Institut für Parallele und Verteilte Systeme

Universität Stuttgart
Universitätsstraße 38
D–70569 Stuttgart

Bachelorarbeit Nr. 65

Ein Marktplatz für TOSCA

Matthias Fetzer

Studiengang:	Softwaretechnik
Prüfer/in:	Prof. Dr.-Ing. habil. Bernhard Mitschang
Betreuer/in:	Dipl. -Inf. Uwe Breitenbücher
	Dipl. -Inf. Oliver Kopp
Beginn am:	16. April 2013
Beendet am:	16. Oktober 2013
CR-Nummer:	H.3.5, J.0

Kurzfassung

Die TOSCA-Spezifikation („Topology and Orchestration Specification for Cloud Applications“) erlaubt es, plattformunabhängig Service-Komponenten und ihre Beziehungen untereinander zu beschreiben. Mit OpenTOSCA wurde eine erste quelloffene Laufzeitumgebung für die TOSCA-Spezifikation an der Universität Stuttgart entwickelt.

Im Rahmen dieser Bachelorarbeit werden, aufbauend auf OpenTOSCA, die Konzepte eines Marktplatzes für TOSCA erläutert und eine erste Implementierung entwickelt sowie in OpenTOSCA integriert.

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

Verzeichnis der Listings

1 Einleitung

Durch die zunehmende Verbreitung von Cloud-Anbietern und -Applikationen [PM10], wuchs der Bedarf an einheitlichen Standards zum Betreiben und Spezifizieren von Cloud-Lösungen. Durch die Spezifikation des TOSCA-Standards („Topology and Orchestration Specification for Cloud Applications“) und die ersten Implementierungen von TOSCA-Laufzeitumgebungen, wurde ein erster Grundstein mittels einheitlicher Standards gelegt.

Motivation

Durch den TOSCA-Standard existiert nun eine Möglichkeit, um plattformunabhängig Topologien zu beschreiben und zu instanziieren. Daher liegt der Wunsch nahe, diese Abstraktion auch an Kunden zu verkaufen. Deshalb benötigen Anbieter eine Möglichkeit ihre Produkte über einen Marktplatz anzubieten. Dieser Marktplatz kann wiederum von Kunden verwendet werden, um diese Produkte zu beziehen.

Ziel dieser Bachelorarbeit ist es, die Konzepte eines Marktplatzes für TOSCA zu erarbeiten und eine erste Implementierung dessen zu erstellen. Weiterhin soll dieser Marktplatz in OpenTOSCA integriert werden.

Aufbau

Dieses Dokument ist in folgende Abschnitte gegliedert:

Kapitel 2 – Grundlagen

In diesem Kapitel werden die Grundlagen für diese Arbeit erläutert. Dies beinhaltet die Themen TOSCA, OpenTOSCA, Winery, das CloudCycle–Ökosystem und elektronische Marktplätze.

Kapitel 3 – Funktionale Anforderungen

In diesem Kapitel werden die funktionalen Anforderungen an einen Marktplatz für TOSCA beschrieben.

Kapitel 4 – Verwandte Arbeiten

In diesem Kapitel werden Arbeiten vorgestellt, welche thematisch mit dieser Bachelorarbeit verwandt sind. Es werden vier Shop-Systeme vorgestellt und evaluiert, ob diese sich als Marktplätze für TOSCA eignen.

Kapitel 5 – Konzept und Architektur

Im fünften Kapitel werden die zugrunde liegenden Konzepte des Marktplatzes und dessen Architektur erläutert. Weiterhin wird auf verschiedene Ansätze bezüglich der Oberfläche eingegangen. Zum Ende des Kapitels werden Anwendungsfälle im Marktplatz formuliert.

Kapitel 6 – Implementierung

In diesem Kapitel wird die Implementierung des Marktplatzes erläutert. Wichtige Datentypen sowie Vorgänge werden dargestellt.

Kapitel 7 – Zusammenfassung und Ausblick

Im letzten Kapitel werden die Ergebnisse dieser Arbeit zusammengefasst, sowie ein zukünftiger Ausblick gegeben.

2 Grundlagen

In diesem Kapitel werden Begriffe und Technologien erläutert, welche für diese Bachelorarbeit von Bedeutung sind. Zuerst wird auf die TOSCA-Spezifikation, OpenTOSCA, Winery sowie das CloudCycle–Ökosystem eingegangen, welches das Zusammenspiel der verschiedenen Akteure und Komponenten erläutert. Später wird auf den Begriff des elektronischen Marktplatzes eingegangen.

2.1 Topology and Orchestration Specification for Cloud Applications (TOSCA)

Der Topology and Orchestration Specification for Cloud Applications-Standard (nachfolgend „TOSCA“ genannt) ist ein von der OASIS (Organization for the Advancement of Structured Information Standards, https://www.oasis-open.org/) entwickelter Standard zur Beschreibung von Cloud-Applikationen und deren Verwaltung.

Mittels TOSCA kann die Struktur einer Cloud-Anwendung (ihre Topologie) als auch die Orchestrierung und die Verwaltung dieser mittels Plänen definiert werden. Abbildung 2.1 zeigt eine solche graphisch dargestellte Topologie.

Im weiteren Abschnitt wird auf die Rollenverteilung in der TOSCA-Spezifikation und auf das Cloud Service Archive (CSAR) genauer eingegangen. Für weitere Informationen sei auf die TOSCA-Spezifikation [OAS13] verwiesen. Eine kompakte Übersicht der Konzepte von TOSCA kann in [BBKL14] gefunden werden.

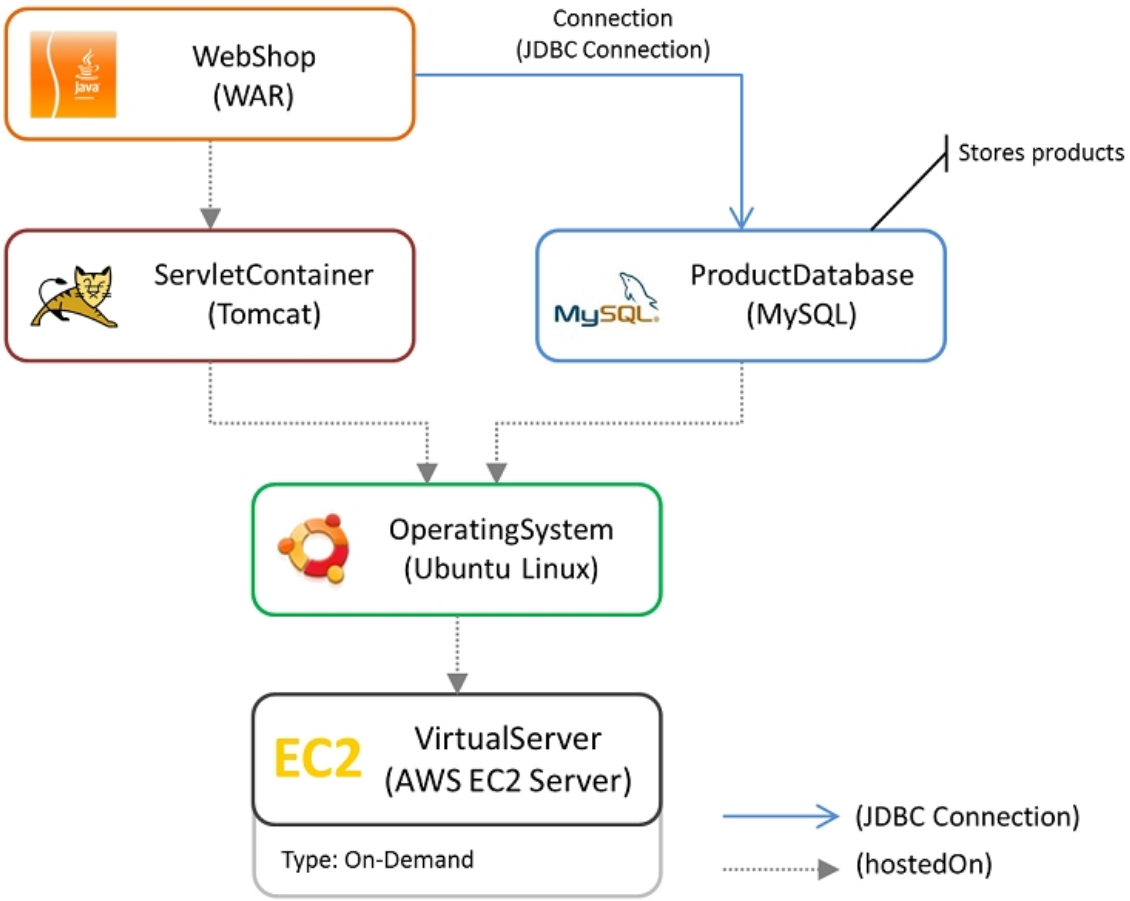

Abbildung 2.1: Topologie eines Java-Web-Shops der mittels Vino4TOSCA [BBK+12] visuell dargestellt wurde. Übernommen von http://www.vino4tosca.org/

2.1.1 TOSCA-Rollen

Als eine Einführung beschreibt die TOSCA-Spezifikation [OAS13] kurz eine mögliche Rollenverteilung im Hinblick auf Cloud-Services und inwiefern diese Rollen von TOSCA profitieren können. Die in der Spezifikation genannten und definierten Rollen sind:

- Cloud Service Consumer – Der Cloud Service Consumer nutzt IT-Services geschäftlich. Er profitiert von TOSCA, steht jedoch nicht in direktem Kontakt mit der Spezifikation.
- Cloud Service Developer – Das Hauptgeschäftsfeld des Cloud Service Developers ist das Entwickeln von Cloud-Diensten. Er benutzt TOSCA um zu beschreiben, wie seine Dienste instanziiert und betrieben werden.
- Cloud Service Provider – Das Hauptgeschäftsfeld des Cloud Service Providers ist das Betreiben von Cloud-Diensten, welche von Cloud Service Developern entwickelt wurden. Er benutzt TOSCA, um Kunden seine Infrastruktur zur Verfügung zu stellen.

Diese Rollen dienten bei der Erstellung der Konzepte dieser Bachelorarbeit als Orientierung.

2.1.2 Cloud Service Archive (CSAR)

Mittels TOSCA spezifizierte Cloud-Applikationen werden als CSAR (Cloud Service Archive, nachfolgend CSAR-Datei genannt) verteilt. Der in diesem Dokument beschriebene Marktplatz hat das Ziel, eine Plattform für den Vertrieb von CSAR-Dateien darzustellen. Daher wird das Konstrukt einer CSAR-Datei nachfolgend genauer ausgeführt.

Technischer Hintergrund:

Bei einer CSAR-Datei handelt es sich um ein ZIP-Archiv [PKW07], mit der Dateiendung „csar“. Eine CSAR-Datei beinhaltet alle benötigten Dokumente, Artefakte und Pläne oder Referenzen zu diesen.

2.2 OpenTOSCA-Container

OpenTOSCA [BBH+13] ist eine webbasierte Open Source Java-Implementierung einer TOSCA-Laufzeitumgebung. Sie ermöglicht das Installieren und Instanziieren von CSAR-Dateien (siehe Abschnitt 2.1.2).

Eine prototypische Implementierung des OpenTOSCA-Containers wurde im Rahmen eines Studienprojektes an der Universität Stuttgart im Oktober 2012 fertiggestellt [OTS13].

OpenTOSCA spielt im Zusammenhang dieser Bachelorarbeit vorallem bei der Integration in den OpenTOSCA-Container eine Rolle (siehe Abschnitt 6.4).

2.3 Winery

Bei Winery [KBBL13] handelt es sich um ein Web-basiertes Modellierungswerkzeug zum Erstellen von TOSCA-Topologien und deren Management-Plänen. Winery unterstützt die visuelle Notation „Vino4TOSCA“ [BBK+12] und ist seit kurzem ein Projekt der Eclipse Foundation[1].

2.4 Das CloudCycle–Ökosystem

Niehues et al. [NKP13] stellen den Inhalt eines Cloud-Ökosystems vor. Ein Cloud-Ökosystem besteht demnach aus unterschiedlichen Komponenten und Akteure. Es werden zuerst die Akteure und Komponenten in einem solchen System definiert und erklärt, sowie die Beziehungen zwischen jenen dargestellt.

Sowohl die Akteure als auch die Komponenten dienten bei der Entwicklung des Konzeptes (Kapitel 5) als Vorlage und Orientierung.

2.5 Elektronische Marktplätze

Da es sich in dieser Arbeit um das Konzept und die Implementierung eines Marktplatzes für TOSCA handelt, muss vorab der Begriff des elektronischen Marktplatzes (im Folgenden „Marktplatz“ genannt) konkretisiert werden. Unter einem elektronischen Marktplatz versteht man eine Vertriebsplattform, welche das Kaufen und Verkaufen von digitalen Gütern ermöglicht. Vergleichbar ist dies mit dem Bild eines klassischen Marktplatzes, auf welchem verschiedene Anbieter verschiedenen Kunden ihre Waren anbieten, nach Gabler Wirtschaftslexikon [Gab13].

[1] https://projects.eclipse.org/projects/soa.winery

3 Funktionale Anforderungen

Um eine Integration des Marktplatzes mit dem OpenTOSCA-Ökosystem zu gewährleisten, muss eine Reihe von Kriterien erfüllt werden:

1. Integration mit OpenTOSCA-Container: Es muss möglich sein, den Marktplatz in den OpenTOSCA-Container zu integrieren. Ebenso müssen (OpenTOSCA)-Metadaten (vgl. Abschnitt 6.2.2) im Marktplatz abgelegt und abgerufen werden können.
2. Lizenz: Die Lizenz des Marktplatzes muss kompatibel zu der Apache 2.0 Lizenz[1] des OpenTOSCA-Ökosystems sein, damit der Marktplatz ohne Lizenzproblematik integriert werden kann.
3. Programmiersprache: Da der OpenTOSCA-Container in Java implementiert ist, muss der Marktplatz ebenfalls in Java implementiert werden, um die Homogenität im OpenTOSCA-Ökosystem zu wahren.

[1] http://www.apache.org/licenses/LICENSE-2.0.html

4 Verwandte Arbeiten

In diesem Kapitel werden Arbeiten vorgestellt, welche thematisch mit der Aufgabenstellung dieser Bachelorarbeit verwandt sind. Da es sich bei der Aufgabenstellung um die Entwicklung eines Marktplatzes für TOSCA handelt, werden im Folgenden vier popupläre, quelloffene, Shop-Systeme vorgestellt [Tor12] und im Hinblick auf den Einsatz als TOSCA-Marktplatz nach folgenden Kriterien betrachtet:

1. Lizenzierung: Unter welcher Lizenz ist das Shop-System veröffentlicht? Ist diese kompatibel zur Apache 2.0 Lizenz? Ist diese Lizenz Kategorie A[1] oder Kategorie B[2]?
2. Erweiterbarkeit: Ist das Shop-System erweiterbar? Können neue Funktionen mittels vordefinierter Schnittstellen hinzugefügt werden, oder muss der Quellcode des Shop-Systems direkt geändert werden?
3. Webservice-API: Gibt es eine Webservice-API? Gibt es die Möglichkeit der automatischen Interaktion mit dem Shop-System mittels einer vordefinierten Webservice-API?
4. ESD[3]: Gibt es eine Möglichkeit der digitalen Lieferung von Produkten? Kann dies auf CSAR-Dateien übertragen werden ?
5. Programmiersprache: In welcher Programmiersprache wurde das Shop-System implementiert?

Die Überprüfung der Erweiterbarkeit, sowie des Vorhandenseins einer Webservice-API dient zur Abklärung einer möglichen Integration in das bestehende OpenTOSCA-Ökosystem.

[1] http://www.apache.org/legal/resolved.html#category-a
[2] http://www.apache.org/legal/resolved.html#category-b
[3] Electronic Software Delivery

4.1 Magento

Magento[4] ist ein Shop-System der Firma Magento, einer 100-%-igen Tochter von eBay[5]. Magento ist eine professionelle Open Source eCommerce Lösung mit einem erheblichen Funktionsumfang. Magento wird sowohl in einer Community Edition als auch in einer Enterprise Edition angeboten. Da nur die Community Edition quelloffen ist, wird nachfolgend nur auf diese eingegangen.

Im Folgenden wird Magento anhand der obig definierten Kriterien bewertet.

1. Lizenzierung: Magento Community Edition ist unter der OSL 3.0[6] lizenziert. Diese Lizenz ist Kategorie-B-Kompatibel zur Apache 2.0 Lizenz.
2. Erweiterbarkeit: Magento kann mittels Modulen in PHP erweitert werden.
3. Webservice-API: Mit Magento kann mittels einer SOAP- und einer REST-API interagiert werden.
4. ESD: Magento bietet die Möglichkeit der digitalen Lieferung von Produkten.
5. Programmiersprache: Magento ist in PHP implementiert.

Fazit

Magento wirkt auf den ersten Blick sehr vielversprechend, vor allem aufgrund der OSL3.0 Lizenz, der Erweiterbarkeit und der Webservice-API. Leider ist Magento in PHP implementiert und passt somit nicht in die Java-Umgebung des OpenTOSCA-Ökosystems.

