesserung der Transparenz, des sicheren Informationsaustauschs und der Rückverfolgbarkeit in Supply Chains und Business-Ökosystemen können aktuelle Forderungen auf der Kunden- sowie Anbieterseite besser erfüllt werden. Insbesondere hat sich herausgestellt, dass die Blockchain aufgrund der Reduzierung des zu einer Geschäftsbeziehung benötigten Vertrauens, eine Unterstützung zur Implementation von lose gekoppelten Beziehungen zwischen Business-Ökosystem-Teilnehmern bietet. Letztendlich hat die Recherche ergeben, dass sich durch den Einsatz von Smart Contracts Regeln für Geschäftsbeziehungen definieren lassen, wodurch eine autonome Governance-Struktur für ein Business-Ökosystem geschaffen werden kann.

In der Analysephase wurden die aus der Literaturrecherche gewonnenen Prozesse zur Transformation einer Supply Chain zu einem Business-Ökosystem angewandt. Folglich konnte im Umfeld des Praxispartners eine Untersuchung möglicher Teilnehmer eines Business-Ökosystems sowie deren Kategorisierung durchgeführt werden. Die Analyse zeigt, dass sich im Stakeholder-Umfeld von Feldschlösschen keine gesamtheitliche Plattform zur Integration sämtlicher Stakeholder umsetzen lässt. Dennoch konnten für einzelne Stakeholder-Fragmente, potenzielle Einsatzgebiete für blockchain-basierte Plattformen identifiziert und präsentiert werden.

Ein weiterer wichtiger Bestandteil der Analysephase war die Ausarbeitung des Use Cases, welcher als Grundlage für den in dieser Arbeit entwickelten Proof-of-Concept-Prototyp diente. Auf die Auswahl des Use Cases, dem Rückvergütung- und Darlehensvertragsmanagement von Feldschlösschen, folgte die Analyse des Ist-Prozesses. Dabei konnten einige Problemstellungen festgestellt und effizienzsteigernde Massnahmen für den Soll-Prozess ergriffen werden. Durch die Implementation des Prototyps in Verbindung mit der vorangegangenen Diskussion wurden nun die Voraussetzungen geschaffen, um die in Kapitel 1.5 gestellten Forschungsfragen zu beantworten.

9.1 Beantwortung der Forschungsfragen

- **Forschungsfrage 1:** *Welche Blockchain-Plattform soll für das Rückvergütungs- und Darlehensvertragsmanagement von Feldschlösschen eingesetzt werden? Welche Möglichkeiten, Herausforderungen und Limitierungen entstehen durch diese Wahl?*

Unter der Anwendung des in Abbildung 5 erstellten Evaluationsmodells und den von Feldschlösschen gestellten Anforderungen konnten private oder Konsortium- (permissioned) Blockchains als erforderlicher Typ für das Rückvergütungs- und Darlehensvertragsmanagement eruiert werden. Mit diesem Ergebnis stellte sich in der folgenden Plattformevaluation Corda als geeignet heraus. Als Hauptgründe dieser Entscheidung wurden die Eignung von Corda für Finanzgeschäfte, die Möglichkeit rechtlich bindende Vereinbarungen einer Transaktion hinzuzufügen, die finanzmarktrelevante Community, vorhandene Referenzprojekte und eine gute Dokumentation angesehen. Die Affinität und Verpflichtung von Corda eine Plattform für den Finanzmarkt und seine Regulatorien zu entwickeln, führten gegenüber der Konkurrenz zu unterschiedlichen Konzepten im Aufbau der DLT. Deshalb wird aus Sicht des Autors dieser Arbeit die Wahrscheinlichkeit als höher eingestuft, dass mit Corda zeitnah eine produktive Lösung im Vertragsmanagement entwickelt werden kann, die dem Schweizer Rechtssystem entspricht.

Weitere Vorteile von Corda für das Vertragsmanagement werden in den Flows, für die Implementation des Business-Layers und der Privacy auf Transaktionsebene gesehen. Die grösste Limitierung entsteht durch die Anzahl an benötigten Nodes in einer produktiven Lösung. Da Gastrobetriebe meist nicht in der Lage sind, ein eigenen Node zu hosten, müsste nach einer kosteneffizienten Lösung für deren Bereitstellung gesucht werden. Corda stellt ausserdem eine Abhilfe in Aussicht, dass ein einzelner Node mehrere Identitäten besitzen kann, wodurch mehrere Gastrobetriebe auf einem Node zusammengefasst

werden könnten. Eine weitere Limitierung wird aufgrund momentan bestehender Gesetzeslücken gesehen. Der Eidgenössische Bundesrat hat jedoch angekündigt, diese Lücken in naher Zukunft zu schliessen. Zuletzt führt die höhere Integrität des Systems auch zu höheren Aufwänden in der Erstellung der Applikation sowie in der Betreibung des Netzwerks, die gegenüber den heutigen Mehrkosten der Vertragsverwaltung zu vergleichen sind.

- **Forschungsfrage 2:** *In welchem Mass ist es bereits heute möglich, einzelne Elemente während der Vertragsentstehung und Vertragsverwaltung in einer Blockchain zu automatisieren? Welche Vorteile hat ein digital geführter Vertrag?*

Durch die Implementation des Prototyps konnte erwiesen werden, dass unter der Anwendung von Blockchain-Plattformen wie Corda, sich bereits einzelne Prozessschritte der Vertragsentstehung und -verwaltung automatisieren lassen. Dabei können nicht nur zeit- oder konditionsbedingte Aufgaben automatisiert werden, sondern auch Entscheidungen.

Grundlage für die Automation in Blockchains bilden die in Smart Contract festgelegten Bedingungen, die eine Aktion ausführen, sobald diese erfüllt sind. Bedingungen können einen beliebigen Komplexitätsgrad aufweisen, jedoch müssen sie in Programmcode abbildbar sein. Mit der im Prototyp umgesetzten Funktion des automatisierten Terminierens eines Vertrages konnte ein erstes Einsatzgebiet einer zeitbasierten Aufgabenausführung veranschaulicht werden. Eine künstliche Intelligenz im Sinne von «Unsupervised Machine Learning» kann allerdings für Entscheidungen nicht verwendet werden. Dies ist aber auch nicht zwingend notwendig. Eine denkbare Bedingung in einem Smart Contract kann auch der Entscheid eines externen Services (in Corda Oracle-Service) darstellen, der über solche Funktionalitäten verfügt. Für eine Blockchain ist zur Überprüfung einer Bedingung nur ausschlaggebend, dass sie beim Senden des gleichen Inputs den immer gleichen Output erhält.

In Bezug auf das Vertragsmanagement von Feldschlösschen bedeutet dies, dass folgende Elemente im Prozess der Vertragsentstehung und -verwaltung automatisierbar sind. Basierend auf hinterlegten Konditionen kann zu den Eingaben des Gastrobetriebs automatisch eine Offerte generiert werden. Entspricht der Vertrag den in Contract und Flows definierten Bedingungen von Feldschlösschen, wäre das System in der Lage, den Vertrag automatisch zu signieren. Über eine Schnittstelle könnten die aus dem Vertrag resultie-

renden Workflows und Zahlungsströme dem ERP-System übergeben werden. Eine anschliessende zyklische Überprüfung des ERP-Systems könnte Zahlungen im Vertrag automatisch nachführen, die dadurch nicht vom Backoffice eingetragen werden müssen. In einem Zukunftsszenario ist sogar denkbar, dass die Nachweisdokumente von einem externen Oracle-Service mittels Machine Learning und Bilderkennung ausgewertet und die Kundenberechtigung damit immer weiter automatisiert werden könnte. Einzig die kundenseitige Erfassung der persönlichen Informationen und Nachweisdokumente müssten auch in Zukunft vom Gastrobetrieb oder Sales Manager durchgeführt werden.

Eine wichtige Erkenntnis aus der Geschäftsprozessintegration ist jedoch, dass der Aufwand für eine vollständige Automatisierung den relativen Nutzen oft übersteigt. Auch in Zukunft werden Spezialfälle existieren, die entweder vom Backoffice geprüft oder von einem Vorgesetzten signiert werden müssen. Ausserdem wäre das Backoffice noch immer für die Behandlung von Rechts- sowie Betreibungsfälle verantwortlich. Eine erhebliche Verbesserung würde aber bereits entstehen, wenn ein Grossteil der Standardfälle automatisiert werden könnte – und Blockchain-Plattformen wie Corda bieten die Möglichkeit dazu.

Die Diskussion des vorherigen Kapitels hat gezeigt, dass die Grundlage zu jeglichem im Prototyp aufgezeigten Nutzen digitale Verträge bilden. Blockchain-basierte Verträge ermöglichen die Haltung einer einzig gültigen Vertragsversion für beide Parteien, sie führen zu Effizienzsteigerungen im Prozess, zu mehr Transparenz und einer höheren Rückverfolgbarkeit. Der Prototyp hat dadurch die Fähigkeit der Blockchain-Technologie aufgezeigt, die in Problemstellung aufgeführten ungelösten Herausforderungen der Wirtschaft besser zu lösen, als bisherige Systeme.

- **Forschungsfrage 3:** *Wie kann durch den Einsatz der Blockchain-Technologie die Transparenz der laufenden und vergangenen Verträge erhöht und dadurch eine Rückverfolgbarkeit auf Kundenebene ermöglicht werden?*

Daten in einer Blockchain sind unveränderbar. Im Prototyp wurde ein Ansatz gewählt, in welchem für einen Vertrag nur ein einziges Vertragsobjekt über die gesamte Laufzeit erstellt wird. Dieses eine Vertragsobjekt wird während der Entstehung und der Vertragslaufzeit über den Status entwickelt. Zu Beginn der Laufzeit ist es eine Offerte, zum Schluss ein terminierter Vertrag. Weil ein Vertrag unveränderlich ist, muss bei Veränderungen eine Kopie des Objekts erstellt werden, die mit dem Vorgänger fest verbunden

wird. Aufgrund dieser linearen Verbindung entsteht für einen Vertrag eine unveränderbare, kettenartige Struktur, über welche die gesamte Historie eines Vertrages ausgelesen werden kann. Im Gegensatz zum heutigen Vertragsmanagement von Feldschlösschen, kann durch diese integre Aufzeichnung der Vertragsentwicklung die Transparenz während dem Vertragslebenszyklus erhöht werden. Zusätzlich können zu einem digitalen Blockchain-Vertrag beliebige Informationen wie Nachrichten, Zahlungen oder Fortschrittspunkte gespeichert werden, die spätere Rückschlüsse über den Vertrag zulassen. Diese Transparenz und Aufzeichnung würde Feldschlösschen auch in Rechtsfällen zugutekommen, wenn sämtliche Informationen eines Vertrags jederzeit und mit einfachem Zugang zur Verfügung ständen.

Die Identität eines Nodes ist fest mit dem Kunden verbunden. Durch die nahtlose unveränderbare Aufzeichnung der Vertragsentwicklung in der Blockchain gehen keine Kunden- und Vertragsinformationen mehr verloren. Selbst wenn ein Vertrag auf einen neuen Pächter der Gastwirtschaft übertragen wird, kann die ursprüngliche Vertragspartei über die historischen Zustände des Vertrags ermittelt werden. In der bestehenden Lösung von Feldschlösschen gehen die Informationen des ursprünglichen Kunden verloren, sobald der neue Pächter den Vertrag übernimmt. Schlussendlich hat der Prototyp gezeigt, dass mittels Blockchain-Technologie eine transparente Rückverfolgung der Verträge auf Kundenebene ermöglicht werden kann.

9.2 Handlungsempfehlungen

Zur Erfüllung des letzten Ziels dieser Arbeit werden in diesem Abschnitt Handlungsempfehlungen für ein mögliches Folgeprojekt an Feldschlösschen abgegeben.

Ein Projekt zur Verbesserung der Prozesse im Rückvergütung- und Darlehensvertragsmanagement der Feldschlösschen Getränke AG wird vom Autor dieser Arbeit empfohlen. Bereits während der Ausarbeitung des Soll-Prozesses des Prototyps konnten einige Potenziale erkannt werden, welche die Transparenz und Effizienz erhöhen und die Arbeit der Mitarbeitenden erleichtern können. Auch wenn sich gegen eine Blockchain und für eine herkömmliche Datenbank entschieden wird, sollte ein System entwickelt werden, dass den Informationsfluss zu einem Vertrag vereinheitlicht regelt und verhindert, dass Informationen zu Kunden oder historischen Verträgen verlorengehen. Die Entscheidung gegen eine Blockchain würde allerdings bedeuten, dass eine Single-Version-of-Truth eines Vertrages nicht umsetzbar wäre. Eine Blockchain-Lösung ist unumgänglich, wenn

nicht auf ein vertrauensgestütztes System oder eine vertrauenswürdige Drittpartei zurückgegriffen werden soll.

Wird sich für eine Blockchain-Lösung entschieden, müssen dennoch weitere Abklärungen getroffen werden. Die grösste Limitierung des Prototyps besteht durch die in Kapitel 8.4 angesprochene Erweiterbarkeit des Systems. Deshalb sollte die Anzahl der zu bereitstellenden Nodes, die technologische Umsetzbarkeit der Bereitstellung und die Kosten dieser Lösung evaluiert werden. Angesichts des Umfangs eines Folgeprojekts und durch fehlende Referenzprojekte wird geraten, einen Prototyp in gleichem Format mit einer öffentlichen Blockchain-Plattform umzusetzen. Auch wenn die Auswertung des Blockchain-Typs dieser Arbeit eine private Blockchain-Lösung ergeben hat, ist es von Wert, die Vor- und Nachteile an einem subjektiven Beispiel gegenüberzustellen. Mit dieser Vorgehensweise könnte ermittelt werden, ob eine öffentliche Blockchain für das Vertragsmanagement wirklich eine zu tiefe Privacy bietet und ausserdem könnten Aufwand und Kosten beider Lösungen abgeschätzt werden. Diese Anstrengungen in der Planung würden zu hohen Kosteneinsparnissen in einem Folgeprojekt führen. Zudem hätte Feldschlösschen die Möglichkeit, anhand eines internen Anwendungsbeispiels Erfahrungen mit Blockchain-Anwendungen zu sammeln und die Erkenntnisse für weitere Evaluationen zu nutzen.

Zuletzt müssen die rechtlichen Rahmenbedingungen für einen blockchain-basierten Vertrag von einem Rechtsexperten geprüft werden. Bevor mit der Umsetzung des Systems begonnen werden kann, muss absolut sichergestellt werden, dass Verträge in einer Blockchain, ohne die Aufbewahrung einer papierbasierten Version, rechtlich ausreichend sind.

Mit den rechtlichen Abklärungen wurden die wichtigsten Handlungsempfehlungen abgegeben und es kann mit Spannung erwartet werden, ob ein blockchain-basiertes Folgeprojekt im Vertragsmanagement der Feldschlösschen Getränke AG zustande kommt.

Literaturverzeichnis

Aagesen, G. & Krogstie, J. (2010). Analysis and Design of Business Processes Using BPMN. In J. vom Brocke & M. Rosemann (Hrsg.), *Handbook on Business Process Management 1: Introduction, Methods, and Information Systems* (S. 213–235). Berlin, Heidelberg: Springer.

Abeyratne, S. A. & Monfared, R. P. (2016). Blockchain ready manufacturing supply chain using distributed ledger. *International Journal of Research in Engineering*, 9(5), 1–9.

Accenture. (o. J.). Blockchain for Contracts, Technology for Digital Contracting. Abgerufen von https://www.accenture.com/no-en/success-blockchain-contracts

Ade, V. (2018, 2. Oktober). Grossbanken bringen Blockchain-Plattform an den Start. *Finanz und Wirtschaft*. Abgerufen von https://www.fuw.ch

Aitken, R. (2017, 26. Juni). Accord Project's Consortium Launching First Legal „Smart Contracts" With Hyperledger. *Forbes*. Abgerufen von https://www.forbes.com

Al-Tahat, M. D., Dalalah, D. & Barghash, M. A. (2012). Dynamic programming model for multi-stage single-product Kanban-controlled serial production line. *Journal of Intelligent Manufacturing*, *23*(1), 37–48.

Almeida, S., Albuquerque, A. & Silva, A. (2019). An Approach to Develop Software that Uses Blockchain. In R. Silhavy (Hrsg.), *Software Engineering and Algorithms in Intelligent Systems* (S. 346–355). Cham: Springer International Publishing.

Apiary. (o. J.). Fast-track your API Design. Abgerufen von https://apiary.io/how-apiary-works

Baghbadorani, M. F. & Harandi, A. (2012). A conceptual model for business ecosystem and implications for future research. *International Proceedings of Economics Development & Research*, *52*(17), 82–86.

Balzert, H., Schröder, M. & Schäfer, C. (2011). *Wissenschaftliches Arbeiten* (2. Aufl.). Witten, Herdecke: W3L-Verlag.

Beer, S. (1984). The viable system model: Its provenance, development, methodology and pathology. *Journal of the operational research society*, 35(1), 7–25.