[4] http://www.magentocommerce.com/
[5] https://www.ebay.com
[6] Open Software License v.3.0 – http://opensource.org/licenses/OSL-3.0

4.2 OpenCart

OpenCart[7] ist eine freie Open Source eCommerce Plattform für Onlinehändler. OpenCart bietet eine professionelle und zuverlässige Grundlage zum Aufbau eines erfolgreichen Onlineshops.

Im Folgenden wird OpenCart anhand der obig definierten Kriterien bewertet.

1. Lizenzierung: GPL v3[8] – Die GPL v3 ist nur bedingt zur Apache 2.0 Lizenz kompatibel. Apache 2.0 Lizenzierter Code kann in GPLv3 Projekten verwendet werden, jedoch nicht vice versa [Apa12].
2. Erweiterbarkeit: Opencart kann mittels Modulen in PHP erweitert werden.
3. Webservice-API: Für Opencart ist keine offizielle Webservice-API vorhanden.
4. ESD: Opencart bietet die Möglichkeit der digitalen Lieferung von Produkten.
5. Programmiersprache: Opencart ist in PHP implementiert.

Fazit

OpenCart ist nicht mit dem OpenTOSCA-Ökosystem kompatibel, da die GPL v3 Lizenz nur bedingt kompatibel ist. Weiterhin ist OpenCart in PHP implementiert und passt somit nicht in die Java-Umgebung des OpenTOSCA-Ökosystems.

4.3 osCommerce

osCommerce[9] ist eine Open Source eCommerce Lösung, die ständig durch eine Open Source Gemeinde weiterentwickelt wird.

Im Folgenden wird osCommerce anhand der obig definierten Kriterien bewertet.

1. Lizenzierung: GPL v2[10] – Die GPL v2 Lizenz ist nicht kompatibel mit der Apache 2.0 Lizenz.
2. Erweiterbarkeit: osCommerce kann mittels Modulen in PHP erweitert werden.
3. Webservice-API: osCommerce bietet keine Webservice-API an.
4. ESD: osCommerce bietet die Möglichkeit der digitalen Lieferung von Produkten.
5. Programmiersprache: osCommerce ist in PHP implementiert.

[7] http://www.opencart.com/
[8] GNU General Public License v3 – http://www.gnu.org/licenses/gpl-3.0.html
[9] http://www.oscommerce.com/
[10] GNU General Public License v2 – http://www.gnu.org/licenses/gpl-2.0.html

Fazit

osCommerce ist durch die eingesetzte GPL v2 Lizenz nicht kompatibel mit der von OpenTOSCA eingesetzten Apache 2.0 Lizenz. Somit kann osCommerce nicht als ein Marktplatz für OpenTOSCA genutzt werden. Weiterhin ist osCommerce in PHP implementiert und kann somit nicht in die OpenTOSCA-Umgebung integriert werden.

4.4 PrestaShop

PrestaShop[11] ist ein modular aufgebautes Shop-System mit umfangreichen Funktionen.

Im Folgenden wird PrestaShop anhand der obig definierten Kriterien bewertet.

1. Lizenzierung: Lizenzierung: PrestaShop ist unter der OSL 3.0 lizenziert. Diese Lizenz ist Kategorie-B-Kompatibel zur Apache 2.0 Lizenz.
2. Erweiterbarkeit: PrestaShop kann mittels Modulen in PHP erweitert werden.
3. Webservice-API: PrestaShop bietet eine RESTful API an.
4. ESD: PrestaShop bietet die Möglichkeit der digitalen Lieferung von Produkten.
5. Programmiersprache: PrestaShop ist in PHP implementiert.

Fazit

PrestaShop ist von der OSL 3.0 Lizenzierung her kompatibel zur, von OpenTOSCA eingesetzten, Apache 2.0 Lizenz. PrestaShop ist jedoch ebenfalls in PHP implementiert und kann somit nicht im OpenTOSCA-Ökosystem eingesetzt werden.

[11] http://www.prestashop.com/

4.5 Zusammenfassung

Zwei der vier untersuchten Shop-Systeme eignen sich aufgrund ihrer Lizenz nicht für einen TOSCA-Marktplatz: OpenCart und osCommerce. Die Lizenzen dieser stehen in Konflikt mit der Apache 2.0 Lizenz des OpenTOSCA-Projektes. Im Falle einer GPLv3 Lizenz könnten bestehende OpenTOSCA-Bibliotheken in das Shop-System integriert werden. Dort erstellter Programmcode könnte jedoch nicht in OpenTOSCA verwendet werden. Die zwei anderen Shop-Systeme (Magento und PrestaShop) stehen unter der OSL v3 Lizenz und sind nur Kategorie-B-kompatibel zur Apache 2.0 Lizenz.

Alle vier Shop-Systeme sind in PHP implementiert. Da der Marktplatz später mit OpenTOSCA integriert werden soll, ist eine Java-Implementierung von Vorteil, um vorhandenen Quellcode und bereits vorhandenes Wissen weiterzuverwenden. Weiterhin soll im kompletten OpenTOSCA-Ökosystem nur eine Programmiersprache eingesetzt werden.

Da keines der genannten Shop-Systeme alle Anforderungen an einen Marktplatz für OpenTOSCA erfüllt (vgl. Kapitel 3) muss ein neuer Marktplatz implementiert werden.

Im weiteren Verlauf werden das Konzept und die Architektur (Kapitel 5) als auch die Implementierung (Kapitel 6) eines Marktplatzes für OpenTOSCA auf Java-Basis erläutert.

5 Konzept und Architektur

In diesem Kapitel werden die grundlegenden Konzepte dieser Arbeit erläutert, sowie die daraus resultierende Architektur beschrieben. Weiterhin wird die mögliche Integration des Marktplatzes in bestehende Systeme (bspw. OpenTOSCA) veranschaulicht und erläutert.

5.1 Generelle Einordnung

Um eine generelle Einordnung dieser Arbeit in ein bestehendes TOSCA-System zu ermöglichen, werden der Ist- sowie der Soll-Zustand, als auch die beteiligten Akteure beschrieben.

5.1.1 Rollen im Marktplatz

Zum besseren Verständnis des später beschriebenen Ist- und Soll-Zustandes werden nachfolgend die Rollen in einem TOSCA-System erläutert. Diese Rollen orientieren sich an denen der TOSCA-Spezifikation [OAS13] als auch an den Rollen des CloudCycle–Ökosystems [NKP13]. Unabhängig vom aktuellen Zustand ergeben sich im Marktplatz die folgenden vier Rollen:

1. **CSAR-Anbieter**: Der Anbieter einer CSAR-Datei erstellt diese und möchte sie seinen potentiellen Kunden anbieten. Er ist für die Erstellung und die Distribution seiner CSAR-Dateien zuständig. Ein CSAR-Anbieter kann mehrere CSAR-Dateien anbieten und auch mehrere Kunden bedienen.
2. **CSAR-Nutzer**: Der CSAR-Nutzer möchte beliebig viele CSAR-Dateien von beliebig vielen CSAR-Anbietern herunterladen und nutzen.
3. **Cloud-Betreiber**: Der Cloud-Betreiber betreibt die benötigte Infrastruktur, um die vom CSAR-Anbieter erstellten Anwendungen zu betreiben. Auf diese Rolle wird im Verlauf dieser Arbeit jedoch nicht weiter eingegangen.
4. **Marktplatzbetreiber**: Der Marktplatzbetreiber betreibt den eigentlichen Marktplatz, welchen der CSAR-Anbieter und der CSAR-Nutzer benutzen.

Akteure im System können beide Rollen einnehmen. Somit kann z.B. ein CSAR-Anbieter auch CSAR-Dateien von anderen CSAR-Anbietern nutzen und entspricht in diesem Fall einem CSAR-Nutzer.

5.1.2 Ist-Zustand

Derzeit ist das Modellierungswerkzeug (Winery) des CSAR-Anbieters, im Falle des OpenTOSCA-Ökosystems, logisch vom TOSCA-Container (OpenTOSCA-Container) des CSAR-Nutzers getrennt. Beide Systeme haben keine Möglichkeit miteinander zu interagieren. Angebotene CSAR-Dateien müssen auf drittem Wege vom CSAR-Anbieter zum CSAR-Nutzer transferiert und vom CSAR-Nutzer manuell eingepflegt werden.

Abbildung 5.1 stellt die Beziehung zwischen einem CSAR-Nutzer und einem CSAR-Anbieter dar. Da jedoch mehrere CSAR-Anbieter und mehrere CSAR-Nutzer vorhanden sind, erhöht sich die Komplexität im gesamten System, wie in Abbildung 5.2 zu sehen ist.

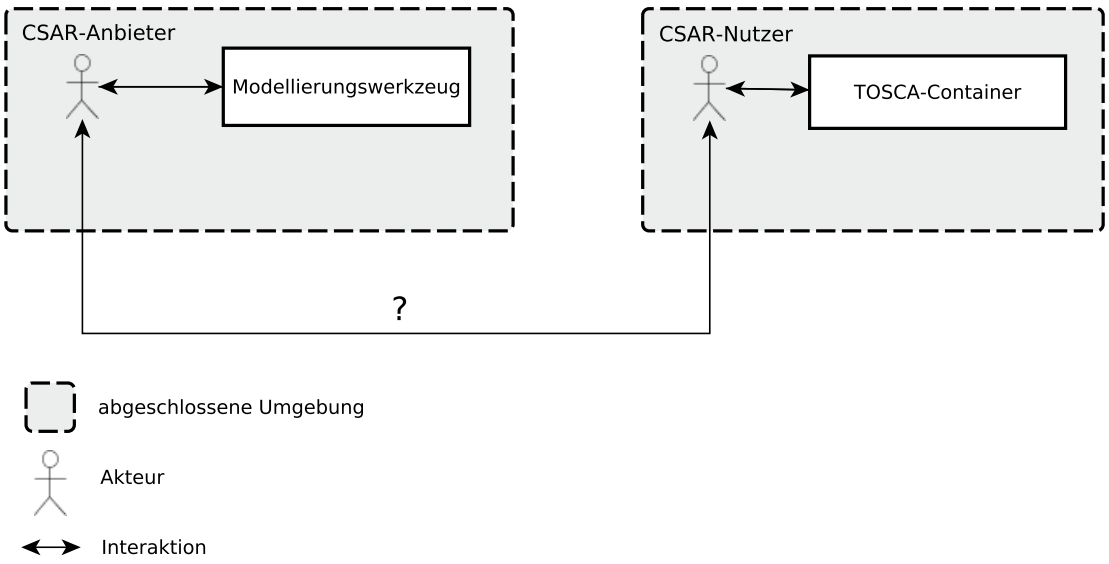

Abbildung 5.1: System ohne Marktplatz

Momentan läuft die Distribution eines CSAR-Paketes folgendermaßen ab:

1. Der Anbieter einer CSAR erstellt diese in seinem Modellierungswerkzeug.
2. Nach der Erstellung muss sich der Anbieter für einen Vertriebskanal entscheiden und seine CSAR (manuell) publizieren und vertreiben.
3. Der CSAR-Nutzer erhält über den vom Anbieter festgelegten Vertriebskanal eine CSAR-Datei und kann diese manuell in seinen TOSCA-Container laden.

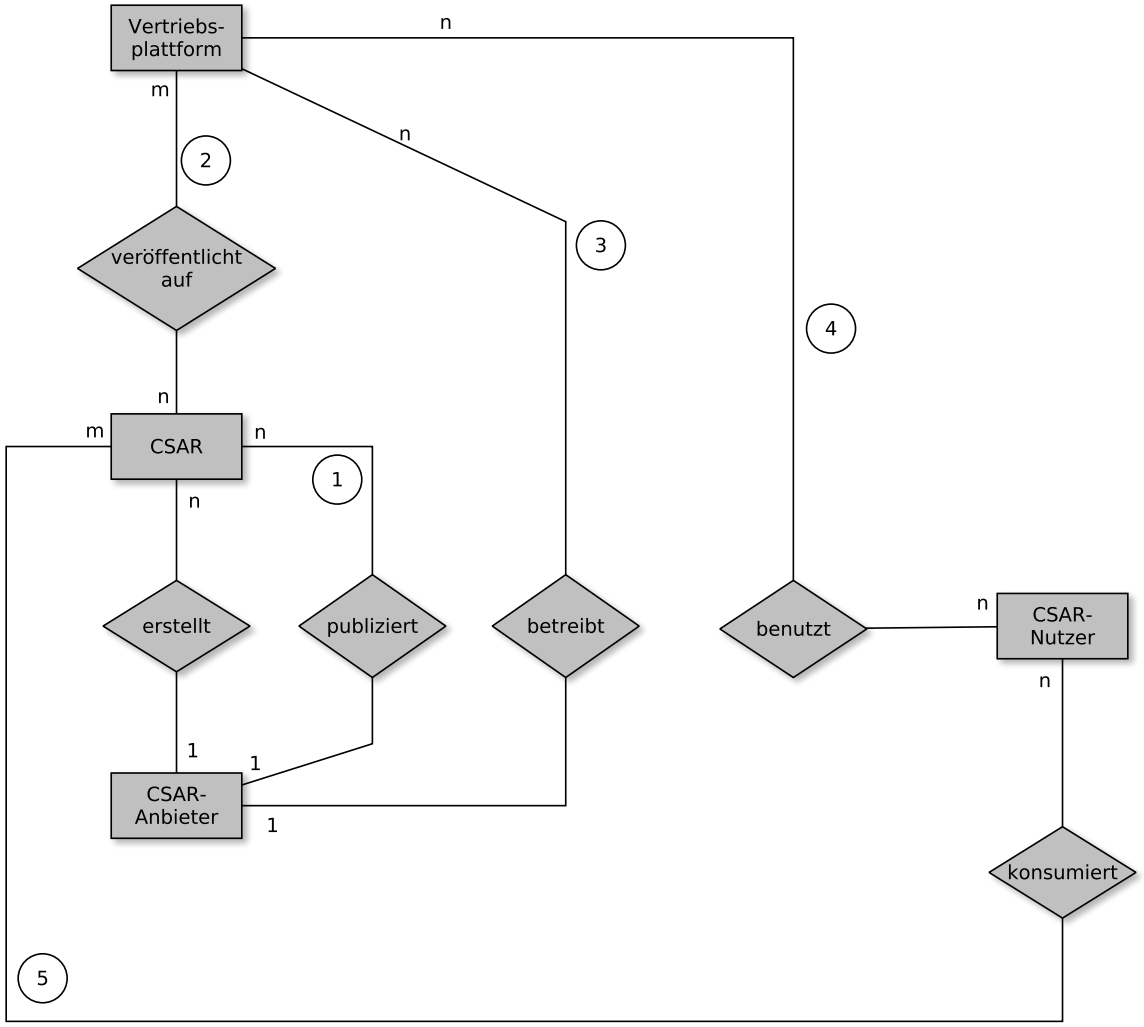

Abbildung 5.2: Relationen zwischen CSAR-Anbieter, Plattform, CSAR und CSAR-Nutzer

Die Kardinalitäten zwischen CSAR-Anbieter, CSAR-Nutzer und CSAR-Datei im derzeitigen Zustand können wie in Abbildung 5.2 beschrieben werden:

- Ein Anbieter publiziert beliebig viele CSAR-Dateien auf beliebig vielen Vertriebsplattformen. Auf einer Vertriebsplattform können beliebig viele Anbieter ihre CSAR-Dateien publizieren. (Abbildung 5.2: ① ②)
- Ein Anbieter betreibt beliebig viele Vertriebsplattformen. (Abbildung 5.2: ③)
- Auf einer Vertriebsplattform werden beliebig viele CSAR-Dateien angeboten. Eine CSAR-Datei wird auf mindestens einer Vertriebsplattform veröffentlicht. (Abbildung 5.2: ②)
- Beliebig viele CSAR-Nutzer benutzen beliebig viele Vertriebsplattformen. (Abbildung 5.2: ④)
- Beliebig viele CSAR-Nutzer benutzen beliebig viele CSAR-Dateien. (Abbildung 5.2: ⑤)

Der Einfachheit halber wird davon ausgegangen, dass sich mehrere Anbieter nicht zusammenschließen, um eine einzige Vertriebsplattform zu betreiben. Aber es wird beachtet, dass ein Anbieter seine CSAR-Dateien auf mehreren Vertriebswegen publiziert.

Aus diesen Gegebenheiten ergeben sich derzeit folgende Nachteile für den Anbieter:

1. Der Anbieter muss sich Gedanken um die Art der Distribution machen und im schlimmsten Falle sich selbst um den Betrieb einer Vertriebsplattform kümmern.
2. Der Anbieter hat keine Möglichkeit, seine CSAR-Dateien einem breiten Publikum direkt zur Verfügung zu stellen, da keine zentrale Möglichkeit, wie beispielsweise ein Marktplatz, existiert.

Abgesehen von den Nachteilen des Anbieters, ergeben sich auch diverse Nachteile für den CSAR-Nutzer:

1. Es gibt keine zentrale Anlaufstelle für CSARs. Somit muss der CSAR-Nutzer manuell nach (neuen) Anbietern und Distributionsplattformen Ausschau halten.
2. Durch die unzähligen Distributionsplattformen der verschiedenen Anbieter, ist es für den CSAR-Nutzer schwierig, stets einen Überblick über die aktuell verfügbaren CSAR-Dateien zu bewahren. (vgl. n- zu m-Beziehungen in Abbildung 5.2)

5.1.3 Soll-Zustand

Um diese Probleme zu adressieren, bietet sich als ein möglicher Lösungsweg eine zentrale Vertriebsplattform in Form eines Marktplatzes an. Dies entspricht dem Zustand, der mittels dieser Arbeit erreicht werden soll.

Mit der Erarbeitung eines Marktplatzes wird eine Verbindungskomponente zwischen einer TOSCA-Laufzeitumgebung und dem Modellierungswerkzeug erreicht. Die dadurch erstellten Verknüpfungen und Beziehungen zwischen CSAR-Anbieter und CSAR-Nutzer werden

in Abbildung 5.3 dargestellt. Die durchgezogenen Pfeile beschreiben die automatisierte Interaktion des Modellierungswerkzeuges und des TOSCA-Containers mit dem Marktplatz. Die gestrichelten Pfeile stellen die manuelle Interaktion mit dem Marktplatz dar.

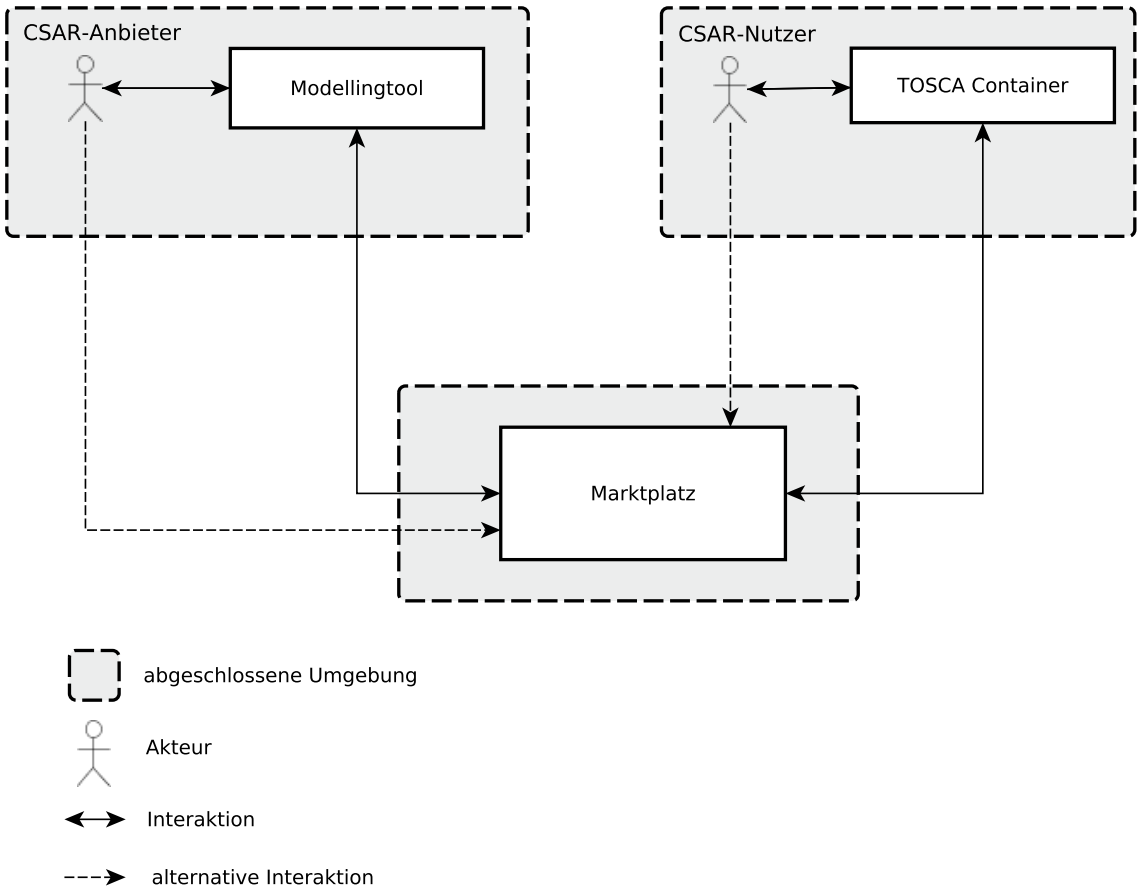

Abbildung 5.3: System mit Marktplatz

In diesem Szenario existiert ein zentraler Marktplatz, in welchem eine Vielzahl an CSAR-Anbietern ihre CSAR-Dateien einer Vielzahl von CSAR-Nutzern zur Verfügung stellen können. Abbildung 5.4 erläutert die Kardinalitäten zwischen Anbieter, CSAR, Marktplatz und Nutzer.

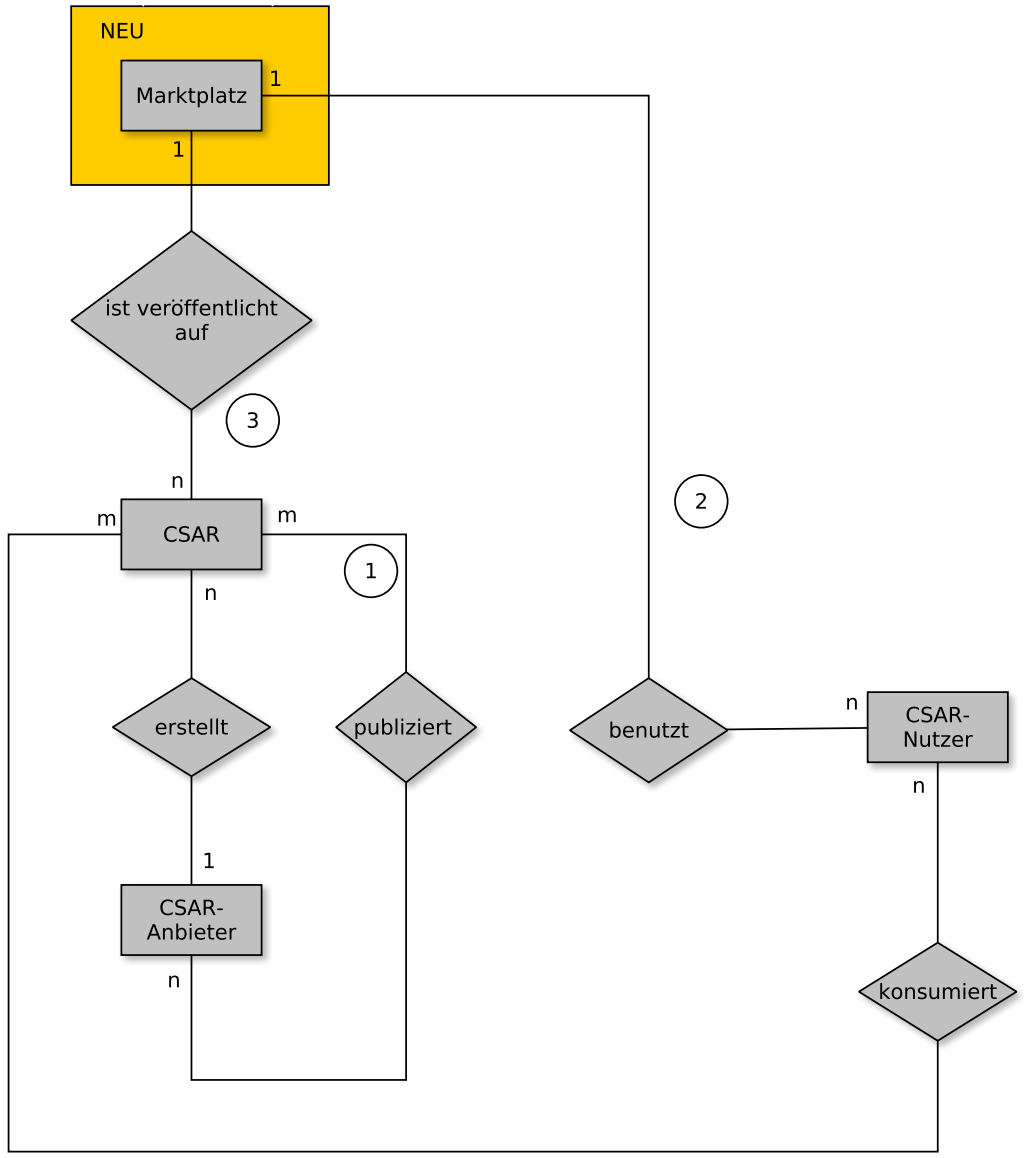

Abbildung 5.4: System mit zentralem Marktplatz

Durch die Einführung eines zentralen Marktplatzes, läuft der Vertrieb einer CSAR-Datei nun wie folgt ab:

1. Der Anbieter erstelle eine CSAR-Datei in seinem Modellierungswerkzeug.
2. Er publiziert diese CSAR-Datei, gegebenenfalls direkt aus dem Modellierungswerkzeug, auf dem Marktplatz.
3. Der CSAR-Nutzer kann direkt in seinem TOSCA-Container die gewünschte CSAR-Datei herunterladen und in seinen Container importieren.

Die Kardinalitäten zwischen CSAR-Anbieter, CSAR-Nutzer und CSAR-Datei im geplanten Zustand, welcher nur einen zentralen Marktplatz (siehe Markierung, Abbildung 5.4) enthält, können wie in Abbildung 5.4 beschrieben werden. Es wird davon ausgegangen, dass ein einzelner zentraler Marktplatz existiert.

- Beliebig viele Anbieter publizieren ihre CSAR-Dateien auf genau einem Marktplatz. Die Veröffentlichung auf weiteren, dritten, Marktplätzen ist weiterhin möglich. (Abbildung 5.4: ①)
- Beliebig viele CSAR-Nutzer nutzen genau einen Marktplatz. (Abbildung 5.4: ②)
- Beliebig viele CSAR-Dateien sind auf genau einem Marktplatz publiziert. (Abbildung 5.4: ③)

Durch diese Änderungen ergeben sich für den Anbieter folgende Vorteile:

1. Der Anbieter erreicht eine maximale Anzahl an CSAR-Nutzern mittels des zentralen Marktplatzes, ohne sich Gedanken um den konkreten Vertrieb machen zu müssen.
2. Der Anbieter muss keine eigene Vertriebsplattform betreiben.

Abgesehen von den Vorteilen des Anbieters ergeben sich auch folgende Vorteile für den CSAR-Nutzer:

1. Durch die zentrale Anlaufstelle muss der CSAR-Nutzer sich nicht mehr über mehrere Vertriebsplattformen informieren.
2. Der CSAR-Nutzer kann nun, ohne einen größeren Aufwand, einen Überblick über alle verfügbaren CSAR-Dateien erhalten.

5.2 Anwendungsfälle

In diesem Abschnitt werden Anwendungsfälle im Marktplatz behandelt, die von dem zu entwickelnden Marktplatz abgedeckt werden müssen. Jeder Anwendungsfall wird tabellarisch dargestellt und erörtert. Jeder Anwendungsfall beinhaltet die beteiligten Akteure (vgl. Abschnitt 5.1.1), die gegebenen Vor- und Nachbedingungen und den Ablauf des Anwendungsfalles. Weiterhin werden mögliche Ausnahmen im Ablauf beschrieben. Folgende Anwendungsfälle werden in diesem Abschnitt vorgestellt:

- Anwendungsfall: Am Marktplatz anmelden – Abschnitt 5.2.1
- Anwendungsfall: CSAR-Dateien suchen – Abschnitt 5.2.2
- Anwendungsfall: CSAR-Datei im Detail anschauen – Abschnitt 5.2.3
- Anwendungsfall: CSAR-Datei herunterladen – Abschnitt 5.2.4
- Anwendungsfall: Eine CSAR-Datei zum Marktplatz hinzufügen – Abschnitt 5.2.5
- Anwendungsfall: Eine CSAR-Datei ändern – Abschnitt 5.2.6
- Anwendungsfall: Eine CSAR-Datei löschen – Abschnitt 5.2.7

5.2.1 Anwendungsfall: Am Marktplatz anmelden

Dieser Anwendungsfall beschreibt den Vorgang der Anmeldung am Marktplatz. Dieser Anwendungsfall ist sowohl für einen CSAR-Nutzer, einen CSAR-Anbieter als auch für den Marktplatzbetreiber gültig. Tabelle 5.1 beschreibt diesen Vorgang mittels eines tabellarischen Anwendungsfalls.

Anwendungsfall:	Am Marktplatz anmelden
Akteure:	CSAR-Nutzer, CSAR-Anbieter oder Marktplatzbetreiber
Vorbedingungen:	• Der Nutzer ist nicht angemeldet. • Der Nutzer besitzt einen gültigen Account.
Ablauf:	1. Der Nutzer klickt auf den Anmeldebutton. 2. Der Nutzer wird auf eine Anmeldeseite weitergeleitet. 3. Der Nutzer bestätigt den Zugriff auf seine Daten. 4. Der Nutzer wird zurück auf den Marktplatz geleitet. 5. Der Nutzer ist angemeldet.
Ausnahmen:	2a) Die Anmeldeseite ist nicht verfügbar. 2a1) Der Nutzer ist nicht angemeldet. 3a) Der Nutzer bestätigt keinen Zugriff auf seine Daten. 3a1) Der Nutzer ist nicht angemeldet.
Nachbedingungen:	Der Nutzer ist am Marktplatz angemeldet.

Tabelle 5.1: Anwendungsfall: Am Marktplatz anmelden

5.2.2 Anwendungsfall: CSAR-Dateien suchen

Dieser Anwendungsfall beschreibt das Suchen/Auflisten von CSAR-Dateien mit Hilfe eines tabellarischen Anwendungsfalls (siehe Tabelle 5.2).

Der CSAR-Nutzer möchte eine CSAR-Liste erhalten. Sofern gewünscht kann der CSAR-Nutzer einen Filter zum Einschränken der zurückgelieferten Liste angeben. Daraufhin liefert der Marktplatz eine passende Liste von CSAR-Dateien zurück.

Anwendungsfall:	CSAR-Dateien suchen
Akteure:	CSAR-Nutzer
Vorbedingungen:	• Der CSAR-Nutzer ist am Marktplatz angemeldet. • Es sind CSAR-Dateien im Marktplatz verfügbar.
Ablauf:	1. Der CSAR-Nutzer legt einen Filter für die gewünschten CSAR-Dateien fest. 2. Der CSAR-Nutzer erhält eine Liste der verfügbaren CSAR-Dateien, welche dem Filter entsprechen.
Ausnahmen:	1a) Der CSAR-Nutzer gibt keinen Filter an. 1a1) Der CSAR-Nutzer erhält eine Liste aller CSAR-Dateien (ungefiltert).
Nachbedingungen:	Der CSAR-Nutzer erhält eine Liste von CSAR-Dateien.

Tabelle 5.2: Anwendungsfall: CSAR-Dateien suchen

5.2.3 Anwendungsfall: CSAR-Datei im Detail anschauen

Folgender Anwendungsfall beschreibt das Betrachten einer CSAR-Datei im Marktplatz. Das Szenario aus Sicht des CSAR-Nutzers ist in Tabelle 5.3 beschrieben.

Anwendungsfall:	CSAR-Datei im Detail anschauen
Akteure:	CSAR-Nutzer
Vorbedingungen:	• Der CSAR-Nutzer ist am Marktplatz angemeldet. • Es sind CSAR-Dateien im Marktplatz verfügbar.
Ablauf:	1. Der CSAR-Nutzer durchsucht die verfügbaren CSAR-Dateien (siehe Anwendungsfall, Tabelle 5.2). 2. Der CSAR-Nutzer wählt eine CSAR-Datei aus und öffnet deren Detail-Ansicht. 3. Der CSAR-Nutzer erhält eine detaillierte Ansicht der CSAR-Datei.
Ausnahmen:	-
Nachbedingungen:	Der CSAR-Nutzer bekommt eine Detailansicht einer gewünschten CSAR-Datei.

Tabelle 5.3: Anwendungsfall: CSAR-Datei im Detail anschauen

5.2.4 Anwendungsfall: CSAR-Datei herunterladen

Dieser Anwendungsfall beschreibt den Vorgang des Herunterladens einer CSAR-Datei aus Sicht des CSAR-Nutzers (vgl. Tabelle 5.4)

Anwendungsfall:	CSAR-Datei herunterladen
Akteure:	CSAR-Nutzer
Vorbedingungen:	• Der CSAR-Nutzer ist am Marktplatz angemeldet. • Es sind CSAR-Dateien im Marktplatz verfügbar.
Ablauf:	1. Der CSAR-Nutzer durchsucht die verfügbaren CSAR-Dateien (siehe Anwendungsfall, Tabelle 5.2). 2. Der CSAR-Nutzer wählt eine CSAR-Datei aus und betätigt den „Download“-Button. 3. Die gewünschte CSAR-Datei wird heruntergeladen, beziehungsweise in den Container des CSAR-Nutzers importiert.
Ausnahmen:	3a) Der Download schlägt fehl. 3a1) Der CSAR-Nutzer erhält eine Fehlermeldung.
Nachbedingungen:	Der CSAR-Nutzer hat die gewünschte CSAR-Datei heruntergeladen.