Bettín-Díaz, R., Rojas, A. E. & Mejía-Moncayo, C. (2018). Methodological Approach to the Definition of a Blockchain System for the Food Industry Supply Chain Traceability. In O. Gervasi, B. Murgante, S. Misra, E. Stankova, C.M. Torre, A.M.A.C. Rocha et al. (Hrsg.), *Computational Science and Its Applications – ICCSA 2018* (S. 19–33). Cham: Springer International Publishing.

Bierman, G., Abadi, M. & Torgersen, M. (2014). Understanding TypeScript. In R. Jones (Hrsg.), *ECOOP 2014 – Object-Oriented Programming* (S. 257–281). Berlin, Heidelberg: Springer.

Boshell, P. (2018). Use of the Blockchain to Contract Digitally. *Cybersecurity Law & Strategy*, *9*(2). Abgerufen von http://www.lawjournalnewsletters.com

ChainThat. (o. J.). Contract Management. Abgerufen von https://www.chainthat.com

Chiu, D. K. W., Karlapalem, K., Li, Q. & Kafeza, E. (2002). Workflow View Based E-Contracts in a Cross-Organizational E-Services Environment. *Distributed and Parallel Databases*, *12*(2), 193–216.

Cohn, M. (2004). *User stories applied: For agile software development*. Boston: Addison-Wesley Professional.

Cojocaru, L.-E., Sarraipa, J., Jardim-Golcalves, R. & Stanescu, A. M. (2014). Digital Business Ecosystem Framework for the Agro-Food Industry. In K. Mertins, F. Bénaben, R. Poler & J.-P. Bourrières (Hrsg.), *Enterprise Interoperability VI* (S. 285–296). Cham: Springer International Publishing.

Corda. (o. J.-a). API: Persistence. Abgerufen von https://docs.corda.net/api-persistence.html#schemas

Corda. (o. J.-b). AWS Marketplace. Abgerufen von https://docs.corda.net/aws-vm.html

Corda. (o. J.-c). Azure Marketplace. Abgerufen von https://docs.corda.net/azure-vm.html

Corda. (o. J.-d). Creating nodes locally. Abgerufen von https://docs.corda.net/generating-a-node.html

Corda. (o. J.-e). Event scheduling. Abgerufen von https://docs.corda.net/event-scheduling.html

Corda. (o. J.-f). Getting set up for CorDapp development. Abgerufen von https://docs.corda.net/getting-set-up.html

Corda. (o. J.-g). Network certificates. Abgerufen von https://docs.corda.net/permissioning.html

Corda. (o. J.-h). Structuring a CorDapp. Abgerufen von https://docs.corda.net/writing-a-cordapp.html

Crosby, M., Pattanayak, P., Verma, S. & Kalyanaraman, V. (2016). Blockchain technology: Beyond bitcoin. *Applied Innovation*, *2*(6), 71.

Dangl, A. (2018, 28. Juni). Sicheres Vertragsmanagement in der Cloud. *KMU Magazin*. Abgerufen von https://www.kmu-magazin.ch

Dorschel, J. & Manz, S. (2017). Contract Management in der Blockchain. *Zeitschrift für das gesamte Kreditwesen*, *71*(11), 2.

Dudley, J. (2019). Cordapp-Example [GitHub-Repository]. Abgerufen von https://github.com/corda/samples/tree/release-V4/cordapp-example

EFD. (2018). *Aktionsplan Digitale Schweiz*. Abgerufen von https://www.bakom.admin.ch/bakom/de/home/digital-und-internet/strategie-digitale-schweiz.html

Eidgenössischer Bundesrat. (2018). *Rechtliche Grundlagen für Distributed Ledger-Technologie und Blockchain in der Schweiz*. Abgerufen von https://www.digitaldialog.swiss/de/dialog/blockchain

Falzone, E. & Bernaschina, C. (2018). Model Based Rapid Prototyping and Evolution of Web Application. In T. Mikkonen, R. Klamma & J. Hernández (Hrsg.), *Web Engineering* (S. 496–500). Basel: Springer International Publishing.

Feldschlösschen. (o. J.). Together Towards ZERO. Abgerufen von https://feldschloesschen.swiss/de/nachhaltigkeit/together-towards-zero/

Finnegan, J., Malone, P., Maranon, M. A. E. & Guillen, P. B. (2007). Contract Modelling for Digital Business Ecosystems. *2007 Inaugural IEEE-IES Digital EcoSystems and Technologies Conference* (S. 71–76). Cairns, Australia.

Fischer, C., Fiedler, I. & Babenko, L. (2019). Blockchain-Technologie im Handel der Zukunft. In G. Heinemann, H. M. Gehrckens & T. Täuber (Hrsg.), *Handel mit Mehrwert: Digitaler Wandel in Märkten, Geschäftsmodellen und Geschäftssystemen* (S. 441–471). Wiesbaden: Springer Fachmedien.

Funk, B., Gómez, J. M., Niemeyer, P. & Teuteberg, F. (2010). Geschäftsprozessmanagement und Prozessmodellierung. In B. Funk, J. Marx Gómez, P. Niemeyer & F. Teuteberg (Hrsg.), *Geschäftsprozessintegration mit SAP: Fallstudien zur Steuerung von Wertschöpfungsprozessen entlang der Supply Chain* (S. 7–53). Berlin, Heidelberg: Springer.

Gendal, R., Carlyle, J., Grigg, I. & Hearn, M. (2016). *Corda: An Introduction*. (White Paper). R3 Consortium. Abgerufen von https://docs.corda.net/releases/release-M7.0/_static/corda-introductory-whitepaper.pdf

Governatori, G., Idelberger, F., Milosevic, Z., Riveret, R., Sartor, G. & Xu, X. (2018). On legal contracts, imperative and declarative smart contracts, and blockchain systems. *Artificial Intelligence and Law*, *26*(4), 377–409.

Grimm, J. H., Hofstetter, J. S. & Sarkis, J. (2016). Exploring sub-suppliers' compliance with corporate sustainability standards. *Journal of Cleaner Production*, *26*(112), 1971–1984.

Gutierrez, F. (2017). *Spring Boot Messaging*. Berkeley, CA: Apress.

GUTS. (o. J.). Powerful features - Designed for fans, artists, venues & organizers. Abgerufen von https://guts.tickets/features/

Habermann, F. (2013). Hybrides Projektmanagement – agile und klassische Vorgehensmodelle im Zusammenspiel. *HMD Praxis der Wirtschaftsinformatik*, *50*(5), 93–102.

Hamedani, M. (2018, 5. November). React vs. Angular: The Complete Comparison. Abgerufen von https://programmingwithmosh.com/react/react-vs-angular/

Hardy, J. (2016, 9. Oktober). What are the advantages of Spring Boot over Node.js for a RESTful web service? [Blog-Beitrag]. Abgerufen von https://www.quora.com/What-are-the-advantages-of-Spring-Boot-over-Node-js-for-a-RESTful-web-service

Hearn, M. (2016). *Corda: A distributed ledger*. (White Paper). R3 Consortium. Abgerufen von https://www.corda.net/content/corda-technical-whitepaper.pdf

Hearn, M. (2017, 17. Juli). Can one single Corda node support multiple parties/accounts? [Blog-Beitrag]. Abgerufen von https://stackoverflow.com/questions/45020131/can-one-single-corda-node-support-multiple-parties-accounts

Heumüller, E. & Richter, S. (2018). Das Blockchain-Ökosystem als Analyse-Ansatz. *Wirtschaftsinformatik & Management*, *10*(3), 60–65.

Höfelmann, D. & Sander, P. (2019). *Entscheidungshilfe für den Einsatz von Blockchain-Technologien in Unternehmen: Vier Frameworks im Vergleich*. (Working Paper). Frankfurt School Blockchain Center. Abgerufen von http://explore-ip.com/2019_Entscheidungshilfe-fuer-den-Einsatz-von-Blockchain-Technologien.pdf

H2. (o. J.). Welcome to H2, the Java SQL database. Abgerufen von https://www.h2database.com/html/main.html

Hull, E., Jackson, K. & Dick, J. (2011). *Requirements Engineering* (3. Aufl.). London: Springer.

Iansiti, M. & Lakhani, K. R. (2017). The truth about blockchain. *Harvard Business Review*, *95*(1), 118–127.

Inhubber. (o. J.). Contract Management and Operating Platform. Abgerufen von https://inhubber.com/

In 28 Minutes. (2017, 3. Dezember). Spring Boot and H2 in memory database - Why, What and How?. Abgerufen von http://www.springboottutorial.com/spring-boot-and-h2-in-memory-database

Jarnjak, F. (2010). Notice of Retraction Flexible GUI in robotics applications using Windows Presentation Foundation framework and Model View ViewModel pattern. *4th International Conference on New Trends in Information Science and Service Science* (S. 176–179). Gyeongju, South Korea.

Jochum, K. (2019, 7. März). Six setzt für die Blockchain-Börse auf Corda von R3. *inside-it.ch*. Abgerufen von https://www.inside-it.ch

Johnson, R., Hoeller, J., Donald, K., Sampaleanu, C., Harrop, R., Risberg, T. et al. (2004). The spring framework–reference documentation. (Dokumentation 3.2.17). Abgerufen von https://docs.spring.io/autorepo/docs/spring-framework/3.2.17.RELEASE/spring-framework-reference/pdf/spring-framework-reference.pdf

Kandiah, G. & Gossain, S. (1998). Reinventing value: The new business ecosystem. *Strategy & Leadership*, *26*(5), 28–33.

Kolmykova, A. (2016). *Supply Chain Integration*. Wiesbaden: Springer Gabler. Abgerufen von https://link.springer.com/content/pdf/10.1007%2F978-3-658-02664-6.pdf

Konfidio. (2018, 23. November). Contract Management: The KCS Solution & How Blockchain will change Contract Management [Blog-Beitrag]. Abgerufen von https://medium.com/konfid-io-blockchain-reports/contract-management-the-kcs-solution-how-blockchain-will-change-contract-management-941f0b74c51a

Konfidio. (o. J.). Blockchain-powered Procurement Contract Management. Abgerufen von https://www.kcs.konfidio.com/

Koren, I. & Klamma, R. (2018). Generation of Web Frontends from API Documentation with Direwolf Interaction Flow Designer. In T. Mikkonen, R. Klamma & J. Hernández (Hrsg.), *Web Engineering* (S. 492–495). Basel: Springer International Publishing.

Korpela, K., Hallikas, J. & Dahlberg, T. (2017). Digital supply chain transformation toward blockchain integration. P*roceedings of the 50th Hawaii international conference on system sciences* (S. 4182–4191). Waikoloa, USA.

Kostrzewa, D., Bach, M., Brzeski, R. & Werner, A. (2016). Performance Aspect of the In-Memory Databases Accessed via JDBC. In S. Kozielski, D. Mrozek, P. Kasprowski, B. Małysiak-Mrozek & D. Kostrzewa (Hrsg.), *Beyond Databases, Architectures and Structures. Advanced Technologies for Data Mining and Knowledge Discovery* (S. 236–252). Cham: Springer International Publishing.

Krishna, P. R., Karlapalem, K. & Dani, A. R. (2005). From Contracts to E-Contracts: Modeling and Enactment. *Information Technology and Management*, *6*(4), 363–387.

Kwak, K. H., Kong, J. T., Cho, S. I., Phuong, H. T. & Gim, G. Y. (2019). A Study on the Design of Efficient Private Blockchain. In R. Lee (Hrsg.), *Computational Science/Intelligence & Applied Informatics* (S. 93–121). Cham: Springer International Publishing.

Lavikka, R., Hirvensalo, A., Smeds, R. & Jaatinen, M. (2017). Transforming a Supply Chain Towards a Digital Business Ecosystem. In H. Lödding, R. Riedel, K.-D. Thoben, G. von Cieminski & D. Kiritsis (Hrsg.), *Advances in Production Management Systems. The Path to Intelligent, Collaborative and Sustainable Manufacturing* (S. 295–301). Cham: Springer International Publishing.

Liu, X. & Partovi, F. (2007). A Collaborate Contract to Retain Long Term Relationship and Quality Improvement in Supply Chains. *2007 International Conference on Service Systems and Service Management* (S. 1–5). Chengdu, China.

Lucassen, G., Dalpiaz, F., van der Werf, J. M. & Brinkkemper, S. (2016). The Use and Effectiveness of User Stories in Practice. In M. Daneva & O. Pastor (Hrsg.), *Requirements Engineering: Foundation for Software Quality* (S. 205–222). Cham: Springer International Publishing.

Luu, L., Chu, D.-H., Olickel, H., Saxena, P. & Hobor, A. (2016). Making smart contracts smarter. *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security* (S. 254–269). New York: ACM.

Mäntymäki, M., Salmela, H. & Turunen, M. (2018). Do Business Ecosystems Differ from Other Business Networks? The Case of an Emerging Business Ecosystem for Digital Real-Estate and Facility Services. In S. A. Al-Sharhan, A.C. Simintiras, Y. K. Dwivedi, M. Janssen, L. Tahat et al. (Hrsg.), *Challenges and Opportunities in the Digital Era* (S. 102–116). Cham: Springer International Publishing.

Maple, C. & Jackson, J. (2019). Selecting Effective Blockchain Solutions. In G. Mencagli, D. B. Heras, V. Cardellini, E. Casalicchio, E. Jeannot, F. Wolf et al. (Hrsg.), *Euro-Par 2018: Parallel Processing Workshops* (S. 392–403). Cham: Springer International Publishing.

Marx, L. (2016, 6. Dezember). Is Angular 2+ MVVM? [Blog-Beitrag]. Abgerufen von https://malcoded.com/posts/angular-2-components-and-mvvm/

Massessi, D. (2018, 12. Dezember). Public Vs Private Blockchain In A Nutshell [Blog-Beitrag]. Abgerufen von https://medium.com/coinmonks/public-vs-private-blockchain-in-a-nutshell-c9fe284fa39f

Matthes, F., Faber, A. & Hernandez-Mendez, A. (o. J.). Discovering Insights In Business Ecosystems. *Business Ecosystem Explorer*. Abgerufen von https://ecosystem-explorer.in.tum.de/#/

Meier, A. (2018). Blockchain. *HMD Praxis der Wirtschaftsinformatik, 55*(6), 1133–1134.

Merz, M. (2004). XML-Based E-Contracting. In M.J. Mendes, R. Suomi & C. Passos (Hrsg.), *Digital Communities in a Networked Society: e-Commerce, e-Business and e-Government* (S. 333–344). Boston, MA: Springer US.

Milton, S. K. & Johnson, L. W. (2012). Service blueprinting and BPMN: a comparison. *Managing Service Quality: An International Journal*, *22*(6), 606–621.

Moore, J. F. (1993). Predators and prey: a new ecology of competition. *Harvard business review*, *71*(3), 75–86.

Moore, J. F. (1996). *The death of competition: leadership and strategy in the age of business ecosystems*. New York: HarperBusiness.

Moothart, A. & Escoto, K. (2018, August). Introduction to Corda Development [Video file]. *Udemy*. Abgerufen von https://www.udemy.com/corda-development/

Morris, N. (2018, April). Finastra launches first live R3 Corda app. *Ledger Insights*. Abgerufen von https://www.ledgerinsights.com/finastra-first-live-r3-corda/

Mulligan, C., Rangaswami, J., Warren, S. & Zhu-Scott, J. (2018, 23. März). These 11 questions will help you decide if blockchain is right for your business. Abgerufen von https://www.weforum.org/agenda/2018/04/questions-blockchain-toolkit-right-for-business/

Nakamoto, S. (2018). *Bitcoin: A peer-to-peer electronic cash system*. (White Paper). Bitcoin. Abgerufen von https://bitcoin.org/bitcoin.pdf

Neuhaus, J. (2017, 28. August). Angular vs. React vs. Vue: A 2017 comparison [Blog-Beitrag]. Abgerufen von https://medium.com/unicorn-supplies/angular-vs-react-vs-vue-a-2017-comparison-c5c52d620176

Newton, D. (2018, 5. Juni). What is Corda? [Blog-Beitrag]. Abgerufen von https://medium.com/corda/what-is-corda-6417b14c8dc7

Nguyen, C. D., Gallagher, E., Read, A. & de Vreede, G.-J. (2009). Generating User Stories in Groups. In L. Carriço, N. Baloian & B. Fonseca (Hrsg.), *Groupware: Design, Implementation, and Use* (S. 357–364). Berlin, Heidelberg: Springer.

Nguyen, Q. K. & Dang, Q. V. (2018). Blockchain Technology for the Advancement of the Future. *2018 4th International Conference on Green Technology and Sustainable Development (GTSD)* (S. 483–486). Ho Chi Minh City, Vietnam.