Tabelle 5.4: Anwendungsfall: CSAR-Datei herunterladen

5.2.5 Anwendungsfall: Eine CSAR-Datei zum Marktplatz hinzufügen

Dieser Anwendungsfall (Tabelle 5.5) behandelt das Hinzufügen einer neuen CSAR-Datei zum Marktplatz.

Anwendungsfall:	Eine CSAR-Datei zum Marktplatz hinzufügen.
Akteure:	CSAR-Anbieter
Vorbedingungen:	• Der CSAR-Anbieter ist am Marktplatz angemeldet.
Ablauf:	1. Der CSAR-Anbieter übergibt der Benutzeroberfläche den Produktnamen. 2. Der CSAR-Anbieter schickt diese Informationen an den Marktplatz. 3. Im Marktplatz ist nun ein neues Produkt mit den übergebenen Daten vorhanden.
Ausnahmen:	2a) Es ist bereits ein Produkt mit dem selben Namen verfügbar. 2a1) Der CSAR-Anbieter erhält eine Fehlermeldung.
Nachbedingungen:	Eine neue CSAR-Datei ist im Marktplatz verfügbar.

Tabelle 5.5: Anwendungsfall: Eine CSAR-Datei zum Marktplatz hinzufügen.

5.2.6 Anwendungsfall: Eine CSAR-Datei ändern

Dieser Anwendungsfall (Tabelle 5.6) beschreibt den Vorgang des Änderns der Metadaten einer CSAR-Datei.

Anwendungsfall:	Eine CSAR-Datei ändern
Akteure:	CSAR-Anbieter
Vorbedingungen:	• Der CSAR-Anbieter ist am Marktplatz angemeldet. • Es ist eine CSAR im Marktplatz vorhanden, welche vom CSAR-Anbieter erstellt wurde.
Ablauf:	1. Der CSAR-Anbieter wählt die zu ändernde CSAR-Datei aus. 2. Der CSAR-Anbieter übermittelt die zu ändernden Metadaten.
Ausnahmen:	2a) Die Datei wurde nicht von diesem CSAR-Anbieter erstellt. 2a1) Der CSAR-Anbieter erhält eine Fehlermeldung. 2b) Die Metadaten können nicht geändert werden. 2b1) Der CSAR-Anbieter erhält eine Fehlermeldung.
Nachbedingungen:	Die Metadaten der CSAR wurden geändert.

Tabelle 5.6: Anwendungsfall: Eine CSAR-Datei ändern

5.2.7 Anwendungsfall: Eine CSAR-Datei löschen

Dieser Anwendungsfall (Tabelle 5.7) beschreibt den Vorgang des Löschens einer CSAR-Datei.

Anwendungsfall:	Eine CSAR-Datei löschen
Akteure:	CSAR-Anbieter
Vorbedingungen:	• Der CSAR-Anbieter ist am Marktplatz angemeldet. • Es ist eine CSAR im Marktplatz vorhanden, welche vom CSAR-Anbieter erstellt wurde.
Ablauf:	1. Der CSAR-Anbieter wählt eine CSAR-Datei aus. 2. Der CSAR-Anbieter löscht diese.
Ausnahmen:	2a) Die Datei wurde nicht von diesem CSAR-Anbieter erstellt. 2a1) Der CSAR-Anbieter erhält eine Fehlermeldung. 2b) Die Datan können nicht gelöscht werden. 2b1) Der CSAR-Anbieter erhält eine Fehlermeldung.
Nachbedingungen:	Die CSAR-Datei ist gelöscht.

Tabelle 5.7: Anwendungsfall: Eine CSAR-Datei löschen

5.3 Architektur

In diesem Abschnitt wird die Architektur der Benutzeroberflächen (Abschnitt 5.4) und die des Marktplatzes (Abschnitt 5.5) beschrieben.

Abbildung 5.5 beschreibt vorab das Zusammenspiel der Benutzeroberflächen und des Marktplatzes als Client-Server-Modell. Weiterhin sind die Datenbanken des Marktplatzes angedeutet.

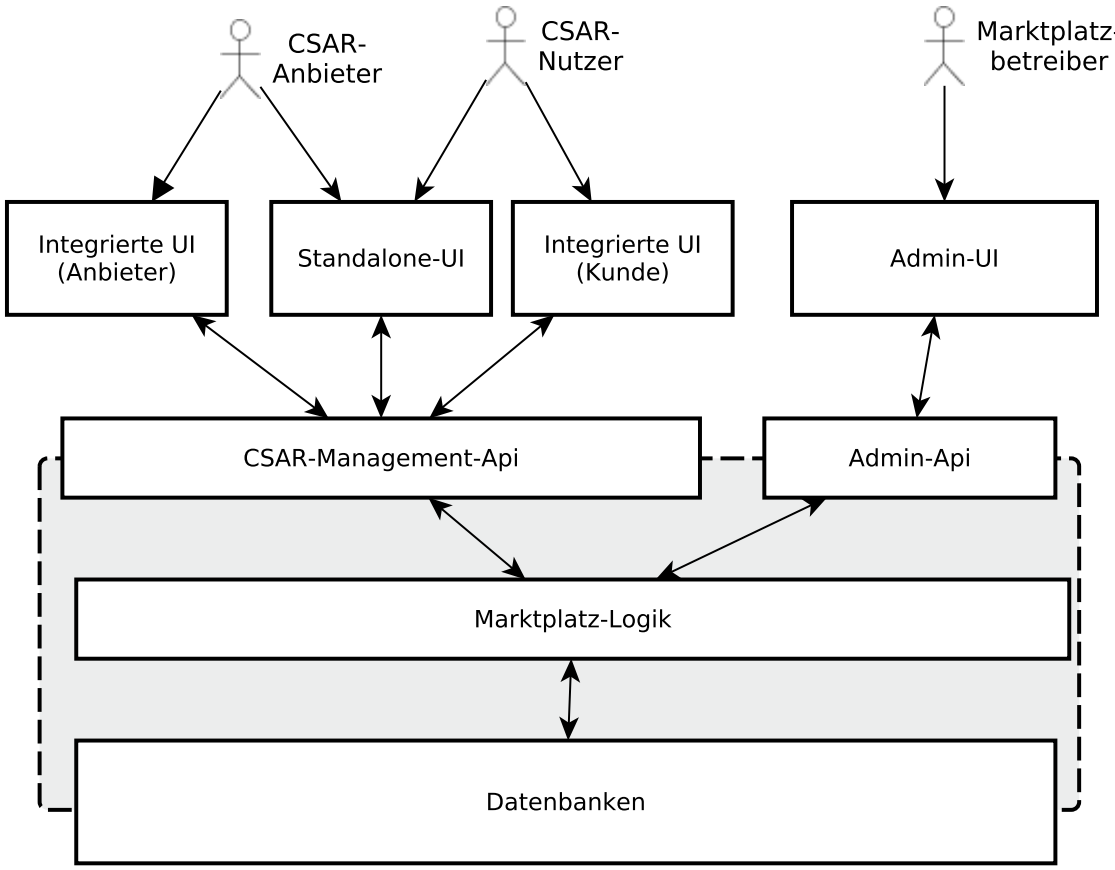

Abbildung 5.5: Marktplatz und Benutzeroberfläche

5.4 Architektur der Benutzeroberfläche

Die Benutzeroberfläche wird in zwei voneinander getrennten Komponenten aufgeteilt. Zum einen in eine Komponente für CSAR-Anbieter und -Nutzer, und zum anderen in eine administrative Schnittstelle zur Verwaltung des Marktplatzes.

Die Komponenten für CSAR-Anbieter und -Nutzer können auf mehreren Arten implementiert werden. Demzufolge sind mögliche Implementierungen für diese Schnittstellen, als auch für die administrative Schnittstelle gegeben. Es ist zu beachten, dass die möglichen Lösungsansätze auch untereinander kombiniert werden können.

Die Benutzeroberfläche kann in drei Varianten realisiert werden:

- Standalone UI (Abschnitt 5.4.1)
- Integrierte UI (Abschnitt 5.4.2)
- Admin UI (Abschnitt 5.4.3)

Die Admin UI wird hier seperat beschrieben, da diese getrennt von den Benutzeroberflächen des Marktplatzes implementiert werden soll.

5.4.1 Standalone UI

Der Marktplatz kann über eine Benutzeroberfläche, ähnlich der eines Web-Shops benutzt werden. Anbieter können dort manuell CSAR-Pakete zur Verfügung stellen. Nutzer des Marktplatzes können CSAR-Dateien herunterladen und diese manuell in ihren TOSCA-Container importieren.

Die Benutzeroberfläche interagiert in diesem Fall ausschließlich mit dem CSAR-Nutzer, beziehungsweise CSAR-Anbieter.

5.4.2 Integrierte UI

Die Benutzeroberfläche des Marktplatzes wird direkt in ein bestehendes System integriert. Je nach Anwendungsfall ergeben sich folgende zwei Möglichkeiten:

Integriert in TOSCA-Laufzeitumgebung:

Der Marktplatz wird direkt in eine TOSCA-Laufzeitumgebung integriert. Über diese Oberfläche kann der Kunde direkt CSAR-Dateien herunterladen und in seinen TOSCA-Container importieren.

Integriert in Modellierungswerkzeug:

Der Marktplatz wird in ein Modellierungswerkzeug integriert. Dies ermöglicht es einem Anbieter seine CSAR-Dateien direkt aus diesem Modellierungswerkzeug auf dem Marktplatz zu publizieren.

In beiden Fällen interagiert die Benutzeroberfläche sowohl mit dem CSAR-Nutzer und dessen TOSCA-Laufzeitumgebung, als auch mit dem CSAR-Anbieter und dessen Modellierungswerkzeugs.

5.4.3 Admin UI

Die administrative Benutzeroberfläche des Marktplatzes wird getrennt von den Benutzeroberflächen des Anbieters/Nutzers betrieben, um eine mögliche Korrelation zu vermeiden.

Die administrative Benutzeroberfläche interagiert in diesem Fall ausschließlich mit dem Marktplatzbetreiber via der Admin-API des Marktplatzes.

5.5 Architektur des Marktplatzes

In diesem Abschnitt wird auf die Architektur des Marktplatzes eingegangen.

5.5.1 Marktplatz – Überblick

Die Architektur des Marktplatzes kann graphisch wie in Abbildung 5.6 dargestellt werden. Diese Architektur kann semantisch in drei Abschnitte aufgeteilt werden:

- Die (von außen) erreichbaren Schnittstellen zur Interaktion mit den Benutzeroberflächen (Abschnitt 5.5.2).
- Die eigentliche Marktplatz-Logik (Abschnitt 5.5.3).
- Die Datenbanken und deren Verwaltung (Abschnitt 5.5.4).

Als (von außen) erreichbare Schnittstellen werden die folgenden beiden APIs angeboten:

- **CSAR-Management-API**: Die CSAR-Management-API ermöglicht den Zugriff auf gespeicherte CSAR-Dateien, als auch deren Metadaten, wie z. B. eine Beschreibung dieser Dateien. Über die CSAR-Management-API können (sofern autorisiert) neue CSAR-Dateien hoch- beziehungsweise heruntergeladen werden, als auch bestehende Dateien editiert werden (Abschnitt 6.2.7).
- **Admin-API**: Die Admin-API ermöglicht die Verwaltung von Nutzern, Datenbanken und Einstellungen des Marktplatzes (Abschnitt 6.2.7).

Beide Schnittstellen werden als RESTful-Service angeboten.

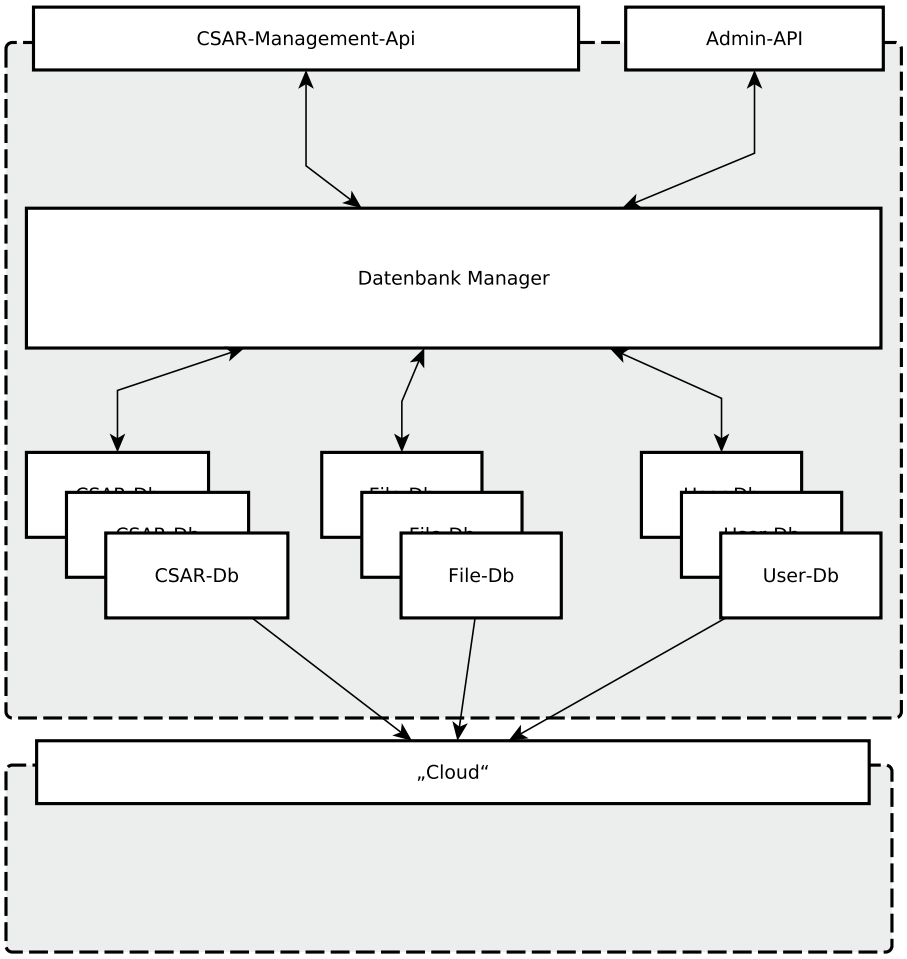

Abbildung 5.6: Architektur des Marktplatzes

5.5.2 Beschreibung der externen Schnittstellen

In diesem Unterabschnitt wird auf die externen Schnittstellen des Marktplatzes eingegangen. Beide Schnittstellen werden als RESTful-Service angeboten. Daher wird nun auf die entsprechenden Ressourcen eingegangen.

Abbildung 5.7 stellt die Ressourcen der eigentlichen API dar. Die weiß unterlegten Kästen repräsentieren statische Ressourcennamen während die gelb unterlegten Kästen variable Ressourcennamen darstellen. Diese Ressourcen werden in Abschnitt 6.2.7 genauer erläutert.

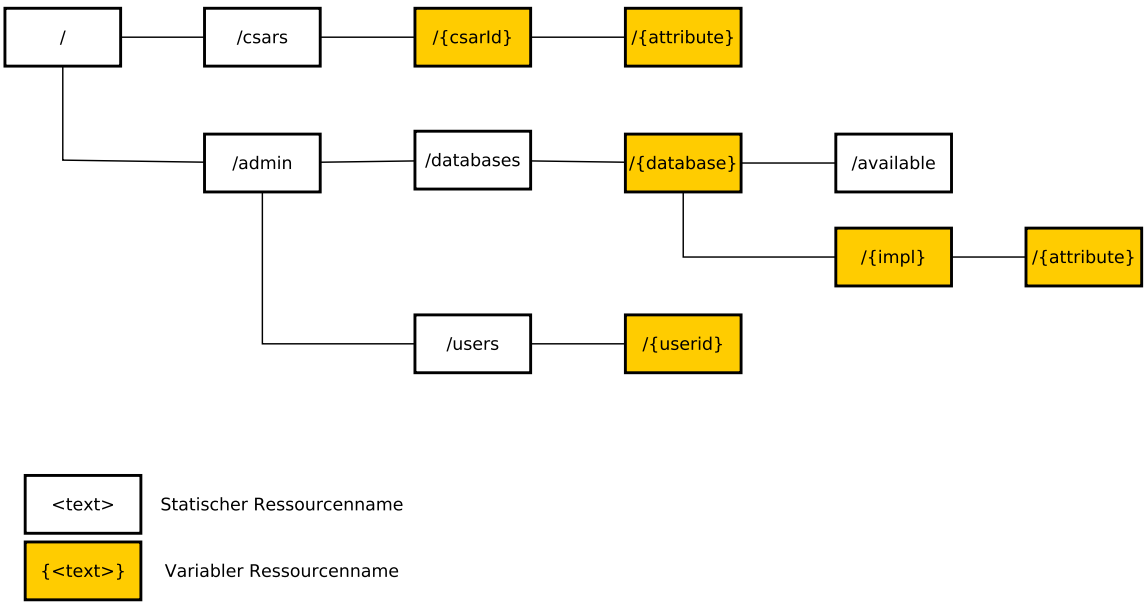

Abbildung 5.7: RESTful-Schnittstelle

5.5.3 Marktplatz-Logik

Die Marktplatz-Logik ist für das semantische Verknüpfen der Inhalte der verschiedenen Datenbanken, sowie für den Datenfluss zuständig.