Peukert, H. (2018, 7. Mai). Vertrag. *Gabler Wirtschaftslexikon*. Abgerufen von https://wirtschaftslexikon.gabler.de/definition/vertrag-49761

Pfander, M. (2016, 17. November). Carlsberg will das Patent auf Gerste. *Tages-Anzeiger*. Abgerufen von https://www.tagesanzeiger.ch/

R3 Consortium. (o. J.). Welcome to Corda !. Abgerufen von https://docs.corda.net/index.html

Razavi, A. R., Krause, P. J. & Strømmen-Bakhtiar, A. (2010). From business ecosystems towards digital business ecosystems. *4th IEEE International Conference on Digital Ecosystems and Technologies* (S. 290–295). Dubai, United Arab Emirates.

Rilee, K. (2018, 17. Februar). Understanding Corda – Transactions vs. Blocks [Blog-Beitrag]. Abgerufen von https://medium.com/kokster/understanding-corda-transactions-vs-blocks-1f3d42d00cc2

Rivero, J. M., Rossi, G., Grigera, J., Burella, J., Luna, E. R. & Gordillo, S. (2010). From Mockups to User Interface Models: An Extensible Model Driven Approach. In F. Daniel & F.M. Facca (Hrsg.), *Current Trends in Web Engineering* (S. 13–24). Berlin, Heidelberg: Springer.

Rivero, J. M., Grigera, J., Rossi, G., Robles Luna, E. & Koch, N. (2012). Towards Agile Model-Driven Web Engineering. In S. Nurcan (Hrsg.), *IS Olympics: Information Systems in a Diverse World* (S. 142–155). Berlin, Heidelberg: Springer.

Rong, K., Hu, G., Lin, Y., Shi, Y. & Guo, L. (2015). Understanding business ecosystem using a 6C framework in Internet-of-Things-based sectors. *International Journal of Production Economics*, *159*(1), 41–55.

Sandhaus, G., Berg, B. & Knott, P. (2014). Vorgehensmodelle für die Softwareentwicklung. In B. Berg, P. Knott & G. Sandhaus (Hrsg.), *Hybride Softwareentwicklung: Das Beste aus klassischen und agilen Methoden in einem Modell vereint* (S. 23–51). Berlin, Heidelberg: Springer.

Sandhaus, G., Knott, P. & Berg, B. (2015). Hybride Softwareentwicklung. *Informatik-Spektrum*, *38*(4), 306–309.

Sandner, P. (2018, 17. Oktober). Vergleich dreier Blockchain-Projekte mit Anwendungspotenzial für den Finanzmarkt [Blog-Beitrag]. Abgerufen von https://www.der-bank-blog.de/blockchain-vergleich/technologie/37480/

Schmitt, C. (2001). Chancen für Loyalitätsprogramme durch das Internet: das Beispiel Lufthansa Miles & More. In S. Helmke & W. Dangelmaier (Hrsg.), *Effektives Customer Relationship Management: Instrumente — Einführungskonzepte — Organisation* (S. 85–99). Wiesbaden: Gabler Verlag.

Schrauzer, S. (2012). Mobile Contracting. *HMD Praxis der Wirtschaftsinformatik*, *49*(4), 83–92.

Schwarzmüller, M. (2019, 9. Januar). Angular 7 (formerly Angular 2) - The Complete Guide [Video file]. *Udemy*. Abgerufen von https://www.udemy.com/the-complete-guide-to-angular-2/

Seebacher, S. & Schüritz, R. (2017). Blockchain Technology as an Enabler of Service Systems: A Structured Literature Review. In S. Za, M. Drăgoicea & M. Cavallari (Hrsg.), *Exploring Services Science* (S. 12–23). Cham: Springer International Publishing.

Slack, N., Brandon-Jones, A. & Johnston, R. (2016). *Operations Management* (8. Aufl.). Harlow: Pearson.

Stadler, T. (o. J.). Customer Supply Chain Unterwegs für unsere Kunden. Abgerufen von https://feldschloesschen.swiss/de/uber-uns/geschaftsleitung/customer-supply-chain/

Sulc, A. (2014, 19. November). Die Knebelverträge der grossen Brauer. *Der Bund*. Abgerufen von https://www.derbund.ch/

Sury, U. (2016). Blockchain und Recht. *Informatik-Spektrum*, *39*(5), 402–410.

Swan, M. (2015). *Blockchain: Blueprint for a new economy*. Sebastopol, CA: O'Reilly.

Thomas, L. D. & Autio, E. (2014). The fifth facet: The ecosystem as an organizational field. *DRUID Society Conference* (S. 16–18). Copenhagen, Denmark.

Uher, T. E. & Davenport, P. (2009). *Fundamentals of building contract management* (2. Aufl.). Sydney: UNSW Press.

Valenta, M. & Sander, P. (2017). *Comparison of Ethereum, Hyperledger Fabric and Corda*. (Working Paper). Frankfurt School Blockchain Center. Abgerufen von http://explore-ip.com/2017_Comparison-of-Ethereum-Hyperledger-Corda.pdf

Voigt, K.-I. (2018, 19. Februar). Supply Chain Management (SCM). *Gabler Wirtschaftslexikon*. Abgerufen von https://wirtschaftslexikon.gabler.de/definition/supply-chain-management-scm-49361#references

von Waldegge, S. H. (2018). Steigerung der Effizienz im Vertragsmanagement. In A. Khare, D. Kessler & J. Wirsam (Hrsg.), *Marktorientiertes Produkt- und Produktionsmanagement in digitalen Umwelten: Festgabe für Klaus Bellmann zum 75. Geburtstag* (S. 85–100). Wiesbaden: Springer Fachmedien.

Walls, C. (2016). *Spring Boot in Action*. Shelter, NY: Manning.

Wang, T., Grefen, P. & Vonk, J. (2008). Ensuring Transactional Reliability by E-Contracting. In Z. Bellahsène & M. Léonard (Hrsg.), *Advanced Information Systems Engineering* (S. 262–265). Berlin, Heidelberg: Springer.

Wüst, K. & Gervais, A. (2018). Do you Need a Blockchain? *2018 Crypto Valley Conference on Blockchain Technology (CVCBT)* (S. 45–54). Zug, Switzerland.

Xu, X., Pautasso, C., Zhu, L., Gramoli, V., Ponomarev, A., Tran, A. B. et al. (2016). The Blockchain as a Software Connector. *2016 13th Working IEEE/IFIP Conference on Software Architecture (WICSA)* (S. 182–191). Venice, Italy.

Yaga, D., Mell, P., Roby, N. & Scarfone, K. (2018). *Blockchain Technology Overview*. (Standard No. 8202). National Institute of Standards and Technology. Abgerufen von https://nvlpubs.nist.gov/nistpubs/ir/2018/NIST.IR.8202.pdf

Yue, L. (2013). Structure and Optimization of the Business Ecosystem – Case Study on Jinguang Group. In R. Dou (Hrsg.), *Proceedings of 2012 3rd International Asia Conference on Industrial Engineering and Management Innovation (IEMI2012)* (S. 619–629). Berlin, Heidelberg: Springer.

Zavolokina, L., Spychiger, F., Tessone, C. J. & Schwabe, G. (2018). Incentivizing Data Quality in Blockchains for Inter-Organizational Networks – Learning from the Digital Car Dossier. *Thirty ninth International Conference on Information Systems*. San Francisco, USA.

Zheng, Z., Xie, S., Dai, H.-N., Chen, X. & Wang, H. (2018). Blockchain challenges and opportunities: a survey. *International Journal of Web and Grid Services, 14*(4), 352–375.

Zhou, W., Li, L., Luo, M. & Chou, W. (2014). REST API Design Patterns for SDN Northbound API. *2014 28th International Conference on Advanced Information Networking and Applications Workshops* (S. 358–365). Victoria, Canada.

Zhu, S. & Xu, Y. (2012). Complexity measure of supply chain networks. *2012 24th Chinese Control and Decision Conference (CCDC)* (S. 2220–2224). Taiyuan, China.

Zubko, H. & Bohner, T. (2018, 19. April). Is blockchain for you or your organization or your Industry? [Blog-Beitrag]. Abgerufen von https://www.hyperledger.org/blog/2018/04/19/lessons-learned-from-hyperledger-fabric-poc-projects

Anhang

A. Abbildungen

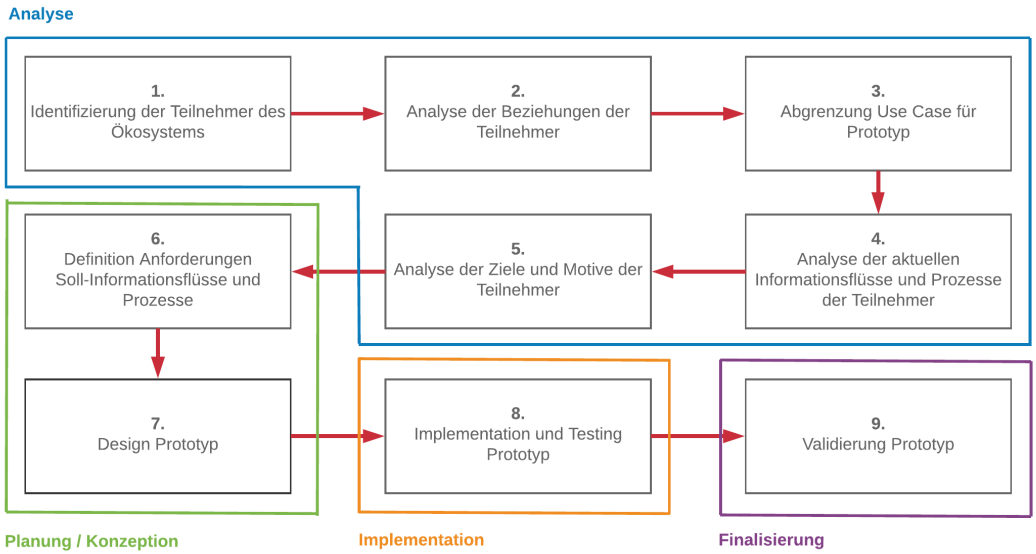

Abbildung 1: Transformationsprozess zu einem Business-Ökosystem (Yue, 2013; Lavikka et al., 2017)

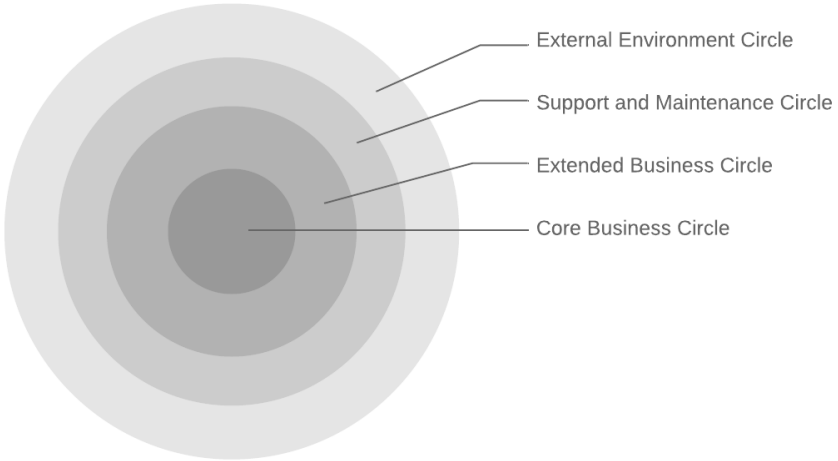

Abbildung 2: Struktur Business-Ökosystem (Yue, 2013)

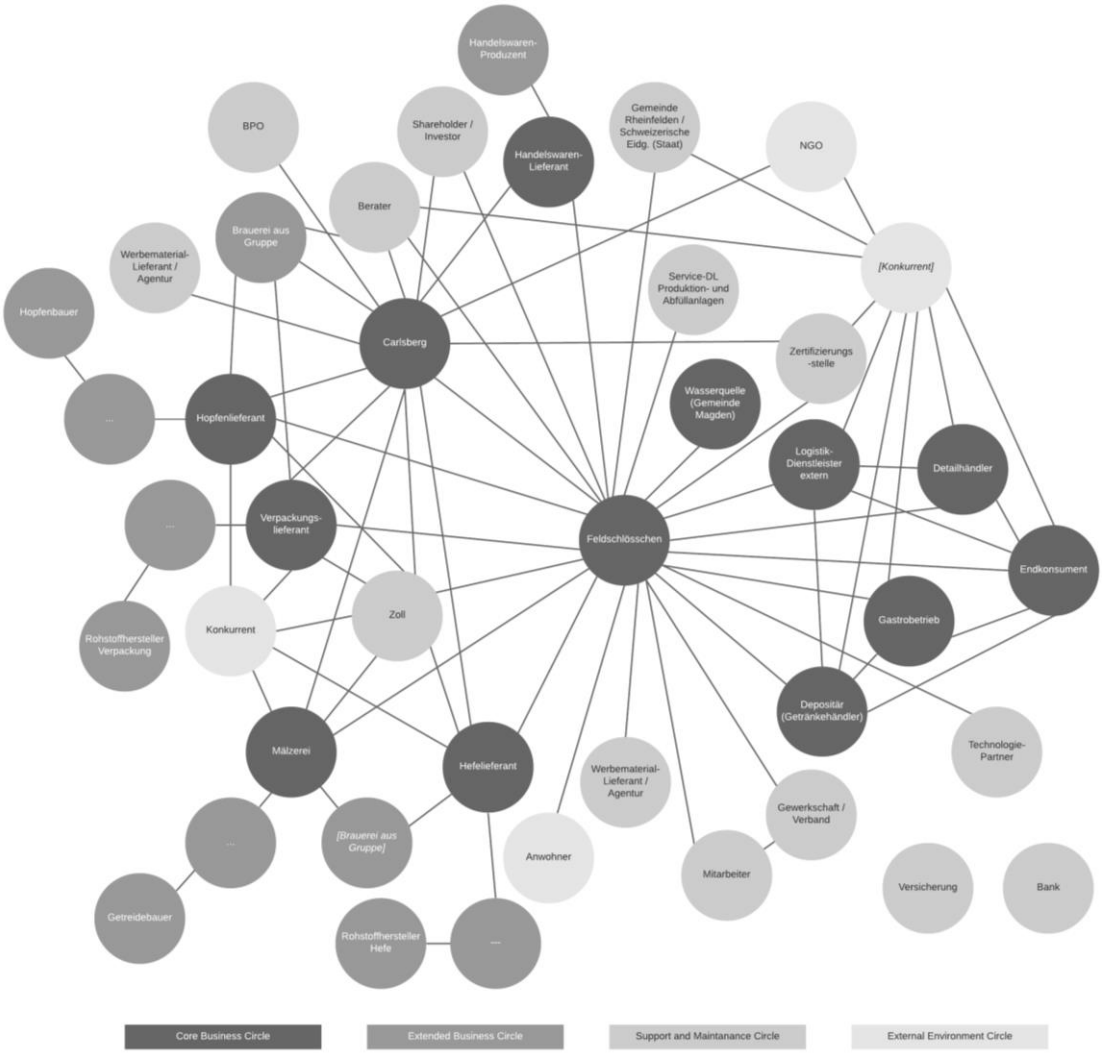

Abbildung 3: Business-Ökosystem der Feldschlösschen Getränke AG

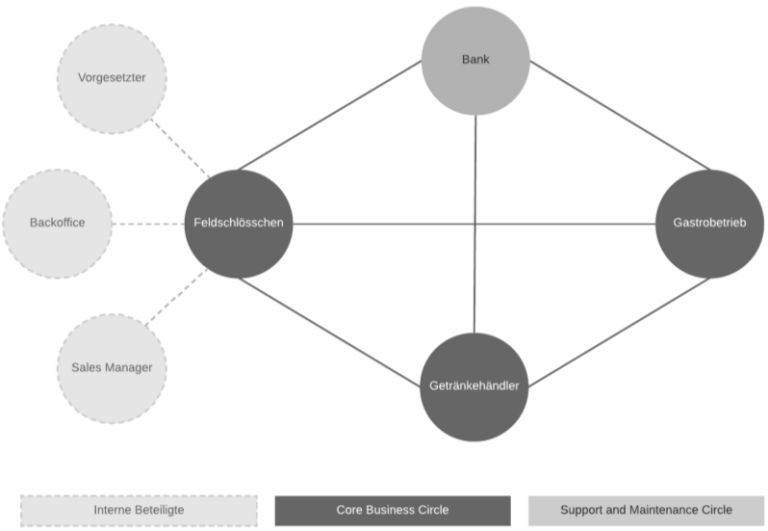

Abbildung 4: Akteure Darlehensgeschäft Feldschlösschen Getränke AG

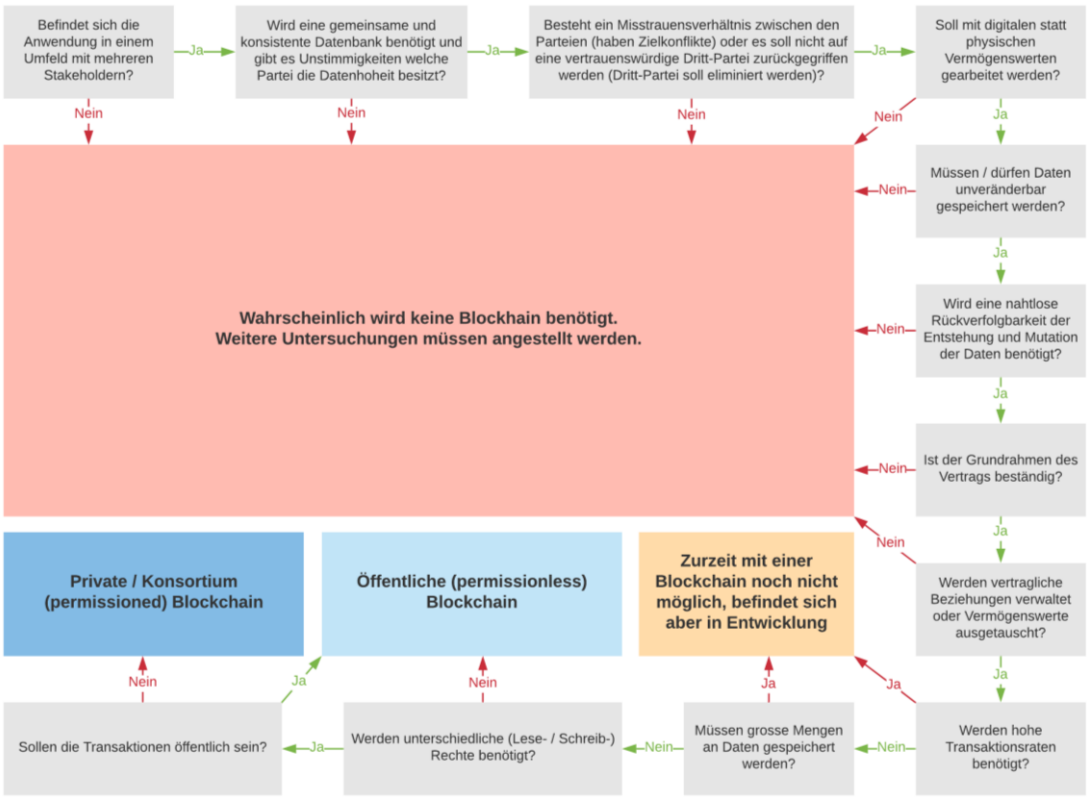

Abbildung 5: Vorgehensmodell Entscheidung Blockchain-Typ (Wüst und Gervais 2018; Mulligan et al., 2018; Zubko & Bohner, 2018; Yaga et al., 2018)

Manual Task	Keine Beteiligung des Systems, alleinige Ausführung durch Person (auch Human Task)
User Task	Interaktion des Systems mit Benutzer über Eingabe oder Ausgabe (z.B. Eingabeformulare)
Service Task	Keine Beteiligung durch Benutzer, automatisierte Ausführung eines Service

Abbildung 6: Typisierung Tasks

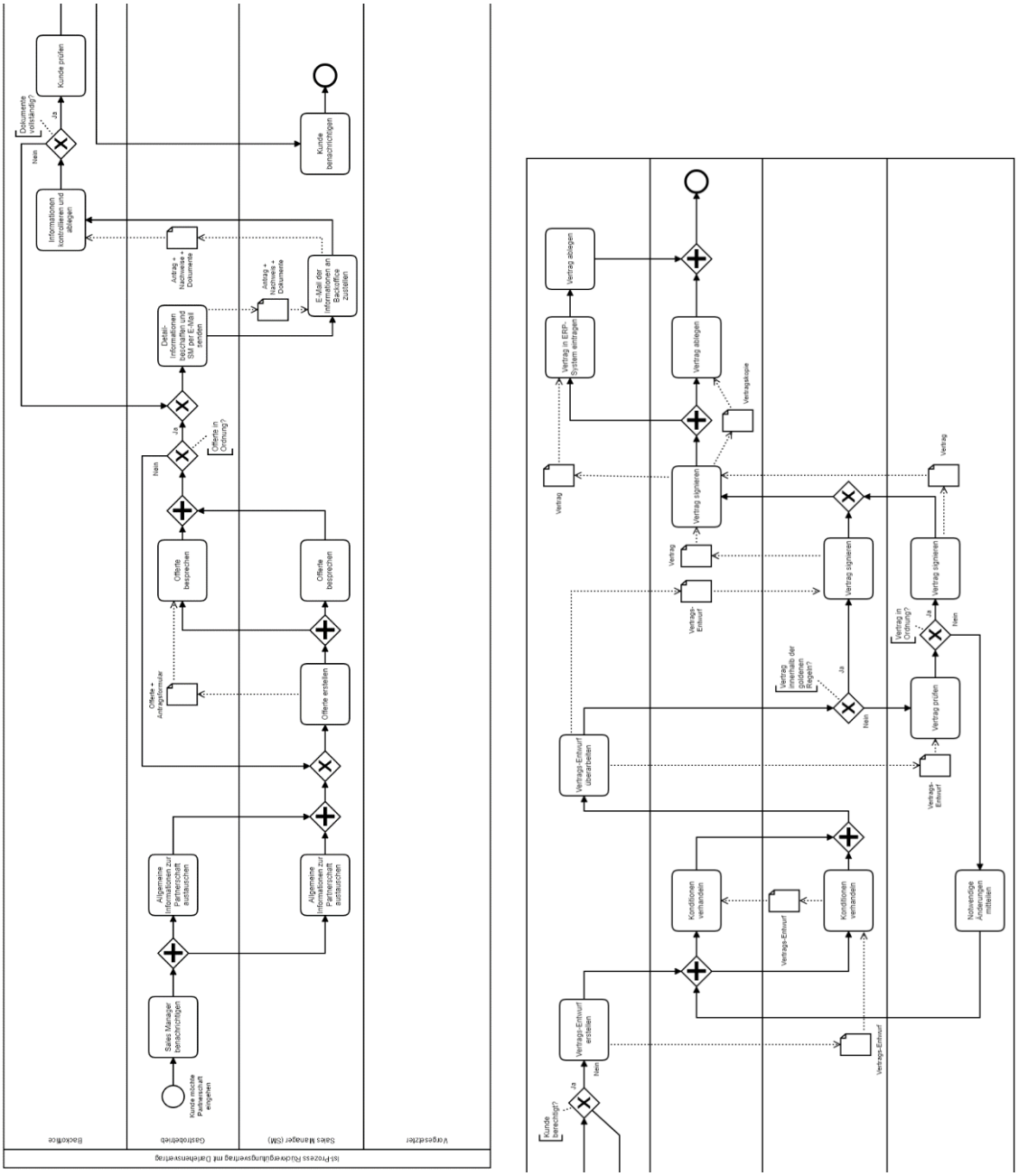

Abbildung 7: Ist-Prozess Vertragsentstehung

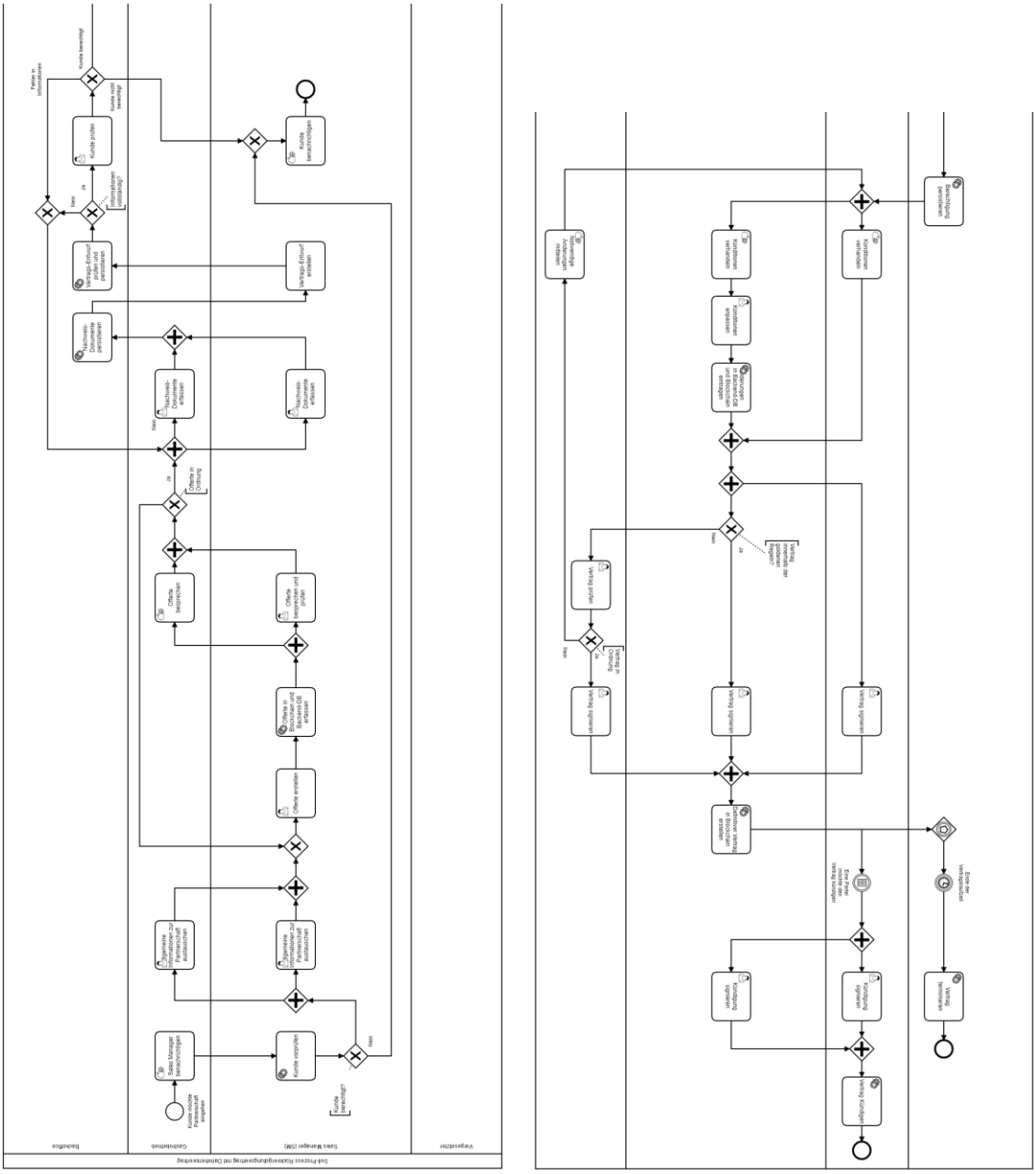

Abbildung 8: Soll-Prozess Vertragsentstehung

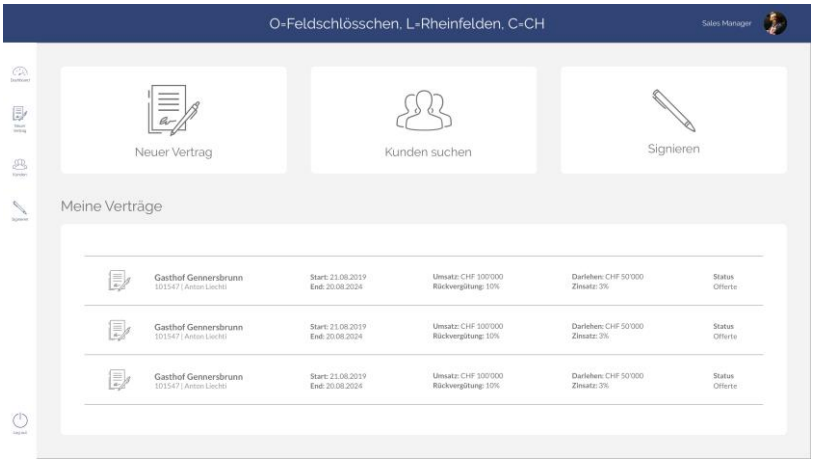

Abbildung 9: Mock-up Dashboard Sales-Manager

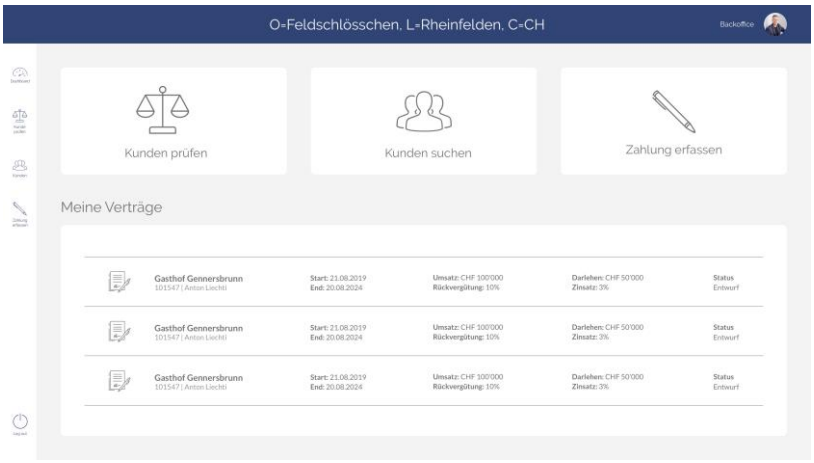

Abbildung 10: Mock-up Dashboard Backoffice

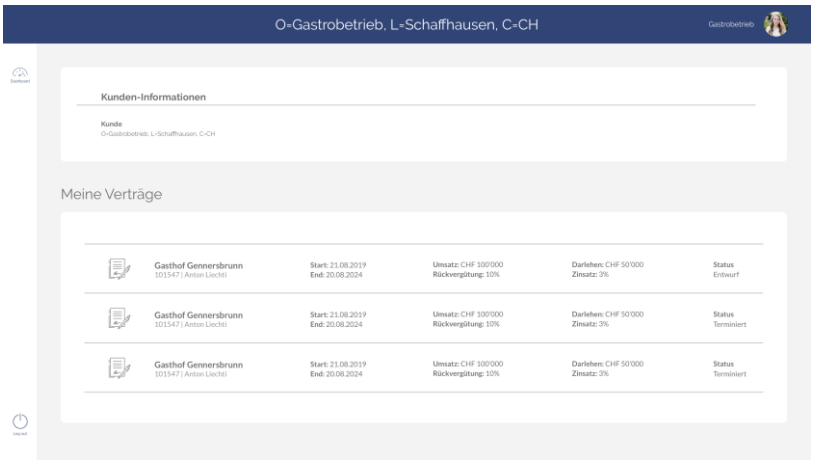

Abbildung 11: Mock-up Dashboard Gastrobetrieb

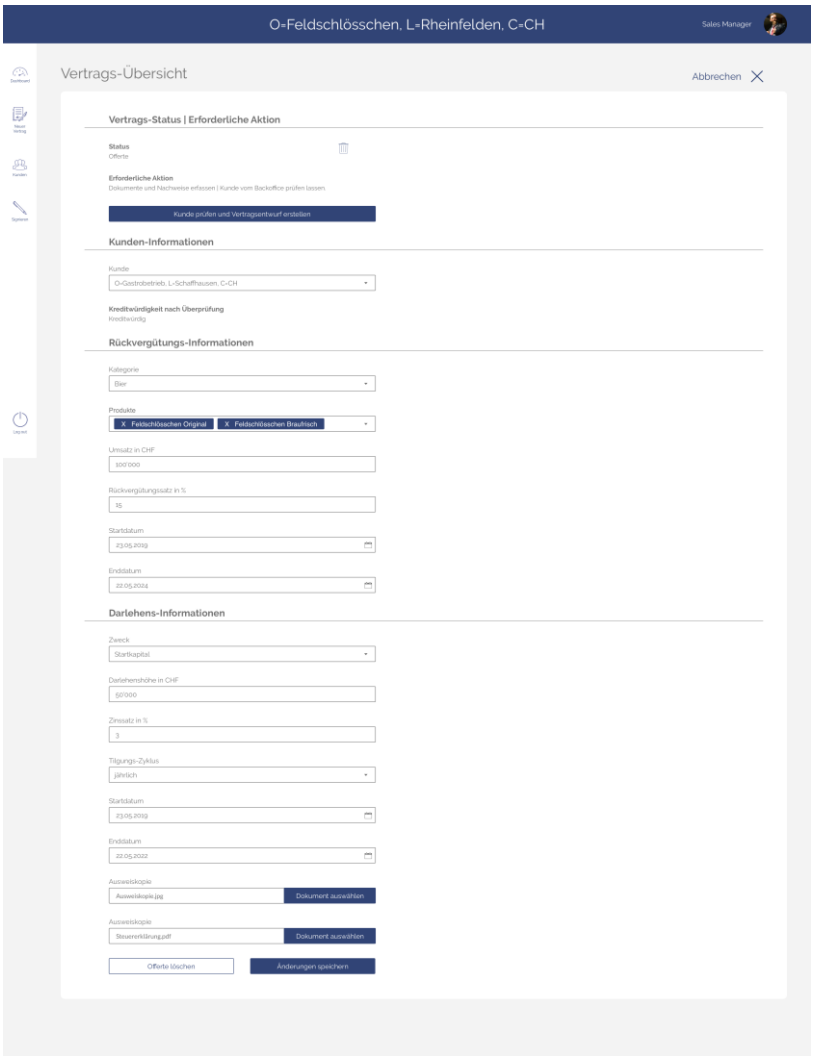

Abbildung 12: Mock-up Eingabeformular

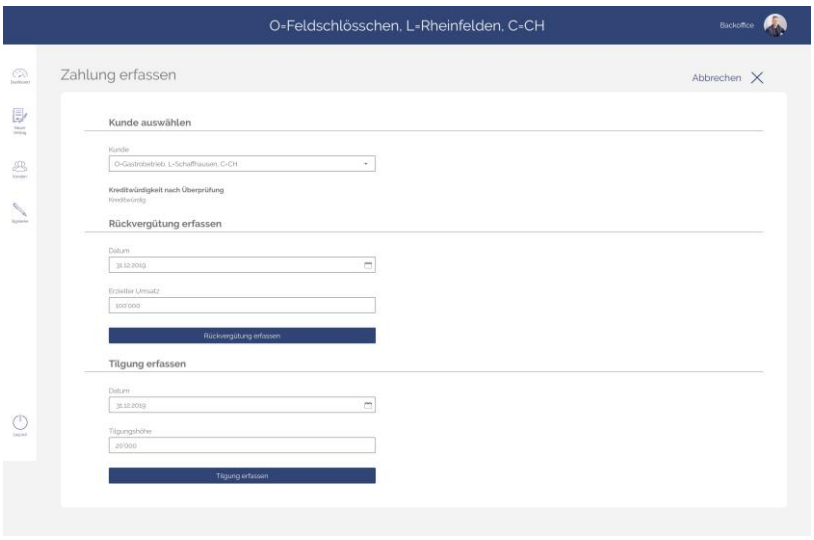

Abbildung 13: Mock-up Erfassung Zahlungen

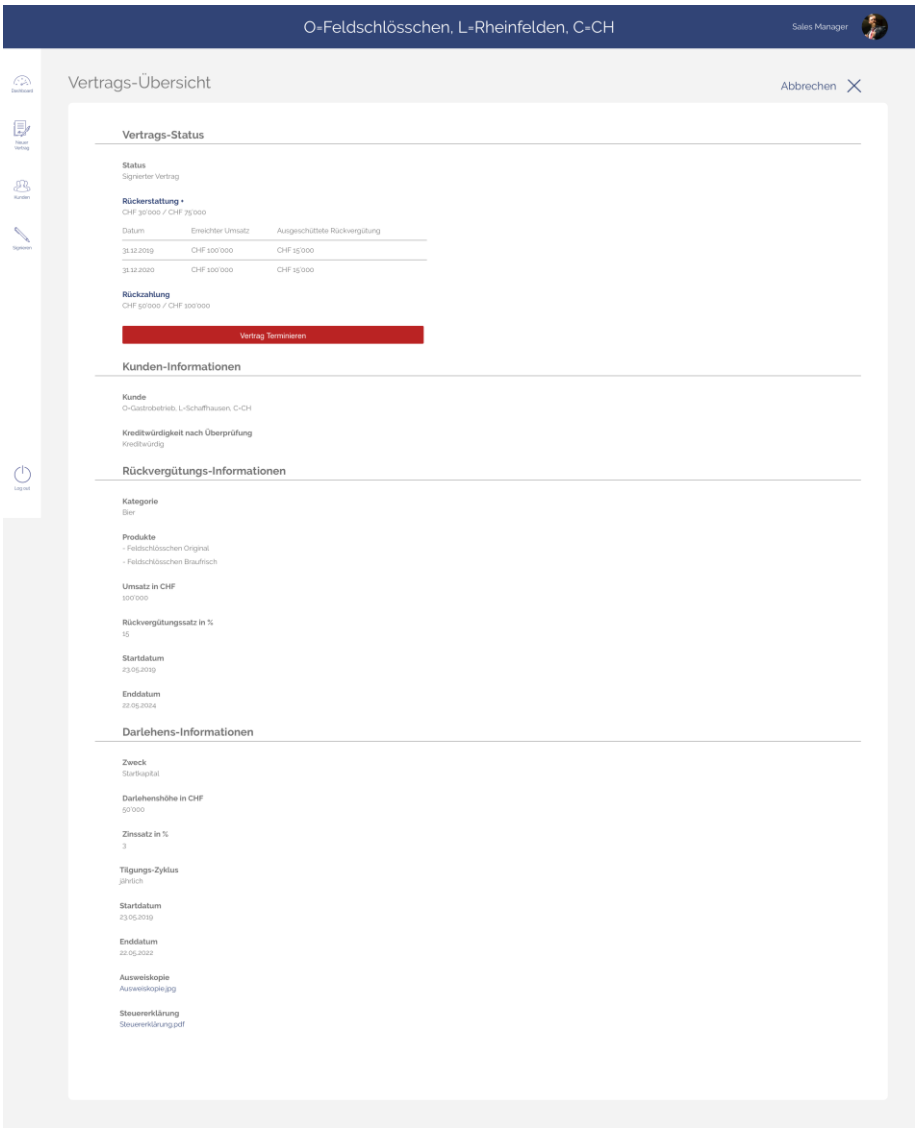

Abbildung 14: Mock-up Vertrags-Übersicht