Im Marktplatz stehen bisher drei Datenbanken zur Verfügung. Diese werden im Folgenden kurz erläutert:

CSAR-DB:

Die CSAR-Datenbank ist für das Vorhalten der Meta-Informationen einer CSAR-Datei zuständig. Diese können mittels einer CSAR-ID abgerufen und/oder geändert werden.

File-DB:

Die File-Datenbank ist für das Speichern und Zugreifen auf Dateien im Marktplatz zuständig. So kann beispielsweise mittels der File-Datenbank die eigentliche CSAR-Datei, aber auch deren Icon oder Vorschaubild heruntergeladen werden.

User-DB:

Die User-Datenbank speichert die Daten der Marktplatznutzer.

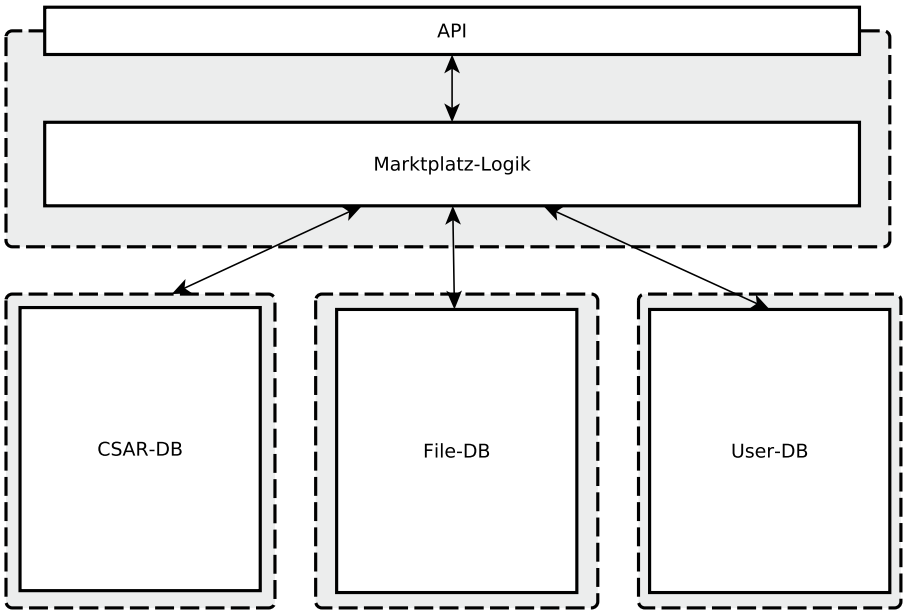

Abbildung 5.8: Marktplatz-Logik

5.5.4 Marktplatz Datenbanken

Der Marktplatz verfügt über Datenbanken zur Verwaltung der CSAR-Dateien, zur Speicherung der zugehörigen Metainformationen, als auch Datenbanken zur Administration des Marktplatzes. Alle Datenbanken werden austauschbar implementiert, um einen schnellen Wechsel verschiedener Implementierungen zu ermöglichen.

Um zwischen diesen Implementierungen zu wechseln, wird eine Datenbank-Management-Komponente benötigt, welche die gewünschten Implementierungen bereitstellen kann (siehe Abbildung 5.9).

Die verschiedenen Implementierungen der Datenbanken können die Daten dann zum Beispiel bei Cloud-Anbietern, aber auch in lokalen Datenbanken, speichern.

Abbildung 5.9 zeigt dies exemplarisch anhand der CSAR-DB. Beim Zugriff auf die CSAR-DB wird mittels des CSAR-DB-Managers anhand von hinterlegten Einstellungen eine Implementierung ausgewählt. Dies geschieht transparent für den Benutzer der Datenbank. Dasselbe geschieht analog dazu bei der File-DB und bei der User-DB.

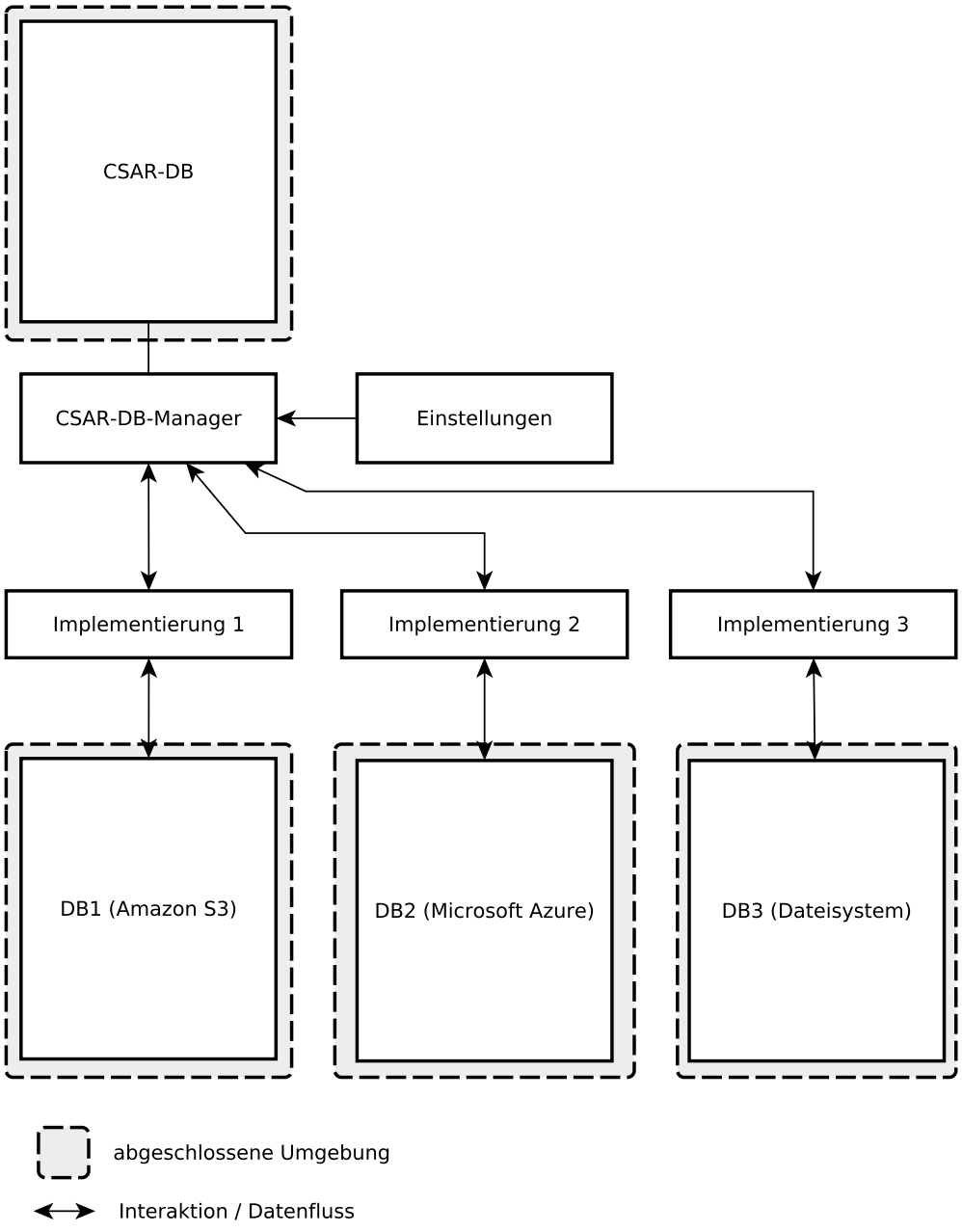

Abbildung 5.9: Austauschbarkeit der Datenbanken

6 Implementierung

Dieses Kapitel behandelt die konkrete Implementierung der im vorigen Kapitel beschriebenen Konzepte. Zuerst wird die Implementierung der Marktplatz-Logik („Back-End“) und danach auf die Implementierung der Benutzeroberflächen („Front-End“) eingegangen.

6.1 Verwendete Technologien

Dieser Abschnitt beschreibt die verwendeten Technologien, welche sowohl bei der Marktplatz-Logik als auch bei den Benutzeroberflächen Verwendung fanden. Die verwendeten Technologien werden kurz beschrieben und die zugrunde liegende Motivation erläutert. Schlussendlich wird ein Fazit über die Verwendung dieser Technologien gegeben.

6.1.1 Buildwerkzeug Gradle

Gradle[1] ist ein Buildwerkzeug, welches auf Java basiert. Die zugrunde liegenden Gradle-Skripte werden in einer auf Groovy basierenden Sprache verfasst. Dadurch sind Gradle-Skripte ausführbare Programme, die sehr flexibel gestaltet werden können. Gradle zeichnet sich durch viele Plug-Ins aus, welche es zum Beispiel ermöglichen direkt Eclipse-Projekte zu erstellen.

Motivation:

Da ein Buildwerkzeug viel Arbeit beim Verwalten von Bibliotheken, als auch beim Compilieren der Anwendung abnimmt, ist es selbstverständlich, dass ein Build-Tool gewählt wurde. Da die Winery zum Zeitpunkt der Entscheidung Gradle benutzte, fiel die Wahl auf Gradle, um bereits vorhandenes Wissen weiterzuverwenden.

[1] http://www.gradle.org

Alternativen:

Als Alternativen zu Gradle sind noch Maven, Apache Buildr und Apache Ant zur Verfügung gestanden. Die Aufgezählten stellen jedoch nur die populärsten [Arh13] der möglichen Alternativen dar.

Gradle erwies sich als sehr nützlich und in der Verwendung als intuitiv. Da Teile von OpenTOSCA jedoch bereits Maven verwenden, wäre die Benutzung von Maven sinnvoller gewesen, um die Benutzung zweier unterschiedlicher Build-Tools zu vermeiden. Winery wird derzeit von Gradle auf Maven umgestellt, um die Eclipse Common Build Infrastructure[2] nutzen zu können. Als Problem erwies sich dies vor allem, als der Marktplatz in die bestehende OpenTOSCA-Benutzeroberfläche integriert wurde.

6.1.2 Jersey

Zur Realisierung der Web-Schnittstellen wurde ein RESTful-Web-Service mittels Jersey[3] implementiert. Auch zur Erstellung eines Clients für diese Schnittstelle wurden die Jersey-Client Bibliotheken verwendet.

Motivation:

In einer Fachstudie von Markus Fischer, Kalman Kepes und Alexander Wassiljew wurden neun Frameworks zur Implementierung von REST basierten Anwendungen in Java evaluiert. Jersey schnitt in dieser Arbeit als eines der besten Frameworks ab. Da sowohl der OpenTOSCA-Container als auch die Winery Jersey verwenden, bot es sich an ebenfalls an Jersey einzusetzen, um bereits vorhandenes Wissen wiederzuverwenden. [FKW13]

Alternativen:

Als weitere Implementierungen von JAX-RS können JBoss' RESTEasy als auch das Spring-Framework genannt werden. Wobei das Spring-Framework weit mehr als nur JAX-RS implementiert und ein komplettes Framework für die Entwicklung mittels Java/Java-EE zur Verfügung stellt.

[2] http://wiki.eclipse.org/CBI
[3] https://jersey.java.net

6.1.3 OAuth 2.0

Um sich am Marktplatz zu authentifizieren, wurde OAuth[4] verwendet. Das OAuth-Protokoll stellt seit einiger Zeit den De-facto-Standard [TYK+12] zur Client-Autorisierung dar. Viele große Anbieter von Web-Diensten bieten eine Autorisierung mittels OAuth an. Als große Anbieter sind Facebook[5], GitHub[6], Google[7] und Twitter[8] zu nennen. Somit ist es für den Benutzer möglich, sich mittels dieser Accounts am Marktplatz anzumelden.

Beschreibung:

OAuth ist ein offenes standardisiertes Protokoll zur Autorisierung von Applikationen. OAuth erlaubt somit verschiedenen Applikationen Datenzugriff untereinander zu gewähren, ohne dass der Benutzer seinen Benutzernamen und sein Passwort an Dritte weitergeben muss.

Der Ablauf einer OAuth-Authentifizierung kann wie in Abbildung 6.1 graphisch beschrieben werden:

1. Der Nutzer erhält vom Marktplatz eine OAuth-Url, welche einen anwendungsspezifischen Schlüssel enthält und auf die OAuth-API des gewünschten OAuth-Providers zeigt.
2. Der Nutzer wird auf die OAuth-Seite des OAuth-Providers weitergeleitet. Dieser erwartet dann eine Bestätigung des Nutzers, dass der Marktplatz auf die Daten des Nutzers zugreifen darf.
3. Sofern der Nutzer dies bestätigt, wird er wieder auf den Marktplatz weitergeleitet.
4. Der Browser des Nutzers leitet diesen zurück auf den Marktplatz. Somit wird ein Grant-Code an den Marktplatz übergeben.
5. Der Marktplatz sendet diesen Grant-Code an den OAuth-Provider.
6. Der OAuth-Provider antwortet dem Marktplatz mit einem Access-Token. Dieser Token kann nun verwendet werden, um auf die freigegebenen Daten des Nutzers zuzugreifen.

Im Falle des Marktplatzes, agiert dieser als OAuth-Client und GitHub bspw. als OAuth-Provider. Damit der Marktplatz als OAuth-Client agieren kann, muss dieser bei dem entsprechenden OAuth-Provider als „Applikation“ (die Bezeichnung variert von Provider zu Provider) registriert werden. Dadurch erhält man eine ID und ein Passwort.

[4] http://www.oauth.net
[5] http://www.facebook.com
[6] http://www.github.com
[7] http://www.google.com
[8] http://www.twitter.com

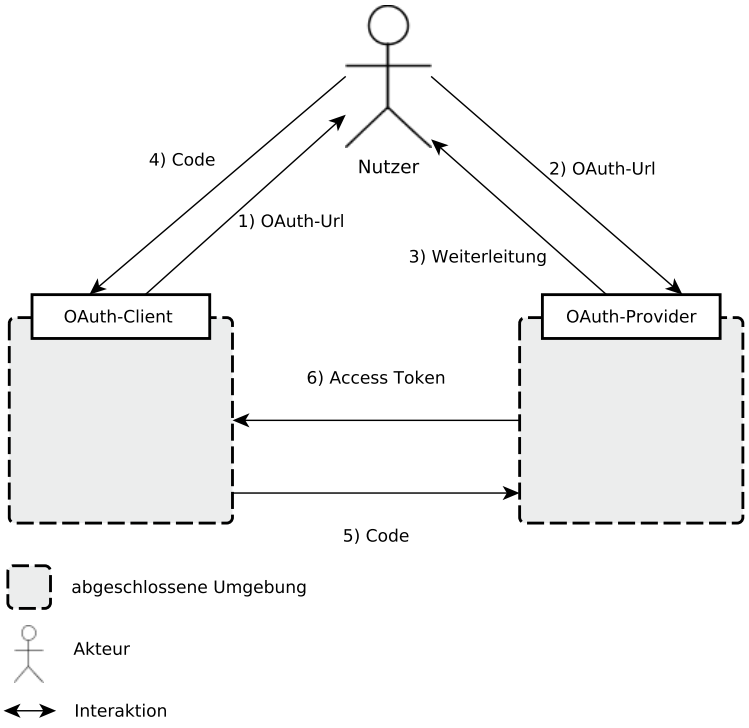

Abbildung 6.1: Funktionsweise von OAuth - Frei nach David Manaster[a]

[a] http://blog.manaster.com/blog/2013/06/16/oauth/

Alternativen:

Als Alternative zur Authentifizierung kann auch OpenID verwendet werden. OpenID kann jedoch ausschließlich zur Authentifizierung verwendet werden, während OAuth auch weiterhin einen Zugriff auf weitere Daten ermöglicht. Dies könnte vor allem in Zukunft für den Marktplatz interessant werden.

Fazit:

Die Verwendung von OAuth vereinfacht den Registrierungsprozess am Marktplatz sehr, da kein Nutzer ein neues Benutzerkonto manuell erstellen muss. Aus Sicht des Programmierens erwies sich OAuth als unproblematisch, wobei zu erwähnen ist, dass OAuth nur eine Schnittstelle zur Autorisierung enthält und man somit weiterhin mittels der anbieterspezifischen Schnittstellen arbeiten muss. Dies wäre bei OpenID nicht der Fall gewesen, würde aber jegliche Erweiterung in Richtung Daten-Zugriff ausschließen.