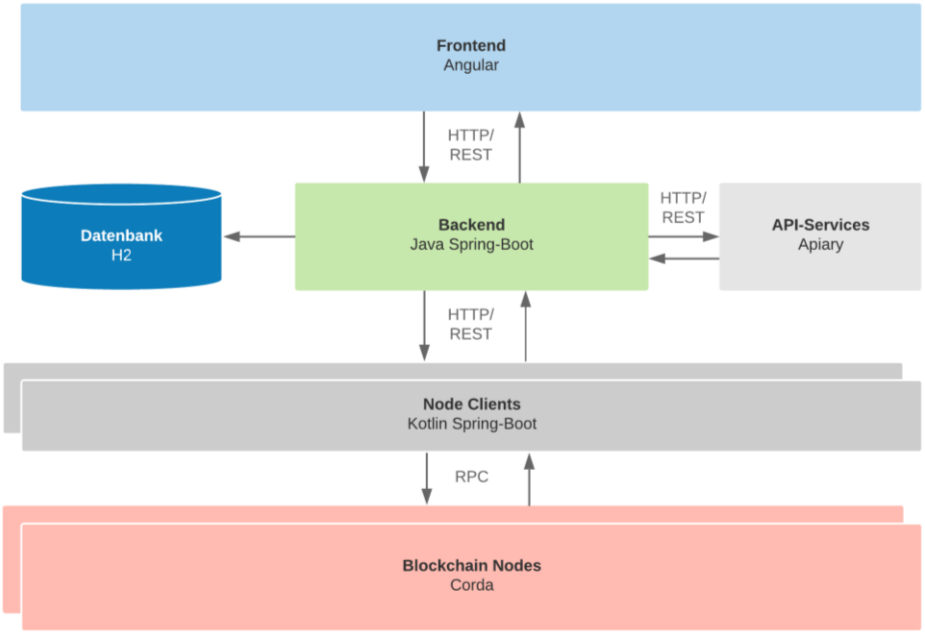

Abbildung 15: Applikations-Architektur Prototyp

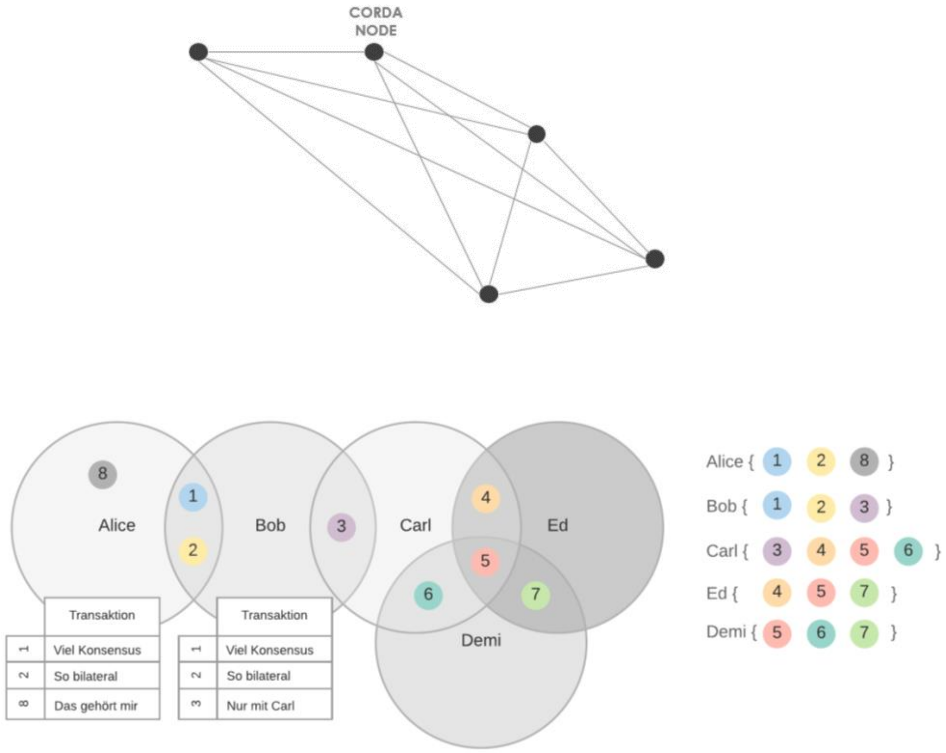

Abbildung 16: Corda-Netzwerk (Moothart & Escoto, 2018)

State 1						
ID	Kreditgeber	Kreditnehmer	Darlehenssumme	Datum	Zinssatz	Rückzahlung
1	Alice	Bob	100	20.12.2019	5%	0

Neuer aktualisierter State 2						
ID	Kreditgeber	Kreditnehmer	Darlehenssumme	Datum	Zinssatz	Rückzahlung
1	Alice	Bob	100	20.12.2019	5%	20

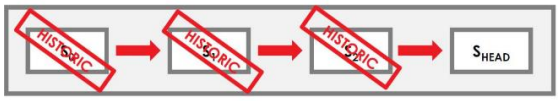

Abbildung 17: Corda-State (Moothart & Escoto, 2018)

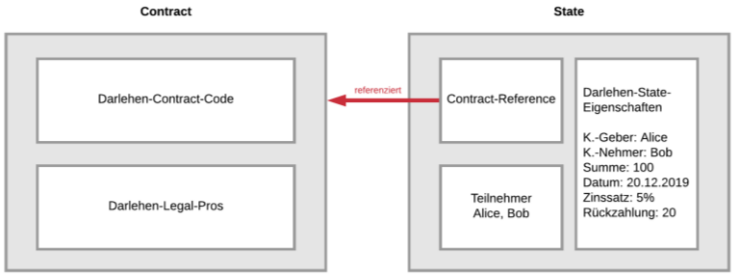

Abbildung 18: Corda-Contract (Moothart & Escoto, 2018)

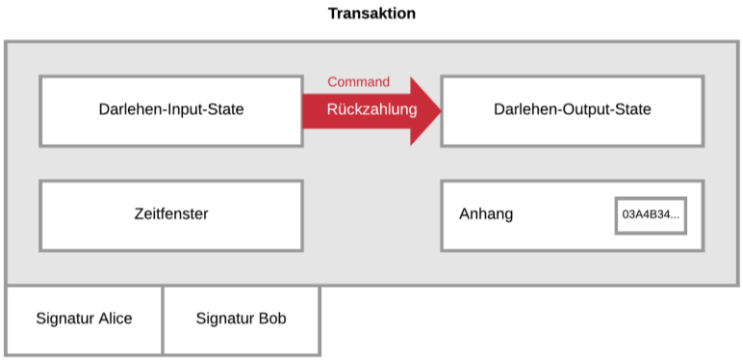

Abbildung 19: Corda-Transaktion (Moothart & Escoto, 2018)

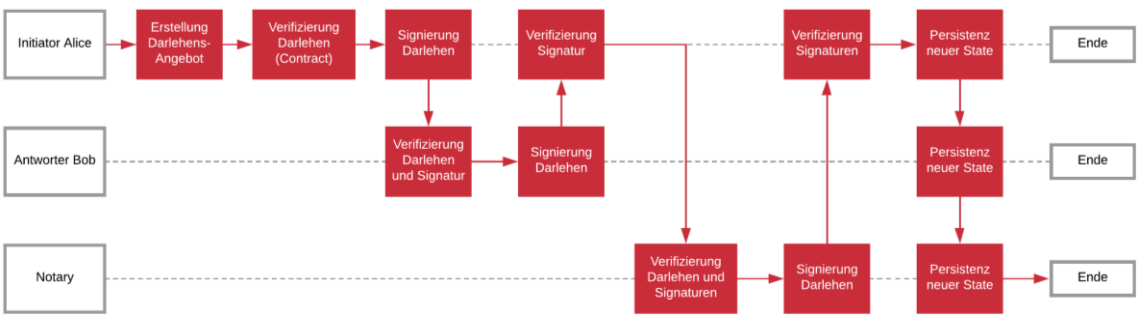

Abbildung 20: Corda-Flow (Moothart & Escoto, 2018)

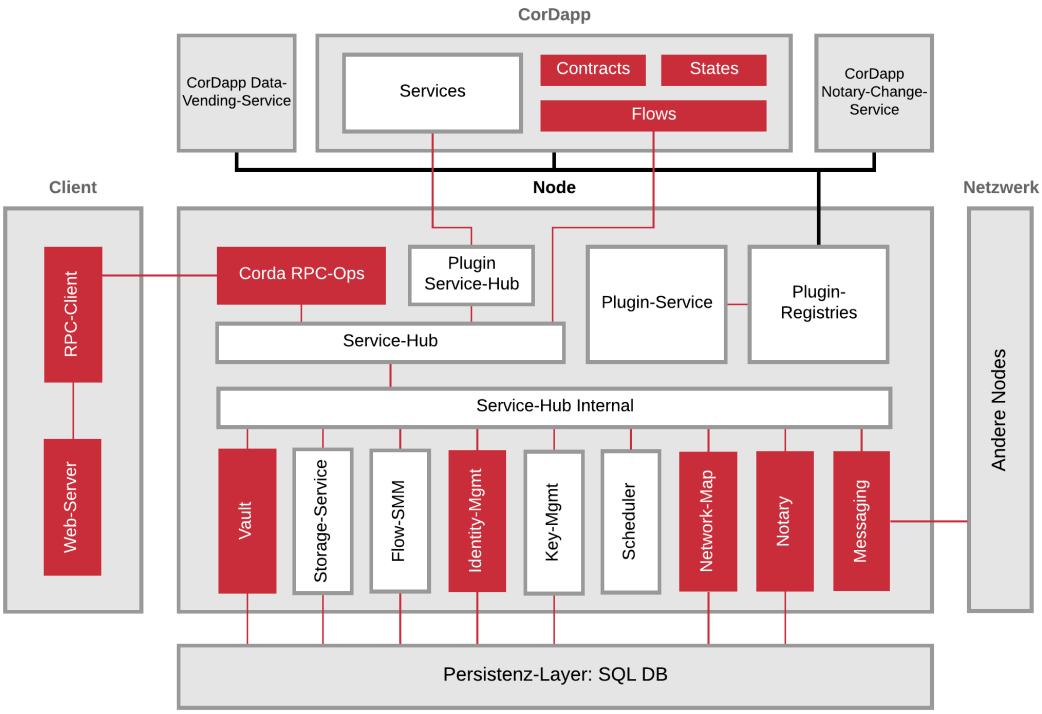

Abbildung 21: Corda-Node (Moothart & Escoto, 2018)

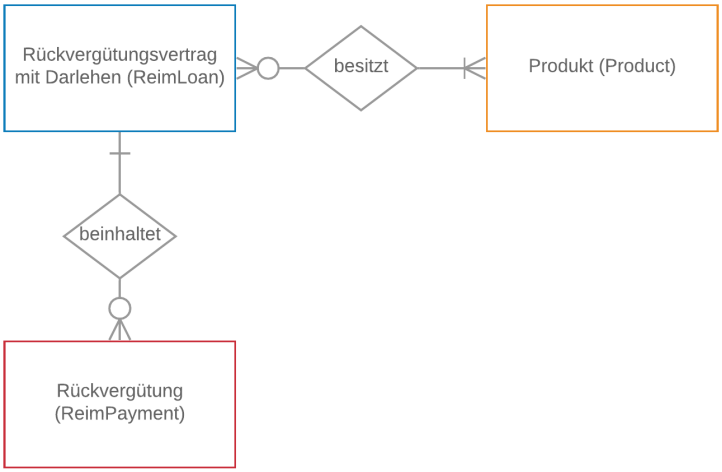

Abbildung 22: ERD des Corda «ReimLoanSchema»

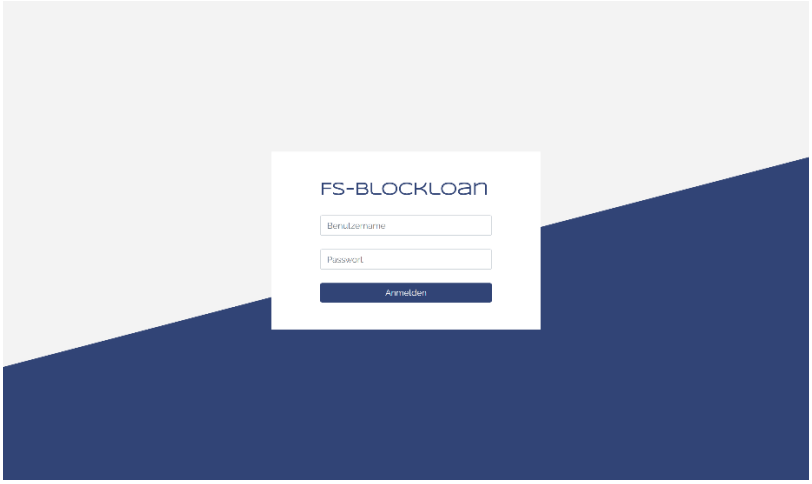

Abbildung 23: Screenshot Log-in-Seite

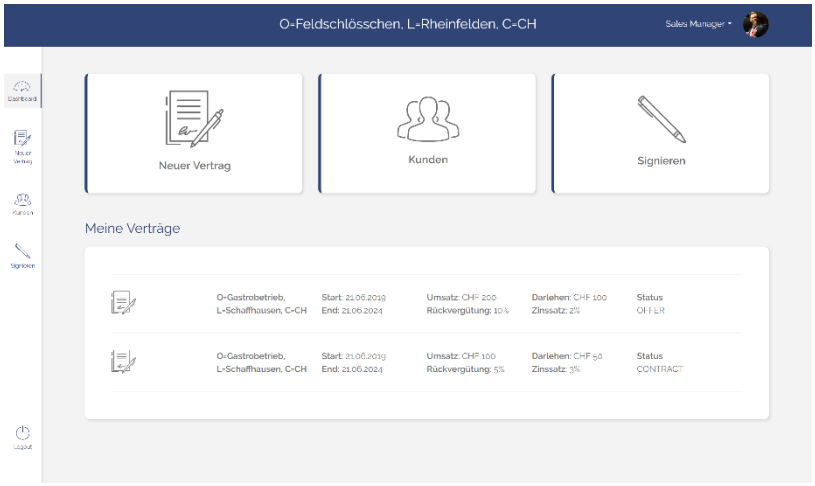

Abbildung 24: Screenshot Dashboard Sales Manager + Vorgesetzter

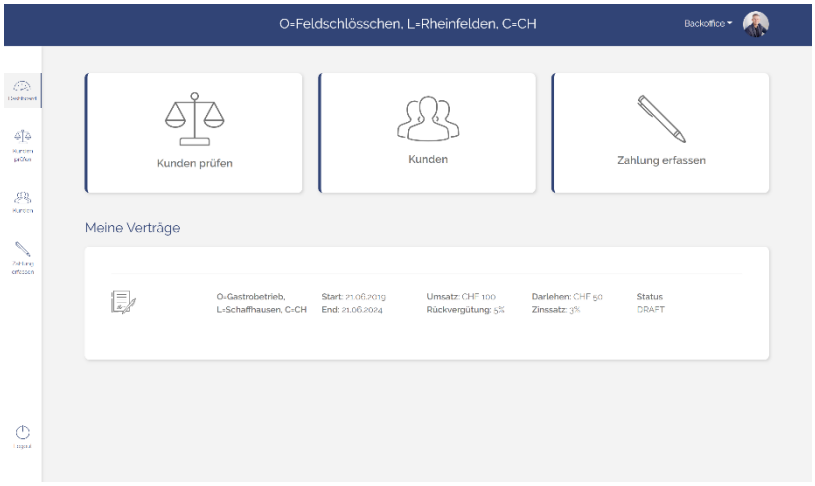

Abbildung 25: Screenshot Dashboard Backoffice

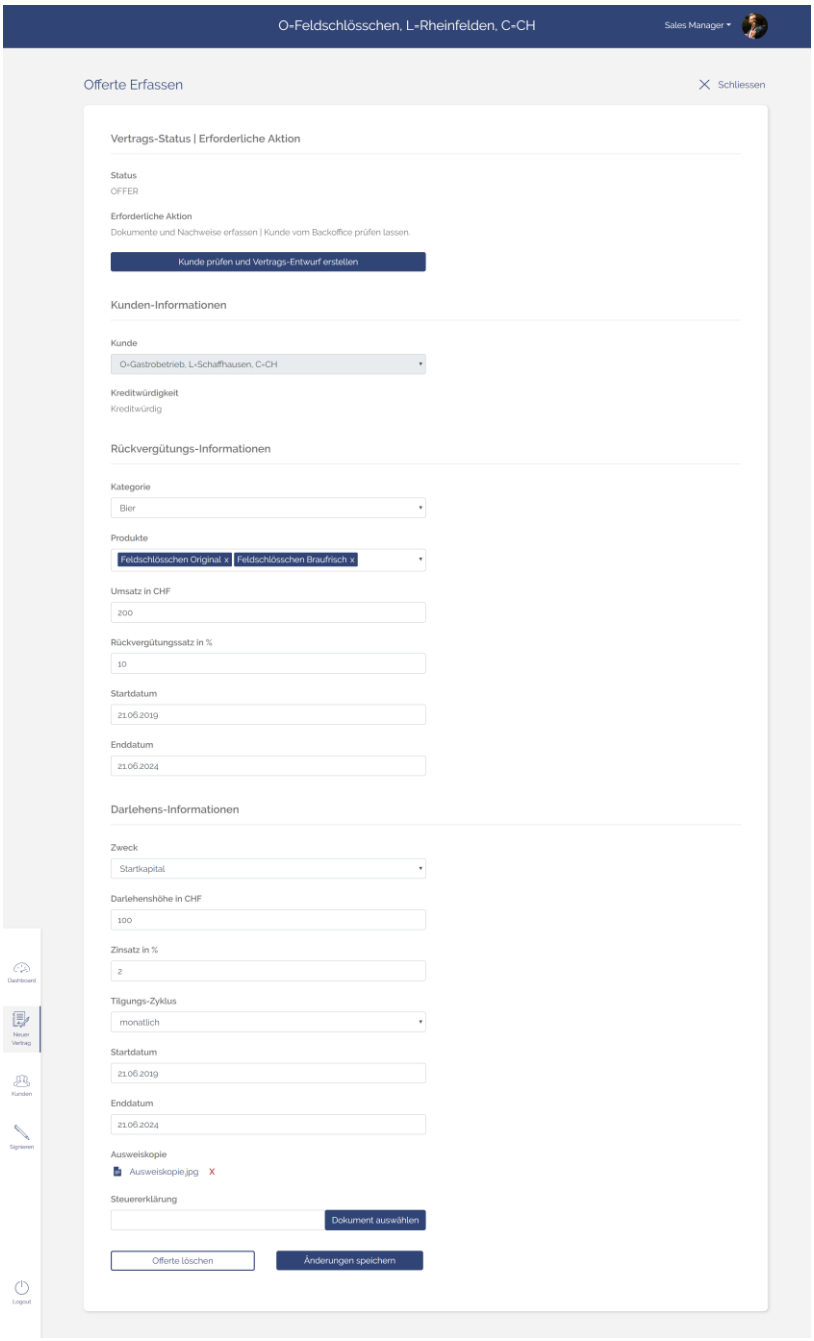

Abbildung 26: Screenshot Eingabeformular eines Vertrags

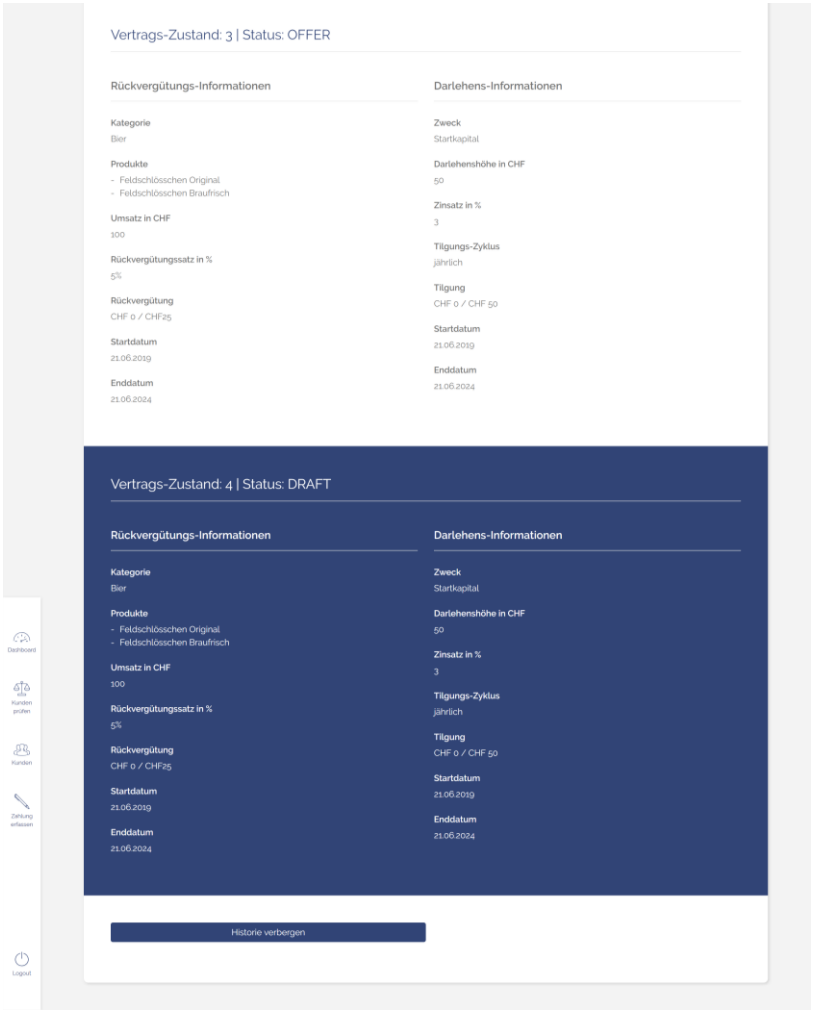

Abbildung 27: Screenshot Historie eines Vertrags

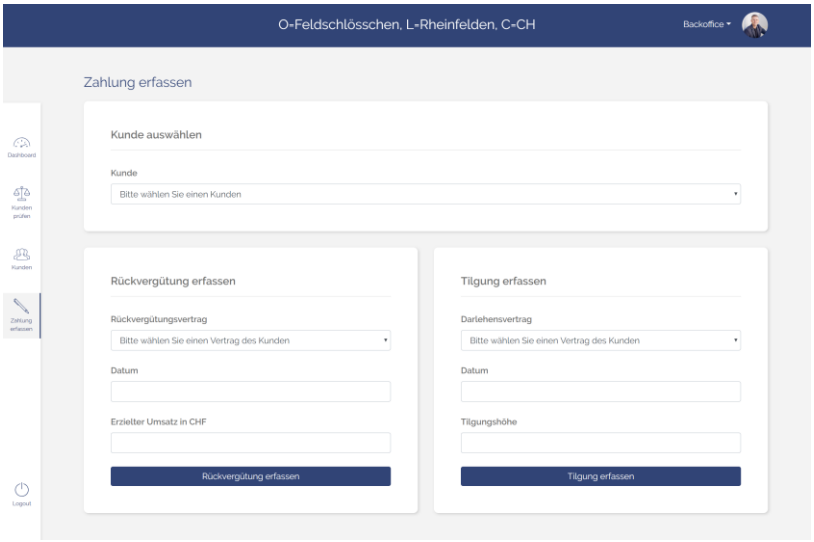

Abbildung 28: Screenshot Maske zur Erfassung von Zahlungen

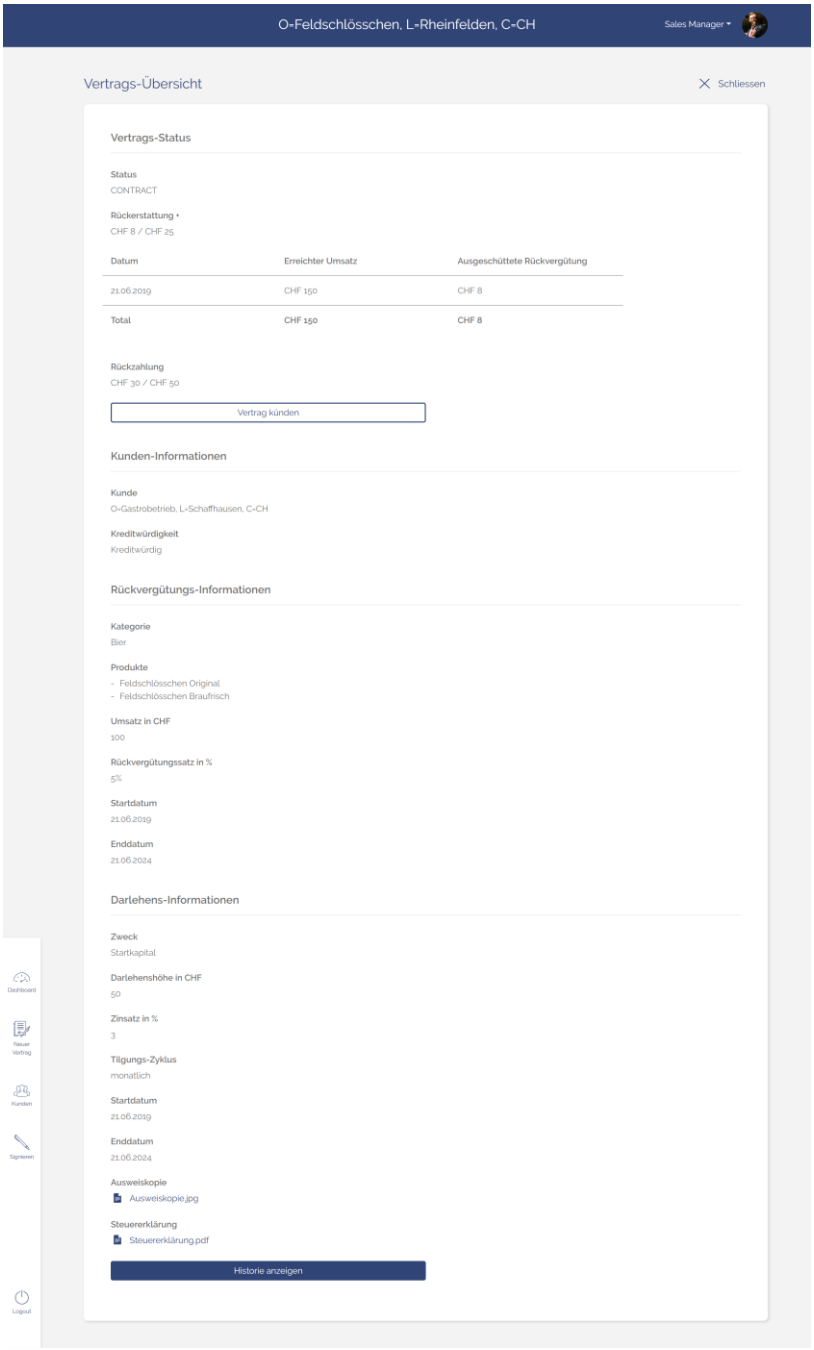

Abbildung 29: Screenshot Vertrags-Übersicht definitiver Vertrag

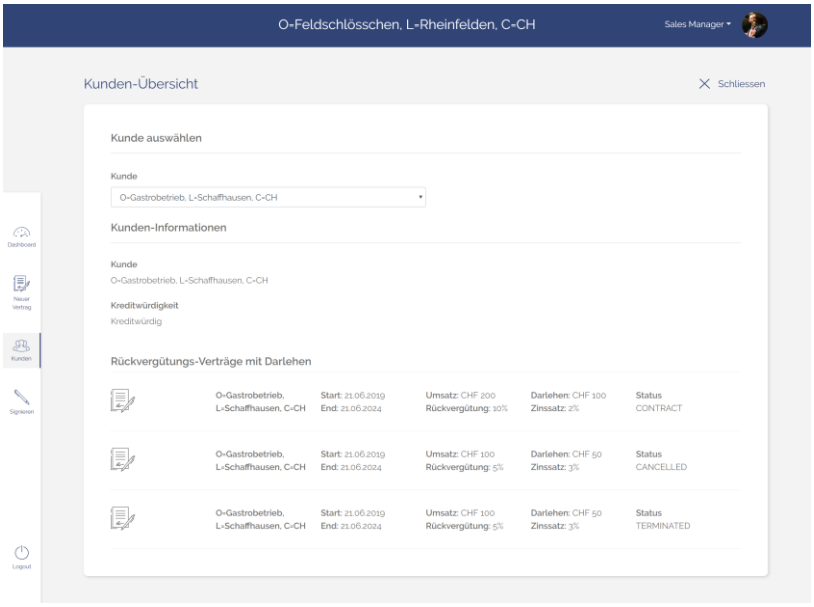

Abbildung 30: Screenshot Kunden-Übersicht

B. Tabellen

Eigenschaft	Öffentliche Blockchain	Konsortium Blockchain	Private Blockchain
Zugang	Permissionless (uneingeschränkt)	Permissioned (eingeschränkt)	Permissioned (eingeschränkt)
Konsensus-Findung	Alle Teilnehmer	Ausgewählte Anzahl Teilnehmer	Meist einzelne Organisation
Privacy / Lese-Berechtigung	Öffentlich	Netzwerkweit / gezielte Rechtevergabe / Transaktionsebene	Netzwerkweit / gezielte Rechtevergabe / Transaktionsebene
Unveränderbarkeit	Manipulationssicher	Fast Manipulationssicher	Fast Manipulationssicher
Transaktionsraten	Tief	Hoch	Hoch
Zentralisierte Datenhaltung	Nein. Hohe Anzahl Nodes in Netzwerk	Nein. Kleinere Anzahl Nodes in Netzwerk	Nein. Kleinere Anzahl Nodes in Netzwerk
Transaktionskosten	Ja. Plattformspezifisch	Keine	Keine