6.2 Marktplatz-Logik

Dieser Abschnitt geht auf die Implementierung der Marktplatz-Logik ein. Zuerst werden verwendete Technologien und deren Funktionsweise vorgestellt. Danach wird auf die wichtigsten Datenobjekte und die Datenbanken eingegangen.

6.2.1 Verwendete Technologien

Dieser Abschnitt behandelt die Verwendeten Technologien, die nur in der Marktplatz-Logik beziehungsweise dem Back-End Verwendung finden.

Apache JClouds

Als Framework zur Dateiablage wird Apache JClouds[9] verwendet.

Beschreibung: Apache JClouds ist eine Open-Source-Bibliothek zur Verwendung von Cloud-Services. JClouds bietet eine Abstraktionsschicht an, welche es dem Entwickler ermöglicht, unabhängig vom Anbieter des Cloud-Services diesen, ohne Kenntnis der zugrunde liegenden Schnittstelle, zu benutzen.

In Abbildung 6.2 ist zu sehen, wie JClouds in den Marktplatz integriert wurde; Der Datenbankmanager ist für die Auswahl der gewünschten Datenbank zuständig. Somit besteht die Möglichkeit Daten auf dem lokalen Datei-System, als auch in der Cloud abzulegen. Mittels JClouds können Implementierungen für weitere Cloud-Provider mit marginalen Änderungen auf weitere Cloud-Anbieter übertragen werden.

Motivation: Da sich die komplette TOSCA-Materie primär mit „der Cloud“ befasst, lag es nahe, dass auch die Daten des Marktplatzes in der Cloud gespeichert werden sollen. Um möglichst providerunabhängig zu bleiben, bot sich die Benutzung einer Bibliothek an, welche mehrere Provider miteinander vereint.

Alternativen: Als Alternativen wären providerspezifische Schnittstellen in Frage gekommen. Dies hätte jedoch zur Folge gehabt, dass der Quell-Code nicht sehr portabel und einheitlich gewesen wäre.

Fazit: Mittels JClouds ist es möglich die Komponenten zur Datenhaltung sehr variabel zu gestalten.

[9] http://jclouds.incubator.apache.org/

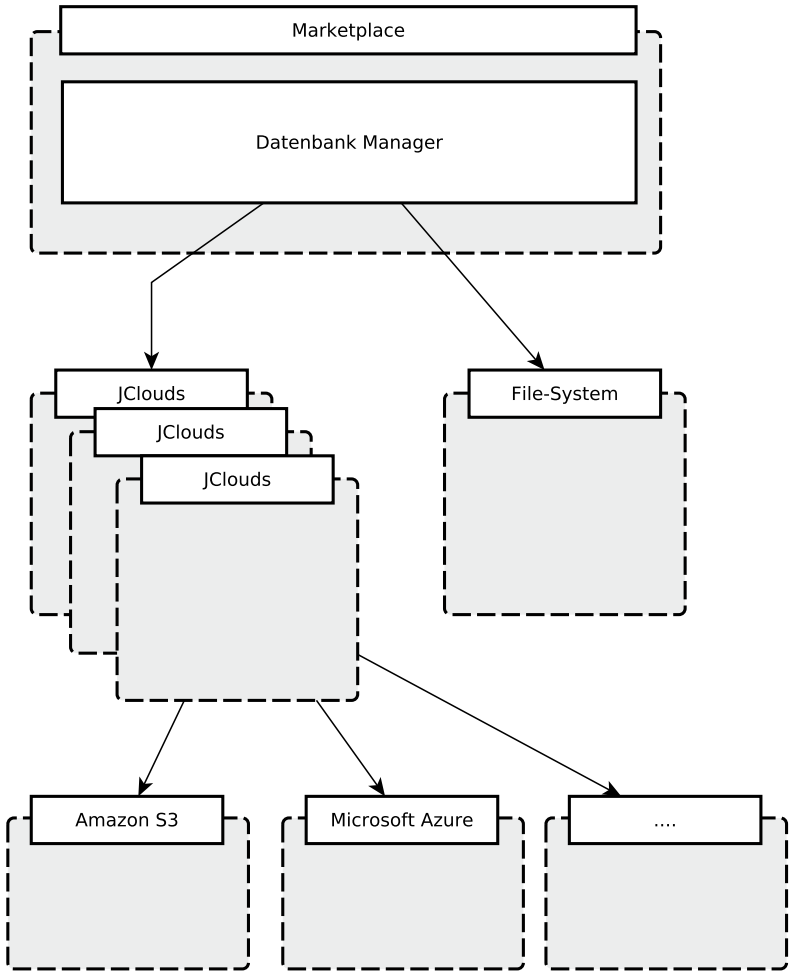

Abbildung 6.2: Funktionsweise von JClouds

Java ServiceLoader

Als elementarer Plug-In-Mechanismus für die verwendeten Datenbanken wurde der Java ServiceLoader benutzt.

Beschreibung: Der Java ServiceLoader ist Teil der Java SE und bietet eine Möglichkeit des dynamischen Ladens von Implementierungen einer (vorher) definierten Schnittstelle.

Motivation: Es wird der Java ServiceLoader eingesetzt, da dieser keine weiteren Abhängigkeiten nach sich zieht und seit Java SE 1.6 in dieser enthalten ist.

Alternativen: Als weitere Frameworks zur Modularisierung ist allen voran OSGi zu nennen. OSGi bietet weitaus mehr Möglichkeiten und Funktionen als der Java ServiceLoader. Im Falle des Marktplatzes war dieser Mehrwert jedoch nicht nötig.

Fazit: Der Java ServiceLoader erwies sich für die Implementierung des Marktplatzes als ausreichend und leichtgewichtig.

6.2.2 Datenobjekt: CSAR

Dieser Abschnitt beschreibt das CSAR-Datenobjekt, welches intern in der Marktplatz-Logik verwendet wird. Dieses CSAR-Datenobjekt enthält Meta-Informationen des Marktplatzes. Die Attribute dieser CSAR-Datei seien wie folgt beschrieben:

- **Name:** Der Name der CSAR-Datei.
- **Overview:** Eine Kurzbeschreibung der angebotenen CSAR-Datei.
- **Description:** Eine ausführliche Beschreibung der angebotenen CSAR-Datei.
- **ImageFileID:** Diese ID wird benötigt, um von der File-Datenbank das gespeicherte Vorschaubild der CSAR zu erhalten. Diese ID wird von der File-Datenbank generiert und wird von der Marktplatz-Logik hinterlegt.
- **CsarFileID:** Analog zu ImageFileDB – für die (eigentliche) CSAR-Datei.
- **IconFileID:** Analog zu ImageFileDB – für das Icon.
- **Tags:** Die Tags dienen zum Gruppieren von CSAR-Dateien. Tags müssen beim Publizieren im Marktplatz angegeben werden.
- **ID:** Die ID der CSAR wird von der CSAR-Datenbank vergeben und dient zur eindeutigen Identifikation dieser CSAR im Marktplatz.
- **Filename:** Der originale Dateiname der CSAR-Datei.

Listing 6.1 CSAR-Datenbank (Java)

```
public interface CsarDb {
       public ArrayList<Csar> getCsars();
       public String addCsar(Csar csar);
       public boolean updateCsar(String Id, Csar csar);
       public boolean deleteCsar(String id);
}
```

6.2.3 Datenobjekt: USER

Dieser Abschnitt beschreibt das User-Datenobjekt, welches intern in der Marktplatz-Logik verwendet wird.

Die Attribute eines USER-Objektes seien nachfolgend beschrieben:

- **UserType:** Gibt den Benutzerlevel des Nutzers an: USER, VENDOR, oder ADMIN. Dieser Status ist standardmäßig „USER“, sofern vom Admnistrator nicht geändert.
- **Username:** Dies ist der Benutzername des Nutzers. Der Name wird mittels OAuth erhalten.
- **ProviderType:** Der ProviderType gibt an, mittels welches OAuth-Providers der Nutzer sich authentifizierte.
- **ProviderId:** Dies ist die Identifikation des Nutzers auf dem von ihm benutzen OAuth-Providers.
- **Session:** Dieses Attribut beinhaltet die aktuelle Session-ID des Nutzers.

6.2.4 CSAR-Datenbank

Dieser Abschnitt beschreibt die Spezifizierung und Implementierung der CSAR-Datenbank, bzw. deren Schnittstelle. In Listing 6.1 sind die wichtigsten Methoden der CSAR-Datenbank-Schnittstelle aufgeführt.

Die dahinterstehende Funktionalität ergibt sich aus den Methoden-Namen. Beim Hinzufügen einer CSAR-Datei wird als Rückgabewert die ID der soeben hinzugefügten CSAR-Datei zurückgegeben. Anhand dieser ID kann von nun an diese CSAR identifiziert werden. Abgesehen von diesen (elementaren) Methoden, bietet die Schnittstelle noch weitere Funktionen zum Suchen von CSAR-Dateien, oder zum Erhalten von Meta-Informationen. Auf diese wird hier aber nicht näher eingegangen.

Derzeit ist eine CSAR-DB auf Basis von MySQL, mittels JDBC implementiert.

Listing 6.2 File-Datenbank (Java)

```
public interface FileDb {
	public File getFile(String id);
	public String saveFile(File file);
	public String updateFile(String id, File file);
	public boolean deleteFile(String id);
}
```

Listing 6.3 User-Datenbank (Java)

```
public interface UserDb {
	public User getUser(String id);
	public boolean addUser(User user);
	public boolean updateUser(String id, User user)
	public boolean deleteUser(String id);
}
```

6.2.5 File-Datenbank

Dieser Abschnitt erläutert die Implementierung bzw. Spezifizierung der File-Datenbank, sowie deren Schnittstelle. Analog zur CSAR-Db sind folgend die wichtigsten Methoden der Schnittstelle in Listing 6.2 gegeben.

Die Funktionalität wird mittels Methoden im CRUD-Prinzip (Create, Read, Update, Delete) zur Verfügung gestellt. Analog zur CSAR-Db wird beim Erstellen, aber auch beim Ändern einer Datei, eine eindeutige ID zurückgegeben.

Derzeit wurden Implementierungen für Amazon S3 und Microsoft Azure mittels jClouds realisiert.

6.2.6 User-Datenbank

Dieser Abschnitt beschreibt die Spezifikation und Implementierung der User-Datenbank. Die Schnittstelle der User-Datenbank ist in Listing 6.3 gegeben. Analog zu der File-Datenbank und der CSAR-DB ergeben sich die Operationen aus den Namen der Methoden.

6.2.7 Die REST-Schnittstelle im Detail

Folgender Abschnitt beschreibt die Implementierung der RESTful-Schnittstelle (siehe Abschnitt 6.2.7) im Detail. Diese Schnittstelle ist in die CSAR-Management-API (Abschnitt 6.2.7) und die Admin-API (Abschnitt 6.2.7) unterteilt.

Als generelle Ressource, welche der CSAR-Management-API und der Admin-API übergeordnet ist, wird eine Schnittstelle zur Authentifizierung angeboten. Diese Authentifizierung ist für beide, voneinander getrennten, Benutzeroberflächen gültig.

Ressource:	/
GET:	
PUT:	
POST:	Ein/Ausloggen, Status abfragen
DELETE:	
Pfad-Parameter:	
Query-Parameter:	
Notiz:	Authentifizierung gilt sowohl für CSAR-Management-API als auch für Admin-API

Tabelle 6.1: REST-Ressource: /

CSAR-Management-API:

Mittels der CSAR-Management-API können Operationen ausgeführt werden, welche im Zusammenhang mit CSAR-Dateien stehen. Die zur Verfügung gestellten Operationen umfassen alle notwendigen CRUD-Operationen[10] für CSAR-Dateien im Marktplatz.

- **C**reate: Siehe Tabelle 6.2
- **R**ead: Siehe Tabelle 6.2, Tabelle 6.3, Tabelle 6.4
- **U**pdate: Siehe Tabelle 6.3, Tabelle 6.4
- **D**elete: Siehe Tabelle 6.3

[10]Create, Read, Update, Delete – Operationen

Ressource:	/csar
GET:	Liste aller CSARs erhalten
PUT:	
POST:	Hinzufügen einer neuen CSAR-Datei
DELETE:	
Pfad-Parameter:	
Query-Parameter:	tags – Filtert die Ausgabe nach den angegebenen Tags

Tabelle 6.2: REST-Resource: /csar

Ressource:	/csar/{csarid}
GET:	Erhalte eine Liste mit Links zu allen Attributen einer CSAR mit der Id {csarid}
PUT:	
POST:	
DELETE:	Löschen der CSAR
Pfad-Parameter:	{csarid} – Id der anzuzeigenden CSAR
Query-Parameter:	

Tabelle 6.3: REST-Resource: /csar/{csarid}

Ressource:	/csar/{csarid}/{attribut}
GET:	Zeigt den Wert eines Attributes der CSAR mit der ID {csarid} an
PUT:	Attribut ändern
POST:	
DELETE:	Attributinhalt löschen
Pfad-Parameter:	{csarid} – Id der anzuzeigenden CSAR {attribut} – Anzuzeigendes Attribut
Query-Parameter:	

Tabelle 6.4: REST-Resource: /csar/{csarid}/{attribut}

Admin-API:

Mittels der Admin-API können administrative Aufgaben im Marktplatz übernommen werden. Diese Schnittstelle wird nur vom Betreiber des Marktplatzes verwendet und ist daher logisch von der CSAR-Management-API getrennt.

Die Admin-API wird nochmals in zwei untergeordnete Ressourcen unterteilt:

- Die Database-Ressource (Tabelle 6.6, Tabelle 6.8, Tabelle 6.9, Tabelle 6.10):
  Diese Ressource stellt Operationen zur Verwaltung der verschiedenen Datenbanken bereit.
- Die Users-Ressource (Tabelle 6.11, Tabelle 6.12, Tabelle 6.13):
  Diese Ressource stellt Operationen zur Verwaltung der Benutzer des Marktplatzes bereit.

Ressource:	/admin/
GET:	Erhalte eine Liste von Links zu verfügbaren Ressourcen
PUT:	
POST:	
DELETE:	
Pfad-Parameter:	
Query-Parameter:	

Tabelle 6.5: REST-Resource: /admin/

Ressource:	/admin/databases/
GET:	Erhalte eine Liste von Links zu den verschiedenen Datenbanken
PUT:	
POST:	
DELETE:	
Pfad-Parameter:	
Query-Parameter:	

Tabelle 6.6: REST-Resource: /admin/databases/

Ressource:	/admin/databases/{database}/
GET:	Zeigt die aktuell verwendete Implementierung für den Datenbanktyp {database} an
PUT:	
POST:	Operation auf aktueller Implementierung ausführen
DELETE:	
Pfad-Parameter:	{database} – Der gewünschte Datenbanktyp: „filedb“, „userdb“ oder „csardb“
Query-Parameter:	

Tabelle 6.7: REST-Resource: /admin/databases/{database}/

Ressource:	/admin/databases/{database}/available
GET:	Zeigt die verfügbaren Implementierungen für den Datenbanktyp {database} an
PUT:	
POST:	
DELETE:	
Pfad-Parameter:	{database} – Der gewünschte Datenbanktyp: „filedb“, „userdb“ oder „csardb“
Query-Parameter:	

Tabelle 6.8: REST-Resource: /admin/databases/{database}/available

Ressource:	/admin/databases/{database}/{implementation}/
GET:	Erhalte eine Liste von Links zu den (spezifischen) Attributen der Implementierung {implementation} für den Datenbanktyp {database}
PUT:	
POST:	
DELETE:	
Pfad-Parameter:	{database} – Der gewünschte Datenbanktyp: „filedb“, „userdb“ oder „csardb“ {implementation} – Klassenname der gewünschten Implementierung.
Query-Parameter:	

Tabelle 6.9: REST-Resource: /admin/databases/{database}/{implementation}/

Ressource:	/admin/databases/{database}/{implementation}/{attribut}/
GET:	Attribut anzeigen
PUT: Attribut ändern	
POST:	
DELETE:	
Pfad-Parameter:	{database} – Der gewünschte Datenbanktyp: „filedb“, „userdb“ oder „csardb“ {implementation} – Klassenname der gewünschten Implementierung. {attribut} – Anzuzeigendes, oder änderndes Attribut
Query-Parameter:	

Tabelle 6.10: REST-Resource: /admin/databases/{database}/{implementation}/{attribut}/

Ressource:	/admin/users/
GET:	Erhalte eine Liste mit Links zu allen Benutzern
PUT:	
POST:	Nutzer hinzufügen
DELETE:	
Pfad-Parameter:	
Query-Parameter:	

Tabelle 6.11: REST-Resource: /admin/users/

Ressource:	/admin/users/{userid}/
GET:	Erhalte eine Link-Liste zu allen Attributen des Nutzers mit der Id {userid}
PUT:	
POST:	
DELETE:	Lösche den Nutzer mit der Id {userid}
Pfad-Parameter:	{userid} – Id des gewünschten Nutzers
Query-Parameter:	

Tabelle 6.12: REST-Resource: /admin/users/{userid}/

Ressource:	/admin/users/{userid}/{attribut}/
GET:	Attribut anzeigen
PUT:	Attribut ändern
POST:	
DELETE:	Attribut löschen/leeren
Pfad-Parameter:	{userid} – Id des gewünschten Nutzers {attribut} – Das zu ändernde Attribut/
Query-Parameter:	

Tabelle 6.13: REST-Resource: /admin/users/{userid}/{attribut}/

6.3 Marktplatz-Client

Damit die Funktionalitäten der Marktplatz-Logik (siehe Abschnitt 6.2.7) möglichst flexibel und einfach in weitere Java-Projekte und -Benutzeroberflächen integriert werden kann, wurde ein Marktplatz-Client in Java entwickelt. Dieser findet Verwendung in den implementierten Benutzeroberflächen. Abbildung 6.3 zeigt die Ansatzpunkte des Marktplatz-Clients in der

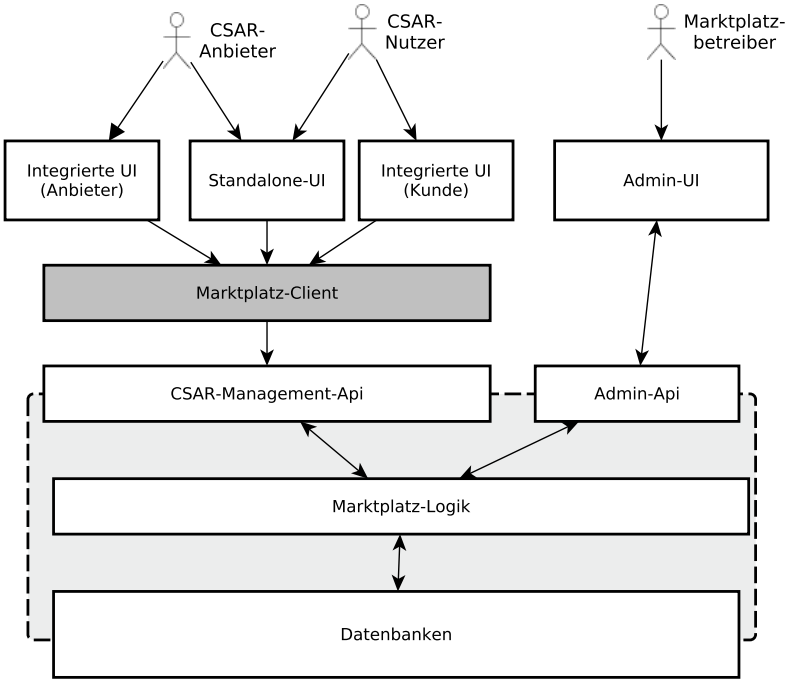

Abbildung 6.3: Marktplatz mit Marktplatz-Client als Proxy

Architektur des Marktplatzes.

6.3.1 Verwendete Technologien

Dieser Unterabschnitt geht kurz auf die verwendeten Technologien im Marktplatz-Client ein und erläutert die zugrunde liegende Motivation hinter diesen.

Jersey-Client-Apache:

Anstatt des Standard-Jersey-Clients wird der Jersey Apache HTTP Client[11] eingesetzt, da dieser eine vereinfachte Benutzung vor allem im Hinblick auf Cookies ermöglicht.

[11] https://jersey.java.net/nonav/apidocs/1.8/contribs/jersey-apache-client/

Listing 6.4 Market-Client Methoden (Java)

```
public interface MarketClientInterface {
       public abstract List<Csar> csarList(String filter);
       public abstract File getCsarFile(String csarId);
       public abstract String getFileName(String csarId);
       public abstract void authenticate(String oauthToken);
       public abstract boolean isAuthenticated();
       public abstract List<String> getCategories();
       public abstract void uploadCsar(String csarName, String csarTags,
                                String fileName, InputStream inCsar);
       public abstract String getCsarDescription(String csarId);
       public abstract void getCsarIconUrl(String csarId);
       public abstract String getCsarPictureUrl(String csarId);
}
```

6.3.2 Angebotene Funktionalität

Der Marktplatz-Client wurde parallel zu den implementierten Benutzeroberflächen entwickelt. Somit deckt die angebotene Funktionalität die derzeit benötigten Schnittstellen für die Benutzeroberflächen ab. Das Java-Interface des Marktplatz-Clients aus Listing 6.4 ist wie folgt beschrieben:

void authenticate(token):

Diese Methode führt die initiale Authentifizierung des Benutzers mittels eines OAuth-Tokens am Marktplatz durch. Dieser Token kann zum Beispiel via GitHub erhalten werden.

boolean isAuthenticated():

Diese Methode überprüft, ob der Marktplatz-Client momentan am Marktplatz angemeldet ist.

List<Csar> csarList(filter):

Mittels dieser Methode erhält der Client eine Liste an verfügbaren CSAR-Dateien. Diese Liste kann optional mittels des Filter-Parameters gefiltert werden.

List<String> getCategories():

Diese Methode ermittelt die verfügbaren Katergorien der sich im Markt befindenden CSAR-Dateien.

String uploadCsar(csarName, csarTags, fileName, inCsar):

Mittels dieser Methode können neue CSAR-Dateien zum Markt hinzugefügt werden. Als Rückgabewert erhält man die Id der erzeugten CSAR-Datei.

String getCsarDescription(String csarId):

Diese Methode liefert die hinterlegte Beschreibung einer CSAR-Datei zurück.

String getCsarIconUrl(String csarId):

Diese Methode ermittelt eine URL, unter welche ein Icon der CSAR-Datei heruntergeladen werden kann.

String getCsarPictureUrl(String csarId):

Diese Methode ermittelt eine URL, unter welche ein Vorschaubild der CSAR-Datei heruntergeladen werden kann.

String getCsarFileName(String csarId):

Mittels dieser Methode erhält man den Dateinamen der gespeicherten CSAR-Datei.

File getCsarFile(String csarId):

Diese Methode liefert die eigentliche CSAR-Datei zurück.

6.4 Integrierte UI (Container)

Folgender Abschnitt behandelt die Implementierung der integrierten Benutzeroberfläche im OpenTOSCA-Container.

6.4.1 Verwendete Technologien

Dieser Abschnitt geht auf die, speziell in der integrierten UI verwendeten, Technologien ein. Die Verwendung dieser Technologien wurde durch die bereits bestehende Benutzeroberfläche von OpenTOSCA gegeben. Daher wird auf diese Technologien und die dahinterstehende Motivation nicht detailliert eingegangen.

Apache Struts:

Apache Struts[12] ist ein Framework, welches dem Entwickler eine weitere Abstraktionsschicht zur Entwicklung von Web-Anwendungen mittels Java EE bietet.

Apache Tiles:

Apache Tiles[13] ist ein Templating-Framework, welches die Entwicklung von Web-Applikationen und Benutzerschnittstellen vereinfachen soll.

JQuery:

JQuery[14] ist eine JavaScript-Bibliothek, welche Methoden zur Manipulation und Traversierung des (HTML)-DOM-Baumes zur Verfügung stellt.

6.4.2 OpenTOSCA-UI – Aktueller Stand

In diesem Abschnitt wird auf den derzeitigen Stand der Benutzeroberfläche des OpenTOSCA-Containers eingegangen. Derzeit unterstützt die OpenTOSCA-Benutzeroberfläche das Hochladen, Durchsuchen, sowie Deployen von CSAR-Dateien. Wie in Abbildung 6.4 zu sehen, sind diese Funktionen mit den Reitern „Overview“ und „Manage“ am oberen Rand zu erreichen.

6.4.3 OpenTOSCA-UI - Erweiterung

Die im vorigen Abschnitt erläuterte Benutzeroberfläche wurde im Zuge dieser Arbeit um einen weiteren Reiter „Marketplace“ erweitert (siehe Abbildung 6.5). Dieser Reiter stellt nun die folgenden Funktionalitäten des Marktplatzes an die OpenTOSCA-UI bereit:

- Anmeldung beim Marktplatz mittels OAuth2 und GitHub.
- Durchsuchen und Anzeigen der verfügbaren CSAR-Dateien im Marktplatz.
- Herunterladen und Importieren von CSAR-Dateien des Marktplatzes.

[12] http://struts.apache.org/
[13] http://tiles.apache.org/
[14] http://jquery.com/

Abbildung 6.4: OpenTOSCA-UI - ohne Marktplatz [OTS13]

6.5 Standalone UI

Zu Entwicklungszwecken wurde zum Anfang dieser Bachelorarbeit eine eigenständige Benutzeroberfläche zur Benutzung des Marktplatzes entwickelt. Diese Benutzeroberfläche wurde in Java und mittels Struts2 realisiert. Da im Laufe der Bachelorarbeit der Fokus sich jedoch zunehmend auf eine in OpenTOSCA integrierte Benutzeroberfläche legte, wurde der Entwicklung einer eigenständigen Benutzeroberfläche vorerst eine mindere Priorität zu eigen.

6.6 Wichtige Datenobjekte

Dieser Abschnitt behandelt die wichtigsten Datenobjekte, welche zwischen einer Benutzeroberfläche und der Marktplatz-Logik ausgetauscht werden. Die Kommunikation kann sowohl im XML- als auch im JSON-Format abgehalten werden. Der Einfachheit halber wird hier jedoch nur auf die Objekte in XML-Darstellung eingegangen. Es werden ausschließlich primitive Datentypen verwendet. Dies ermöglicht eine einfachere Portabilität der Schnittstelle.

Abbildung 6.5: OpenTOSCA-UI - mit Marktplatz

6.6.1 CSAR-Datei

Eine CSAR-Datei repräsentiert eine CSAR im Marktplatz. Alle Elemente beinhalten Verlinkungen über welche die eigentlichen Werte der Elemente abgerufen werden können. Der Name der CSAR-Datei, sowie ihre ID im Marktplatz sind als Attribute in der Nachricht enthalten. Die Verfügbaren Attribute können Listing 6.5 entnommen werden.

6.6.2 CSAR-Liste

Eine CSAR-Liste repräsentiert eine List von CSAR-Dateien. Dieses Datenobjekt beinhaltet eine Liste von CSAR-Links (siehe Abschnitt 6.6.3). Listing 6.6 stellt dies als XML dar.

Listing 6.5 CSAR-Datei (XML)

```
<xs:complexType name="csarResponse"> <xs:sequence>
    <xs:element name="name" type="xs:string" minOccurs="0"/>
    <xs:element name="version" type="xs:string" minOccurs="0"/>
    <xs:element name="file" type="xs:string" minOccurs="0"/>
    <xs:element name="icon" type="xs:string" minOccurs="0"/>
    <xs:element name="description" type="xs:string" minOccurs="0"/>
    <xs:element name="overview" type="xs:string" minOccurs="0"/>
    <xs:element name="tags" type="xs:string" minOccurs="0"/>
  </xs:sequence>
  <xs:attribute name="csarId" type="xs:string"/>
  <xs:attribute name="csarName" type="xs:string"/>
</xs:complexType>
```

Listing 6.6 CSAR-Liste (XML)

```
<xs:complexType name="csarListResponse">
  <xs:sequence>
    <xs:element name="csarList" type="csarLink" minOccurs="0" maxOccurs="unbounded"/>
  </xs:sequence>
</xs:complexType>
```

6.6.3 CSAR-Link

Ein CSAR-Link repräsentiert einen Verweis auf eine CSAR-Datei im Markplatz. Neben des eigentlichen Verweises enthält ein CSAR-Link den CSAR-Namen und die CSAR-Id der referenzierten CSAR. Die XML-Darstellung kann Listing 6.7 entnommen werden.

6.6.4 Value-Response

Eine ValueResponse (Listing 6.8) ist eine generische Antwort, um einen direkten Wert zu übertragen. Da dieser Typ generisch ist, wird dieser im Marktplatz immer benutzt, wenn ein einzelner (primitiver) Wert abgefragt wird.

Listing 6.7 CSAR-Link (XML)

```
<xs:complexType name="csarLink">
  <xs:sequence>
    <xs:element name="link" type="xs:string" minOccurs="0"/>
  </xs:sequence>
  <xs:attribute name="csarId" type="xs:string"/>
  <xs:attribute name="csarName" type="xs:string"/>
</xs:complexType>
```

Listing 6.8 Value-Response (XML)

```
<xs:complexType name="valueResponse">
  <xs:sequence>
    <xs:element name="value" type="xs:string" minOccurs="0"/>
  </xs:sequence>
</xs:complexType>
```

Listing 6.9 Array-Value-Response (XML)

```
<xs:complexType name="arrayValueResponse">
  <xs:sequence>
    <xs:element name="values" type="xs:string" maxOccurs="unbounded"/>
  </xs:sequence>
</xs:complexType>
```

6.6.5 Array-Value-Response

Eine Array-Value-Response (Listing 6.9) wird benutzt, um eine Liste/ein Array an Werten als Antwort zu übergeben. Dieser Antwort-Typ ist generisch und wird benutzt, wenn eine Liste an (primitiven) Werten übergeben wird.

6.7 Sequenzdiagramme

Um die Funktion des Marktplatzes und das Zusammenspiel der Datenbanken besser zu erläutern, werden in diesem Abschnitt die Anwendungsfälle aus Abschnitt 5.2 nochmals mittels Sequenzdiagrammen im Detail beschrieben.

6.7.1 Sequenzdiagramm: Am Marktplatz anmelden

Abbildung 6.6 beschreibt den Anwendungsfall aus Abschnitt 5.2.1 als Sequenzdiagramm. Der Ablauf kann folgendermaßen beschrieben werden:

1. Der Nutzer möchte sich einloggen.
2. Die Benutzeroberfläche fordert eine Login-URL vom Marktplatz an.
3. Diese Login-URL wird der Benutzeroberfläche zurückgegeben.
4. Die Benutzeroberfläche leitet den Benutzer auf diese URL weiter.
5. Der Nutzer bestätigt den Zugriff auf seine Daten und wird zurück auf den Marktplatz geleitet.
6. Durch die Umleitung des Benutzers auf die Benutzeroberfläche erhält diese einen OAuth-Code vom Benutzer.
7. Die Benutzeroberfläche gibt diesen Code an die Marktplatz-Logik weiter.
8. Die Marktplatz-Logik tauscht den Code bei dem OAuth-Provider in einen Token um und speichert diesen.
9. Eine Statusmeldung wird bis zum Nutzer propagiert.

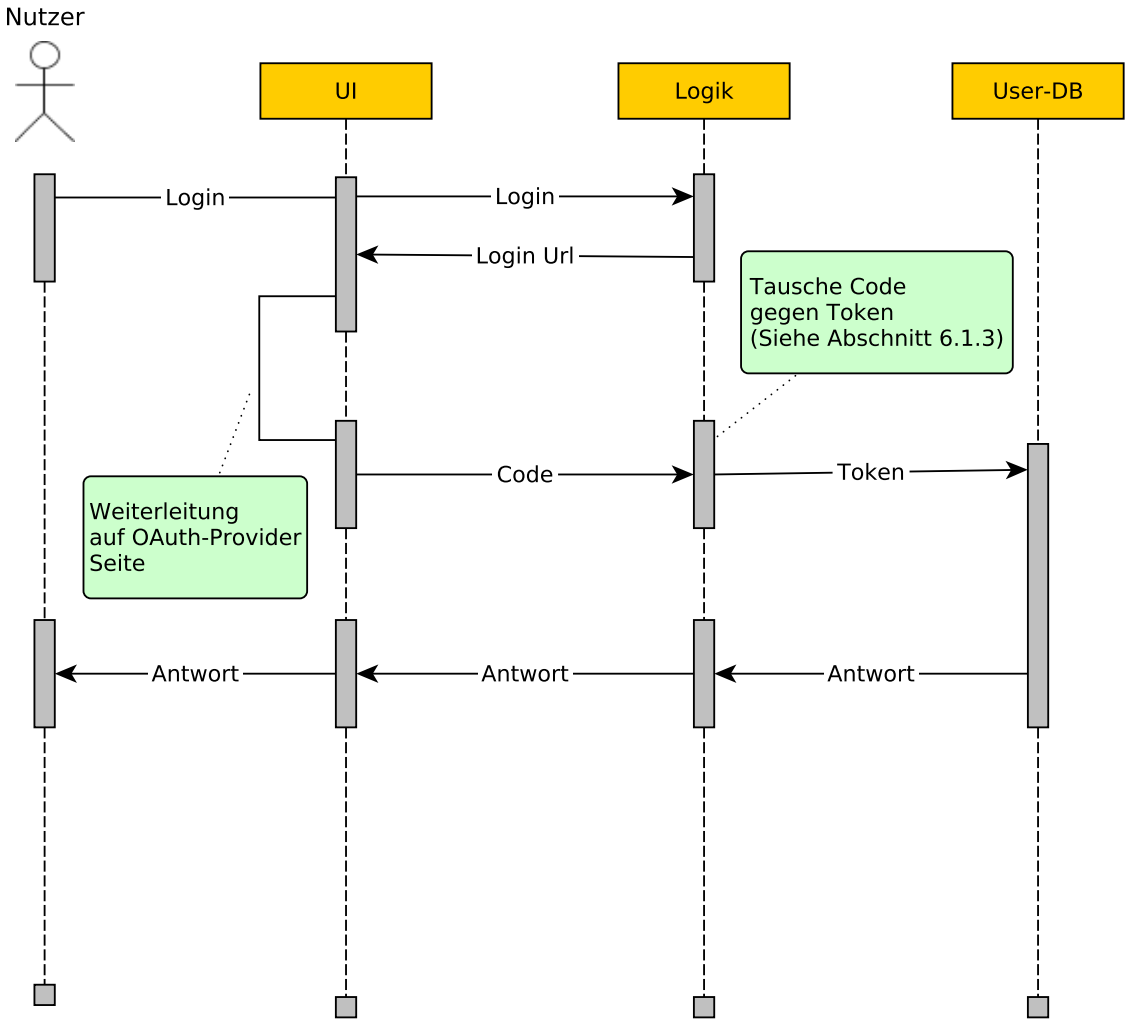

Abbildung 6.6: Sequenzdiagramm: Anmeldung

6.7.2 Sequenzdiagramm: CSAR-Dateien suchen

Abbildung 6.7 beschreibt den Anwendungsfall aus Abschnitt 5.2.2 als Sequenzdiagramm. Der Ablauf kann wie folgt beschrieben werden:

1. Der Nutzer fordert eine CSAR-Liste an.
2. Die Benutzeroberfläche gibt dies an die Marktplatz-Logik weiter.
3. Die Marktplatz-Logik fordert eine CSAR-Liste bei der CSAR-DB an.
4. Die Liste wird bis zur Benutzeroberfläche zurückgegeben.
5. Die Benutzeroberfläche fordert für jede CSAR der Liste nun mittels der CSAR-ID bei der Marktplatz-Logik ein Icon an.
6. Die Marktplatz-Logik fordert bei der CSAR-DB die IconFileID mittels der CSAR-ID an.
7. Mittels der IconFileID fordert die Marktplatz-Logik bei der File-DB nun das eigentliche Icon an und gibt dieses zurück.
8. Das Icon wird bis zur Benutzeroberfläche zurückgegeben und angezeigt.