Tabelle 1: Vergleich Blockchain-Typen (Zheng et al., 2018)

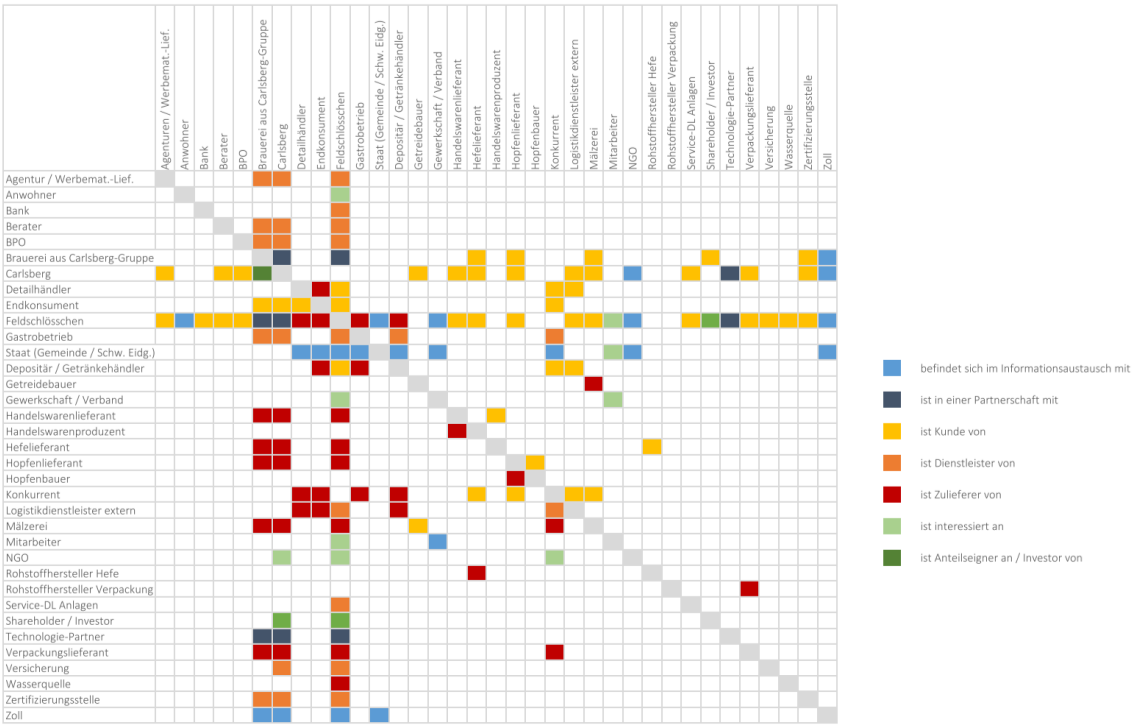

Tabelle 2: Beziehungsmatrix des Business-Ökosystems von Feldschlösschen

Frage	Antwort
Befindet sich die Anwendung in einem Umfeld mit mehreren Stakeholdern?	**Ja**. Beteiligt sind die Gastrobetriebe, die Getränkehändler, die Banken und Feldschlösschen.
Wird eine gemeinsame und konsistente Datenbank benötigt und gibt es Unstimmigkeiten welche Partei die Datenhoheit besitzt?	**Ja.** Es wird eine gemeinsame und sichere Ablage der Verträge benötigt. Keine der Parteien will, dass der Vertrag nur bei der Gegenpartei aufbewahrt wird.
Besteht ein Misstrauensverhältnis zwischen den Parteien (haben Zielkonflikte) oder es soll nicht auf eine vertrauenswürdige Dritt-Partei zurückgegriffen werden (Dritt-Partei soll eliminiert werden)?	**Ja.** Würde zwischen den Parteien kein Misstrauensverhältnis bestehen, würde kein Vertrag benötigt werden. Aufgrund des Zeitaufwandes und der entstehenden Kosten soll jedoch nicht auf eine Drittpartei zurückgegriffen werden.
Soll mit digitalen statt physischen Vermögenswerten gearbeitet werden?	**Ja.** Die digitalen Vermögenswerte (Transaktionsinhalte) stellen die Verträge dar, die nicht mehr papierbasiert aufbewahrt werden sollen.
Müssen / dürfen Daten unveränderbar gespeichert werden?	**Ja.** Die Unveränderbarkeit der Verträge und damit verbundenen Transaktionen stellt eine wichtige Anforderung der Plattform dar.
Wird eine nahtlose Rückverfolgbarkeit der Entstehung und Mutation der Daten benötigt?	**Ja.** Ziel von Feldschlösschen ist es, die Verträge und Ereignisse während der Vertragslaufzeit auf Kundenebene zurückverfolgen zu können.
Ist der Grundrahmen des Vertrags beständig?	**Ja.** Der Grundrahmen des Vertrags wird mit der Offerte gesetzt und nicht mehr verändert.
Werden vertragliche Beziehungen verwaltet oder Vermögenswerte ausgetauscht?	**Ja.** Dies bezieht sich darauf, dass die Speicherung der Daten transaktionsorientiert abläuft und ein Wertaustausch mit beliebigem Inhalt stattfinden muss. Durch den Austausch von Verträgen ist diese Anforderung gegeben.
Werden hohe Transaktionsraten benötigt?	**Nein.** Die benötigen Transaktionsraten belaufen sich nicht über die aktuellen Höchstraten der Blockchain-Technologie.
Müssen grosse Mengen an Daten gespeichert werden?	**Nein.** In der Blockchain wird nur der Vertragsinhalt gespeichert. Dazugehörige Nachweisdokumente müssen aufgrund der Limitierungen der Blockchain in einer Datenbank gespeichert werden.
Werden unterschiedliche (Lese- / Schreib-) Rechte benötigt?	**Ja.** Nicht alle Teilnehmer sollen die Transaktionen der anderen Teilnehmer sehen können.
Sollen die Transaktionen öffentlich sein?	**Nein.** Dies Anforderung von Feldschlösschen, da sensible Informationen ausgetauscht werden.
↓ Ergebnis ↓	
Private oder Konsortium- (permissioned) Blockchain	

Tabelle 3: Evaluation Blockchain-Typ

Eigenschaften	Hyperledger Fabric	R3 Corda	Quorum	Ethereum
Organisation	Linux Foundation	R3 Konsortium	JP Morgan	Etherium Community
Use Case	Industrieübergreifend, vorallem Unternehmens-Applikationen im B2B Bereich	Industrieübergeifend, jedoch spezialisiert für Bedürfnisse der Finanzindustrie	Industrieübergreifend	Industrieübergreifend, vorallem im B2C Bereich
Privacy / Typ	Private / Permissioned	Private / Permissioned	Private / Permissioned	Public / Permissionless
Smart Contract Unterstützung	Ja	Ja	Ja	Ja
Programmiersprache	GoLang, Java	Kotlin, Java	Solidity	Solidity
Konsensus Algorithmus	- Trusted Solo - PBFT - Kafka - Zukünftig RAFT	- Trusted Solo - BFT - RAFT	- Istanbul BFT - RAFT	POW (in Zukunft POS)
Währung / Token	- Keine eigene Währung - Erstellung eigener Tokens via Chaincode	«Fungible Assets» über States	Keine	Ether
Transaktionskosten	Nein	Nein	Nein	Ja
Transaktionsraten in Transaktionen pro Sekunde	> 1000 tps	~ 170 tps	wenige 100 tps	~ 20 tps
Zugriffskontrolle	Auf Organisationsebene und über verschiedene Channels (Art Sub-Netzwerk)	Auf Organisationsebene	Über Attributs- oder Rollendefinition in Smart Contracts	Keine
Transaktions-Privacy	Innerhalb von Channels	Transaktion ist nur für Transaktions-Parteien sichtbar	Öffentliche und Private Transaktionen über Definition in Smart Contract	Alle Transaktionen sind öffentlich
Set-up	●	●	●	●
Dokumentation	●	●	●	●
Release-Fähigkeit / Aktualität	●	●	●	●
Sicherheit	●	●	●	●
Verwaltung	●	●	●	●

● Sehr gut ● Gut ● Mittel ● Schlecht

Tabelle 4: Vergleich Eigenschaften ausgewählter Blockchain-Plattformen (Maple & Jackson, 2019; Höfelmann & Sander, 2019)

	Vorteile	Nachteile
Hyperledger Fabric	- standardisierte Baukastenlösung für eine breite Vielfalt von Anwendungsfällen - hohe Skalierbarkeit und Datensicherheit - Erstellung von privaten Channels für das Informations- und Rechtemanagement - grosse Community, starke Weiterentwicklung und gute Dokumentation - Möglichkeit zur Erstellung einer für das Netzwerk gültigen Kryptowährung oder Tokens - Projekt der gemeinnützigen Linux-Foundation	- modulare Architektur kann noch nicht alle Spezialfälle und Anwendungsbereiche des Finanzsektors abdecken - aufwendiger Aufbau des Netzwerks - Verwaltung der Channels sowie der Rechte (Zertifikate) kann sehr komplex werden
Corda	- Anwendungsfokus auf Services im Finanzbereich - hohe Skalierbarkeit und Datensicherheit - Integration von klassischen Rechtsdokumenten - grosse Community, starke Weiterentwicklung und gute Dokumentation - einfachere Umsetzung des Netzwerkaufbaus und der Rechtevergabe als bei Hyperledger Fabric	- zentrale Kontrolle durch R3 Konsortium (private Unternehmen) - starke Bedeutung von Notary-Node (zuständig für Konsensus in der Blockchain) führt zu Rezentralisierung

Quorum	- Verbindung der Vorteile einer öffentlichen (Ethereum) und privaten Blockchain - ausgerichtet für Finanz-Services - hohe Skalierbarkeit und Datensicherheit - Zugriff auf Kryptowährung Ether	- kleinere Community und schlechtere Dokumentation - kleinere Datenmengen können auf der Blockchain gespeichert werden, da jeder Node eine Kopie der gesamten Chain besitzt - Solidity ist eine weniger verbreitete Programmiersprache als Java oder Kotlin

Tabelle 5: Vergleich Vor- und Nachteile ausgewählter Blockchain-Plattformen (Sandner, 2018)

Vertragsinformationen	Nachweisdokumente
Rückvergütungsvertrag	**Rückvergütung- und Darlehensvertrag**
Kunde (definiert durch Corda X500 Name) *(F)*	Ausweiskopie *(F)*
Produkte-Kategorie (Bier, Mineral) *(F)*	Steuererklärung *(F)*
3 Produkte pro Kategorie *(F)*	
Umsatz in CHF *(F)*	
Rückvergütungssatz in % *(F)*	
Totaler Rückvergütungsbetrag (Umsatz x Rückvergütungssatz x Laufzeit)	
Rückvergütungen *(Z)* - Datum *(Z)* - Erreichter Umsatz *(Z)* - Rückvergütungsbetrag in CHF	
Startdatum Rückvergütungsvertrag *(F)*	
Enddatum Rückvergütungsvertrag *(F)*	
Darlehensvertrag	
Darlehens-Zweck *(F)*	
Darlehenshöhe in CHF *(F)*	
Zinssatz in % *(F)*	
Tilguns-Zyklus (jährlich, monatlich, individuell) *(F)*	
Startdatum Darlehensvertrag *(F)*	
Enddatum Darlehensvertrag *(F)*	
Bereits bezahlte Tilgung des Darlehens *(Z)*	
Status des Vertrags	
(F) = Eingabeformular Vertrag	*(Z) = Eingabeformular Zahlungen*

Tabelle 6: Informationen und Dokumente Rückvergütungs- und Darlehensvertrag

	Ziele	Lösung durch Plattform
Feldschlösschen	- Rückverfolgbarkeit erhöhen / Nachweis im Rechtsstreit verbessern *(f)* - Transparenz erhöhen *(f)* - physische Verträge verhindern *(f)* - einzig gültige Version von Verträgen / Single-Version-of-truth *(f)* - Datensicherheit - Personalaufwand reduzieren *(f)* - Prozess-Durchlaufgeschwindigkeit erhöhen *(f)* - Prozessablauf standardisieren und kontrollieren *(f)* - langfristige Kundenbindung *(w)* - geringe Verluste aus Darlehensverträgen *(w)* - Vertragseinhaltung / zeitige Zahlung *(w)* - Sortiments-Exklusivität *(w)*	- unveränderbare Erfassung aller Informationen auf Kundenebene - für alle Beteiligten sind alle Daten sichtbar - digitale Verträge und Signatur - dezentralisierte Datenbank - Sicherung der Verträge durch Blockchain - Automatisierung Prozesse / digitale Ablage - paralleles Signieren und Verhandeln - Automatisierung Prozesse - keine - Minderung der Personalkosten - Automatisierung Prozesse / Auto-Enforcement - keine
Gastrobetrieb	- einfache Darlehensaufnahme (f) - Sicherheit der sensiblen Daten - einzig gültige Version / Single-Version-of-truth *(f)* - Transparenz erhöhen (Konditionen, Rückvergütung, Zahlungsvorgang) (f) - freie Wahl des Sortiments (w) - niedrige Bindung an die Brauerei (w) - Vertragseinhaltung / zeitige Rückvergütung - vorteilhafte Konditionen (w)	- standardisierter Prozess, schnelle Rückmeldung, GUI gestützte Führung - durch Blockchain gesichert - dezentralisierte Datenbank - unveränderbare Erfassung aller Informationen/ für alle Beteiligten sind alle Daten sichtbar - keine - keine - Automatisierung Prozesse / Auto-Enforcement - niedrigere Aufwände bei der Brauerei, Rückverfolgbarkeit der Loyalität und Vertragserfüllung
Legende: *(f)* = funktional | *(w)* = wirtschaftlich | Zielübereinstimmung | Zielkonflikt		

Tabelle 7: Ziele und Motive der Business-Ökosystem-Teilnehmer

Feldschlösschen	
Sales Manager	
SM1	Als Sales Manager möchte ich für einen Kunden eine neue Offerte eröffnen können, sodass der Prozess der Informationserfassung beginnen kann.
SM2	Als Sales Manager möchte ich die mit *(F)* markierten Informationen aus Tabelle 6 für einen Rückvergütungs- mit Darlehensvertrag erfassen können.
SM3	Als Sales Manager möchte ich, nachdem alle Informationen erfasst wurden, einen Vertragsentwurf generieren können.
SM4	Als Sales Manager möchte ich eine Offerte oder einen Vertrags-Entwurf löschen können.
SM5	Als Sales Manager möchte ich den aktuellen Status eines Vertrages einsehen können, sodass der Kunde über den Entscheid informiert werden kann.
SM6	Als Sales Manager möchte ich nach Kunden suchen können (Limitierung Prototyp: Es existiert nur ein Kunden-Node).
SM7	Als Sales Manager möchte ich alle Verträge zu einem Kunden anzeigen können.
SM8	Als Sales Manager möchte ich einen Vertrag bearbeiten und signieren können, sofern er sich innerhalb der «goldenen Regeln» befindet.
SM9	Als Sales Manager möchte ich alle Verträge angezeigt bekommen, welche ich zu signieren habe.
SM10	Als Sales Manager möchte ich die Historie einer Vertragsentstehung zurückverfolgen können.
SM11	Als Sales Manager möchte ich alle unter dem Vertrag vorgenommenen Rückvergütungen und Tilgungen einsehen können.
SM12	Als Sales Manager möchte ich einen Vertrag kündigen können.
Backoffice	
B1	Als Backoffice möchte ich alle Verträge anzeigen können, bei denen ich den Kunden zu prüfen habe.
B2	Als Backoffice möchte ich die gesamte Historie eines Vertrages ansehen können.
B3	Als Backoffice möchte ich einem Antrag entnehmen können, ob ein Kunde kreditwürdig ist oder nicht.
B4	Als Backoffice möchte ich einen Kunden suchen können, sodass die zum Kunden gehörenden Verträge aufgelistet werden (Limitierung Prototyp: Es existiert nur ein Kunden-Node).
B5	Als Backoffice möchte ich eine Zahlung oder eine Tilgung vornehmen können
B6	Als Backoffice möchte ich einen Vertrag terminieren können, sodass keine Änderungen am Vertrag mehr vorgenommen werden können (Funktion die zur Simulation benötigt wird).
Vorgesetzter	
V1	Als Vorgesetzter möchte ich die gleichen Funktionalitäten besitzen wie der Sales Manager.
V2	Als Vorgesetzter möchte ich einen Vertrag signieren können, auch wenn er ausserhalb der «goldenen Regeln» liegt.
Gastrobetrieb	
G1	Als Gastrobetrieb möchte ich einen Zugang erhalten, mit dem ich meine Verträge und ihre Status einsehen kann.
G2	Als Gastrobetrieb möchte ich die mit *(F)* markierten Informationen aus Tabelle 6, für einen Rückvergütungs- mit Darlehensvertrag erfassen können.
G3	Als Gastrobetrieb möchte ich einen Vertrag signieren können.
G4	Als Gastrobetrieb möchte ich die gesamte Historie eines Vertrags ansehen können.
G5	Als Gastrobetrieb möchte ich einen Vertrag kündigen können.

Tabelle 8: User Stories

System	
Blockchain	
S1	In Corda sollen drei repräsentative Parteien aufgesetzt werden. Der Notary Node, gehostet von Feldschlösschen, ein Gastrobetrieb und Feldschlösschen als Partei. Die einzelnen Akteure innerhalb von Feldschlösschen sollen keinen eigenen Node darstellen. Als unterzeichnende Partei existiert deshalb nur Feldschlösschen.
S2	In Corda soll ein CorDapp mit einem State, einem Contract und mit Flows zur Erfüllung der Anforderungen in Tabelle 8 erstellt werden. Das gesamte CorDapp bezieht sich auf einen Rückvergütung- mit Darlehensvertrag als einzelnen Vertrag mit zwei Teilen.
S3	Aus Corda sollen die manuellen Signaturen aus dem Frontend mittels HTTP-Request überprüft werden, bevor ein Vertrag erstellt werden kann.
S4	In Corda soll eine zeitbedingte Beispiels-Automatisierung für die Terminierung eines Vertrags implementiert werden.
Node Clients	
S5	Die Node Clients sollen eine RPC-Verbindung zum zugehörigen Node aufbauen.
S6	Die Node Clients sollen die Funktionalitäten der Corda-Flows für das Backend mittels einer API zur Verfügung stellen.
S7	Die Node Clients sollen alle sich im Netzwerk befindenden Peers zu einem Node auslesen können und diese Daten mittels API dem Backend zur Verfügung stellen.
S8	Die Node Clients sollen alle zu einem Node gehörenden Verträge auslesen können und diese Daten mittels API dem Backend zur Verfügung stellen.
S9	Die Node Clients sollen die gesamte Historie eines Vertrags von einem Node auslesen können und diese Daten mittels API dem Backend zur Verfügung stellen.
Backend	
S10	Das Backend soll eine Datenbank umfassen, in welcher Vertrags-Metadaten und die Nachweisdokumente gespeichert werden können.
S11	Das Backend soll über eine REST API vom Frontend angesteuert werden können.
S12	Das Backend soll HTTP-Requests an die Node Clients senden können, damit die im CorDapp festgelegten Flows ausgeführt werden können.
S13	Das Backend soll über HTTP-Requests die Verträge von dem gewünschten Node auslesen können und deshalb als Integrationsebene der Node Clients dienen.
S14	Das Backend soll über HTTP-Requests die Kreditwürdigkeit des Gastrobetriebs von der Moneyhouse-Mock-API auslesen können.
S15	Das Backend soll die Transformation der Vertragsobjekte zwischen Blockchain und Frontend vornehmen.
Frontend	
S16	Das Frontend soll eine Benutzeroberfläche für die Rollen Sales Manager, Backoffice, Vorgesetzter und Gastrobetrieb bieten.
S17	Das Frontend soll sämtliche Daten über die Backend-API mittels HTTP-Request beziehen.
S18	Das Frontend soll verschiedene Eingabeformulare für die in Tabelle 6 definierten Informationen bereitstellen.

Tabelle 9: Anforderungen System

Eigenschaft	Typ
Ausgeber	*Party (Corda-Klasse)*
Kunde	*Party (Corda-Klasse)*
Kategorie	*String*
Produkte	*Liste von Produkten (Eigens definierte Klasse)*
Vereinbarter Umsatz	*Long*
Rückvergütungssatz	*Long*
Rückvergütungs-Startdatum	*Date*
Rückvergütungs-Enddatum	*Date*
Rückvergütungsbetrag	*Long*
Rückerstattungen	*Liste von Rückerstattungen (Eigens definierte Klasse)*
Darlehens-Zweck	*Sting*
Darlehens-Betrag	*Long*
Darlehens-Zinssatz	*Long*
Tilgungs-Zyklus	*String*
Darlehens-Startdatum	*Date*
Darlehens-Enddatum	*Date*
Ausweiskopie	*String*
Steuererklärung	*String*
Tilgung	*Long*
Status	*Vertragsstatus (Eigens definierte Enum-Klasse)*
Linear-Id	*UniqueIdentifier (Corda-Klasse)*
Teilnehmer	*Liste von Parties (Ausgeber und Kund)*

Tabelle 10: Eigenschaften eines «ReimLoanStates»

Command Name	Verwendung / Übersetzung
CreateOffer	*Offerte erstellen*
UpdateContent	*Vertragsinhalt aktualisieren*
CreateDraft	*Vertrags-Entwurf aus einer Offerte erstellen*
CreateContract	*Definitiver Vertrag von einem Vertrags-Entwurf erstellen*
LoanSettle	*Tilgung*
Reimbursement	*Rückerstattung*
Terminate	*Terminieren des Vertrags über das Enddatum*
CancelState	*Setzen des Gekündigt-Status (weiterhin transformierbar)*
Cancel	*Kündigen des Vertrages (Transaktion ohne erneut verwendbaren Output-State - nicht mehr transformierbar)*

Tabelle 11: Commands eines «ReimLoanContracts»

API Basis-URL Feldschlösschen Node		API Basis-URL Gastrobetrieb Node
http://localhost:50005/api		http://localhost:50006/api

Endpunkt	Methode	Beschreibung
/me	*GET*	Auslesen der eigenen Node-Identität
/peers	*GET*	Auslesen aller sich im Netzwerk befindender Nodes
/reim-loans	*GET*	Auslesen aller Rückvergütungs- mit Darlehensverträgen auf die der aktuelle Node zugriff besitzt
/reim-loans/my	*GET*	Auslesen aller Rückvergütungs- mit Darlehensverträgen bei dem der aktuelle Node der Aussteller des Vertrages ist
/reim-loans/{role}	*GET*	Auslesen aller Rückvergütungs- mit Darlehensverträgen basierend auf der Rolle des Frontend-Benutzers
/reim-loans/{linearIdString}/history	*GET*	Auslesen der gesamten Historie eines Rückvergütungs- mit Darlehensvertrag über die Linear-ID
/reim-loan/create-offer	*POST*	Erstellen einer neuen Offerte
/reim-loan/create-draft	*POST*	Erstellen eines Vertrags-Entwurfes basierend auf einer Offerte
/reim-loan/create-contract	*POST*	Erstellen eines definitiven Vertrages basierend auf einem Vertrags-Entwurf
/reim-loan/{linearIdString}/pay-reim	*POST*	Hinzufügen einer neuen Rückvergütung
/reim-loan/settle	*POST*	Hinzufügen einer neuen Tilgung
/reim-loan/terminate	*POST*	Erstellen eines terminierten Vertrages (nicht idempotent)
/reim-loan/delete	*POST*	Löschen einer Offerte oder eines Vertrag-Entwurfs. Die Methode POST wurde gewählt, weil ein Vertrag in Corda nicht gelöscht, sondern nur auf CONSUMED gesetzt werden kann und die Funktion ausserdem nicht idempotent ist. Für den Benutzer verhält sich das System aber wie bei einer Löschung.
/reim-loan/cancel	*POST*	Künden eines Vertrages. Dabei wird der Vertrag-Status auf CANCELLED gesetzt und deshalb ein «gekündigter» Vertrag erstellt (nicht idempotent).
/reim-loan/update-content	*PUT*	Aktualisieren des Vertragsinhaltes bis zum Zeitpunkt eines definitiven Vertrages

Tabelle 12: API-Endpunkte Node Clients

CONTRACT_META_ENTITY		
Attribut	**Typ**	**Beschreibung**
CONTRACT_META_ID	*INTEGER(10)*	Primärschlüssel Contract_Meta_Entity
LINEAR_ID	*VARCHAR(255)*	Linear_ID des in der Blockchain gespeicherten ReimLoanState
SIGN_FS	*BOOLEAN(1)*	Boolean, ob der Ausgeber (Feldschlösschen) den Vertrag bereits signiert hat
SIGN_CUSTOMER	*BOOLEAN(1)*	Boolean, ob der Kunde (Gastrobetrieb) den Vertrag bereits signiert hat
CANCEL_FS	*BOOLEAN(1)*	Boolean, ob der Ausgeber (Feldschlösschen) die Vertrags-Kündigung bereits signiert hat
CANCEL_CUSTOMER	*BOOLEAN(1)*	Boolean, ob der Kunde (Gastrobetrieb) die Vertrags-Kündigung bereits signiert hat
CUSTOMER_CHECKED	*BOOLEAN(1)*	Boolean, ob der Kunde (Gastrobetrieb) bereits vom Backoffice geprüft wurde
PASSPORT_NAME	*VARCHAR(255)*	Name der hochgeladenen Ausweiskopie
PASSPORT_BASE	*CLOB(2147483647)*	Base64 enkodierter String der Ausweiskopie
TAX_DECLARATION_NAME	*VARCHAR(255)*	Name der hochgeladenen Steuererklärung
TAX_DECLARATION_BASE	*CLOB(2147483647)*	Base64 enkodierter String der Steuererklärung

Tabelle 13: Struktur «CONTACT_META_ENTITY»

API Basis-URL Backend		
http://localhost:8080/api		
Endpunkt	**Methode**	**Beschreibung**
/reim-loans/{client}/{role}	*GET*	Auslesen aller Rückvergütungs mit Darlehensverträgen basierend auf dem Client (Port) und der Rolle des im Frontend angemeldeten Benutzers
/reim-loans/{linearIdString}/history	*GET*	Auslesen der gesamten Historie eines Rückvergütungs- mit Darlehensvertrag über die Linear-ID
/reim-loan/{linearId}/issuer-signature	*GET*	Gibt zurück, ob der Vertrag im Forntend vom Ausgeber bereits signiert wurde
/reim-loan/{linearId}/customer-signature	*GET*	Gibt zurück, ob der Vertrag im Forntend vom Kunden bereits signiert wurde
/reim-loan/{linearId}/issuer-cancel-signature	*GET*	Gibt zurück, ob die Kündigung des Vertrags im Forntend vom Ausgeber bereits signiert wurde
/reim-loan/{linearId}/customer-cancel-signature	*GET*	Gibt zurück, ob die Kündigung des Vertrags im Forntend vom Kunden bereits signiert wurde
/reim-loan/create-offer	*POST*	Erstellen einer neuen Offerte
/reim-loan/create-draft	*POST*	Erstellen eines Vertrags-Entwurfes basierend auf einer Offerte
/reim-loan/create-contract	*POST*	Erstellen eines definitiven Vertrages basierend auf einem Vertrags-Entwurf
/reim-loan/{linearIdString}/pay-reim	*POST*	Hinzufügen einer neuen Rückvergütung
/reim-loan/settle	*POST*	Hinzufügen einer neuen Tilgung
/reim-loan/terminate	*POST*	Erstellen eines terminierten Vertrages (nicht idempotent)
/reim-loan/delete	*POST*	Löschen einer Offerte oder eines Vertrag-Entwurfs. Die Methode POST wurde gewählt, weil ein Vertrag in Corda nicht gelöscht, sondern nur auf CONSUMED gesetzt werden kann und die Funktion ausserdem nicht idempotent ist. Für den Benutzer verhält sich das System aber wie bei einer Löschung.
/reim-loan/cancel	*POST*	Künden eines Vertrages. Dabei wird der Vertrag-Status auf CANCELLED gesetzt und deshalb ein «gekündigter» Vertrag erstellt (nicht idempotent).
/reim-loan/update-content	*PUT*	Aktualisieren des Vertragsinhaltes bis zum Zeitpunkt eines definitiven Vertrages
/reim-loan/check-customer	*PUT*	Aktualisieren der Meta-Vertragsdaten, ob der Kunde vom Backoffice geprüft wurde

Tabelle 14: API-Endpunkte Backend

C. Code-Blöcke

blockloan/workflow-kotlin/build.gradle

```
// Definition des Feldschlösschen Node in dem Gradle-CordForm-Plugin
node {
    name "O=Feldschloesschen,L=Rheinfelden,C=CH"
    p2pPort 10004
    rpcSettings {
        address("localhost:10005")
        adminAddress("localhost:10006")
    }
    rpcUsers = [[user: "user1", "password": "test", "permissions": ["ALL"]]]
    configFile = "workflows-kotlin/src/main/resources/node-feldschloesschen.conf"
}
```

Code-Block 1: Definition Corda-Node via CordForm-Gradle-Plugin

blockloan/contracts-kotlin/src/main/kotlin/com.example.schema/ReimLoanSchema.kt

```
// Entität für ein Produkt.
@CordaSerializable
@Entity
@Nullable
@Table(name = "product")
data class Product(

        @Id
        @GeneratedValue(strategy = GenerationType.IDENTITY)
        @Column(name = "product_Id")
        var productId: Long = 0,

        @Column(name = "article_number")
        var productNr: Long,

        @Column(name = "product_name")
        var productName: String

) : Serializable{
    // Default constructor.
    constructor(): this(0, 0, "")
}
```

Code-Block 2: Corda Schema Entität für ein Produkt eines Rückvergütungsvertrags

blockloan/contracts-kotlin/src/main/kotlin/com.example.schema/ReimLoanState.kt

```
// Mit dieser Funktion kann eine neue Tilgung zum bisherigen Tilgungs-Betrag addiert werden.
// Der neue kopierte State (z. B. für einen neuen Output-State) wird zurückgegeben.
fun settleLoan(amount: Long) = copy(loanSettlement = this.loanSettlement.plus(amount))
```

Code-Block 3: Funktion zum Erstellen des neuen Output-States während einer Tilgung

blockloan/contracts-kotlin/ src/main/kotlin/com.example.schema/ReimLoanContract.kt

```kotlin
// Bedinungen zum Create-Offer (Erstellung Offerte) Command
is Commands.CreateOffer -> requireThat {

    // Auslesen Output-State
    val outputReimLoan = tx.outputStates.single() as ReimLoanState
    // Generieren aktuelles Datum
    val currentDate = Date.from(Instant.now())
    // Generieren eines CordaX500Name aus Strings.
    val feldschloesschen = CordaX500Name(
     organisation = "Feldschloesschen", locality = "Rheinfelden", country = "CH")
    "Es darf kein Input Rückvergütungsvertrag vorhanden sein." using (tx.inputs.isEmpty())
    "Es darf nur ein Output-State generiert werden." using (tx.outputs.size == 1)
    "Feldschlösschen darf keinen Rückvergütungsvertrag aufnehmen." using (outputReimLoan.customer.name !=
     feldschloesschen)
    "Es muss mindestens eine Kategorie angegeben werden." using (outputReimLoan.category.isNotEmpty())
    "Bei der Erstellung der Offerte kann noch keine Tilgung stattgefunden haben." using
     (outputReimLoan.loanSettlement == 0L)
}
```

Code-Block 4: Auszug Bedingungen für den CreateOffer-Command

blockloan/clients/ src/main/kotlin/com.example.server/NodeRPCConnection.kt

```kotlin
/**
 * Instanziert eine RPC-Verbindung zu einem Node.
 *
 * Die RPC-Verbindung wird auf den Werten des `build.gradle`-File und dem gestarteten Client gebildet.
 */
@Component
open class NodeRPCConnection(
        @Value("\${$CORDA_NODE_HOST}") private val host: String,
        @Value("\${$CORDA_USER_NAME}") private val username: String,
        @Value("\${$CORDA_USER_PASSWORD}") private val password: String,
        @Value("\${$CORDA_RPC_PORT}") private val rpcPort: Int) {

    lateinit var rpcConnection: CordaRPCConnection
        set
    lateinit var proxy: CordaRPCOps
        set

    @PostConstruct
    fun initialiseNodeRPCConnection() {

        val rpcAddress = NetworkHostAndPort(host, rpcPort)
        val rpcClient = CordaRPCClient(rpcAddress)
        val rpcConnection = rpcClient.start(username, password)
        proxy = rpcConnection.proxy
    }

    @PreDestroy
    fun close() {
        rpcConnection.notifyServerAndClose()
    }
}
```

Code-Block 5: Instanziierung einer RPC-Verbindung zu einem Corda-Node