6.7.3 Sequenzdiagramm: CSAR-Datei im Detail anschauen

Abbildung 6.8 beschreibt den Anwendungsfall aus Abschnitt 5.2.3 als Sequenzdiagramm. Der Ablauf kann wie folgt beschrieben werden:

1. Der Benutzer möchte die Metadaten einer CSAR im Detail anschauen.
2. Die Benutzeroberfläche fordert eine Link-Liste zu den CSAR-Attributen mittels der CSAR-ID an.
3. Die Marktplatz-Logik gibt eine Link-Liste zu den Attributen zurück.
4. Die Benutzeroberfläche fragt für jedes gewünschte Attribut bei der Marktplatz-Logik den Wert ab.
5. Die Marktplatz-Logik ermittelt den Wert des Attributes mittels der CsarId und der CSAR-DB.
6. Die Marktplatz-Logik gibt die Attribut-Werte zurück und die Benutzeroberfläche zeigt diese an.

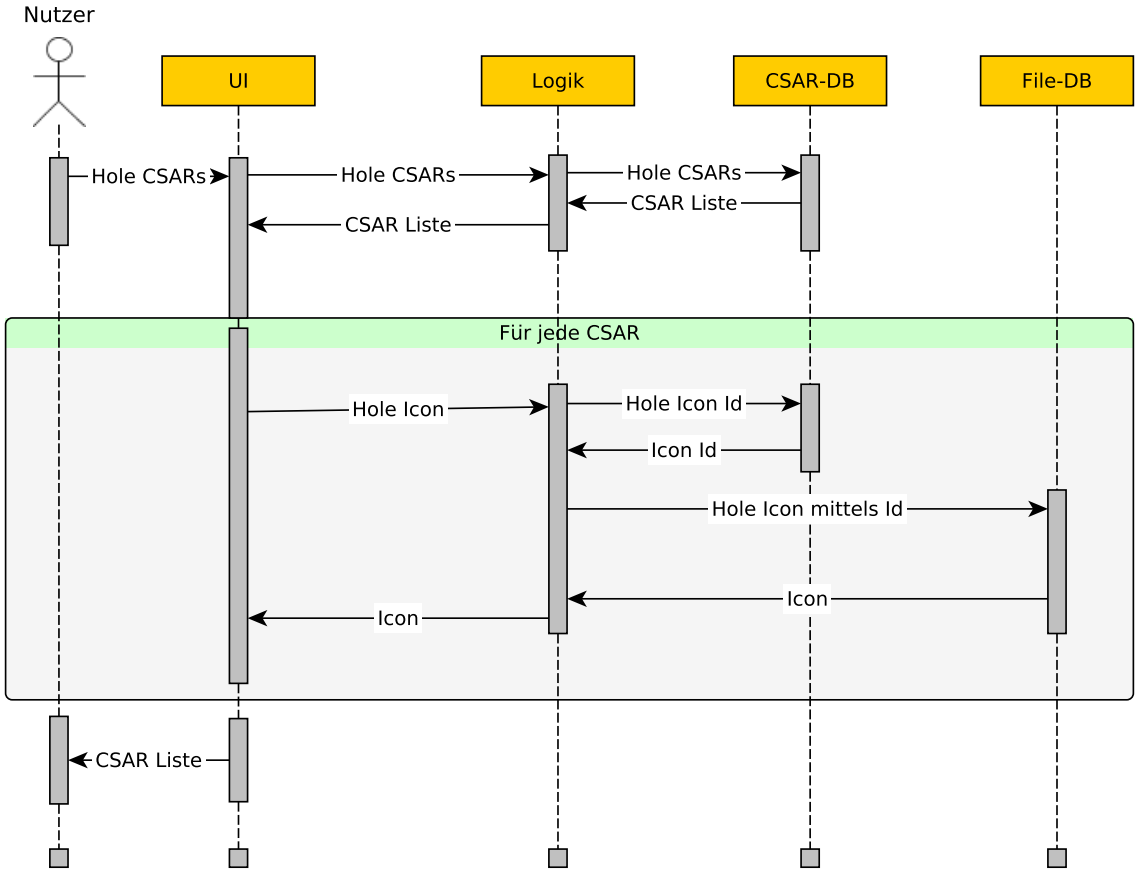

Abbildung 6.7: Sequenzdiagramm: CSAR-Dateien suchen

6.7.4 Sequenzdiagramm: CSAR-Datei herunterladen

Abbildung 6.9 beschreibt den Anwendungsfall aus Abschnitt 5.2.4 als Sequenzdiagramm. Der Ablauf kann wie folgt beschrieben werden:

1. Der Benutzer möchte die CSAR-Datei einer CSAR herunterladen.
2. Die Benutzeroberfläche übergibt die CsarId an die Marktplatz-Logik.
3. Die Marktplatz-Logik fordert mittels der CsarId bei der CSAR-DB die CsarFileId an.
4. Mittels der CsarFileId fordert die Marktplatz-Logik die eigentliche Datei bei der File-DB an.
5. Die File-DB liefert die Datei zurück.
6. Die Datei wird bis zum Benutzer weitergereicht.

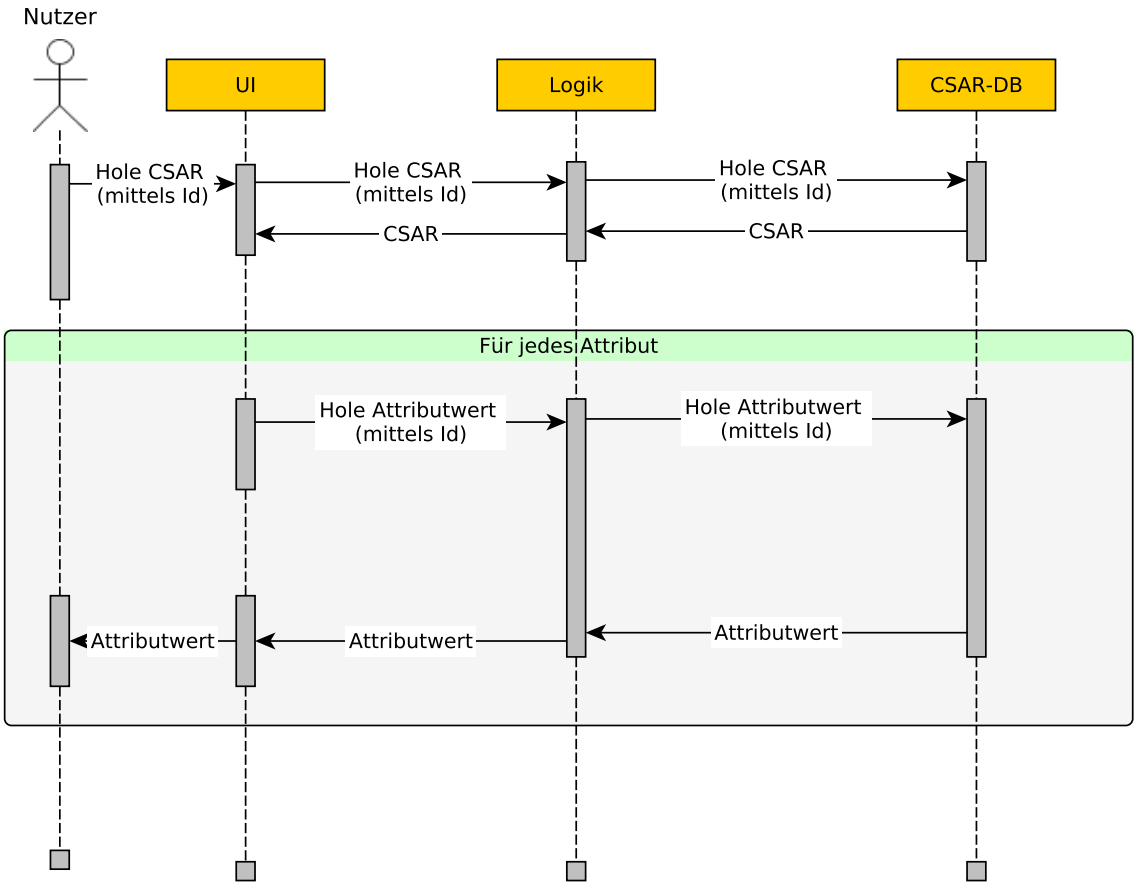

Abbildung 6.8: Sequenzdiagramm: CSAR-Datei im Detail anschauen

6.7.5 Sequenzdiagramm: Eine CSAR-Datei zum Marktplatz hinzufügen

Abbildung 6.10 beschreibt den Anwendungsfall aus Abschnitt 5.2.5 als Sequenzdiagramm. Der Ablauf kann wie folgt beschrieben werden:

1. Der Nutzer möchte eine neue CSAR zum Marktplatz hinzufügen und übergibt der Benutzeroberfläche die benötigten Metadaten und eine CSAR-Datei.
2. Die Benutzeroberfläche übergibt diese Daten an die Marktplatz-Logik.
3. Die Marktplatz-Logik erstellt eine neue CSAR in der CSAR-DB und erhält eine CsarId.
4. Die Marktplatz-Logik speichert die (eigentliche) CSAR-Datei mittels der File-DB und erhält eine CsarFileId.
5. Die Marktplatz-Logik speichert die CsarFileId in der CSAR-DB und gibt die CsarId zurück.

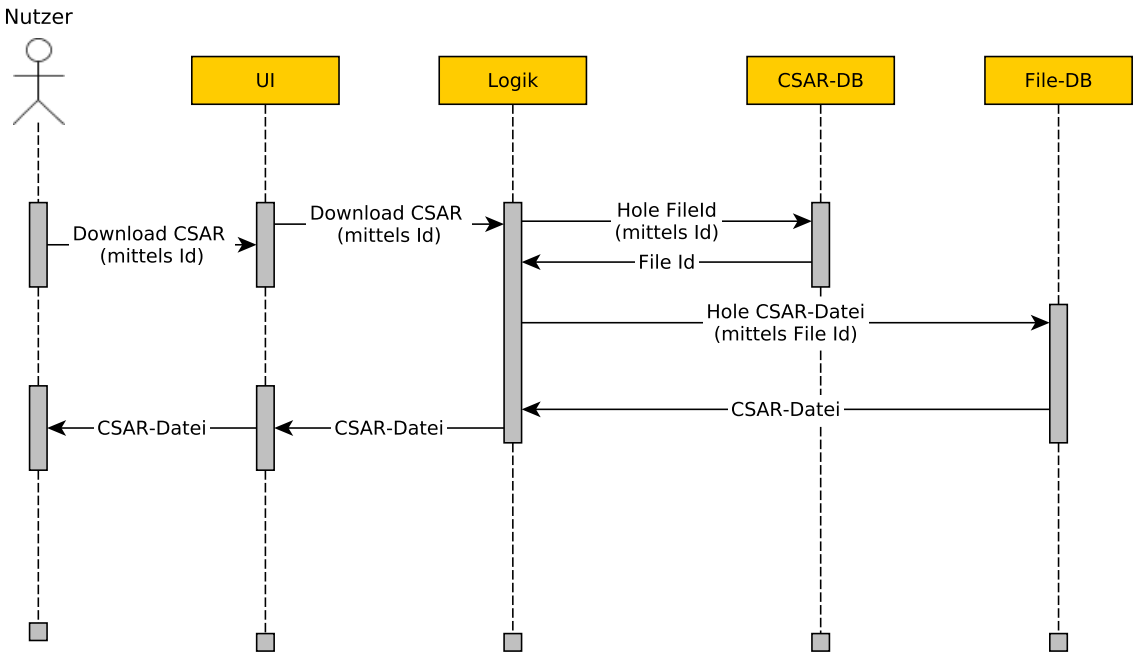

Abbildung 6.9: Sequenzdiagramm: CSAR-Datei herunterladen

6.7.6 Sequenzdiagramm: Eine CSAR-Datei ändern

Abbildung 6.11 beschreibt den Anwendungsfall aus Abschnitt 5.2.6 als Sequenzdiagramm. Der Ablauf kann wie folgt beschrieben werden:

1. Der Nutzer möchte die Metadaten einer CSAR-Datei ändern und übergibt die neuen Metadaten an die Benutzeroberfläche.
2. Die Benutzeroberfläche übergibt die neuen Metadaten und die CsarId an die Marktplatz-Logik.
3. Die Marktplatz-Logik speichert die neuen Daten in der CSAR-DB.

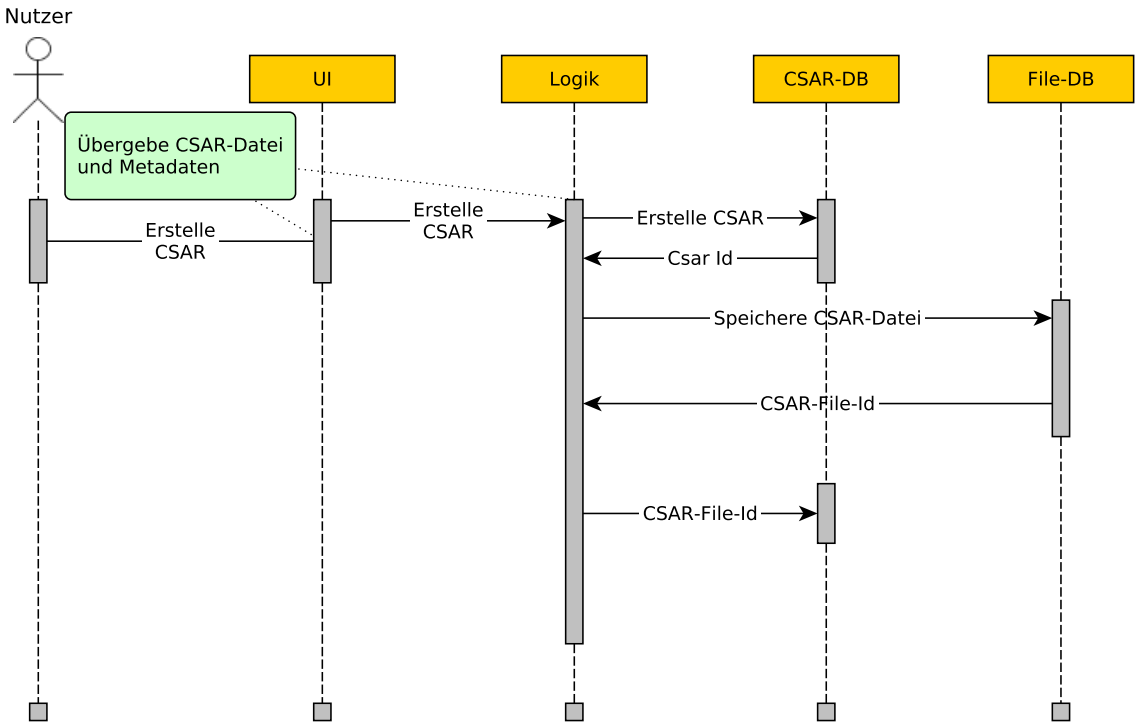

Abbildung 6.10: Sequenzdiagramm: CSAR-Datei zum Marktplatz hinzufügen.

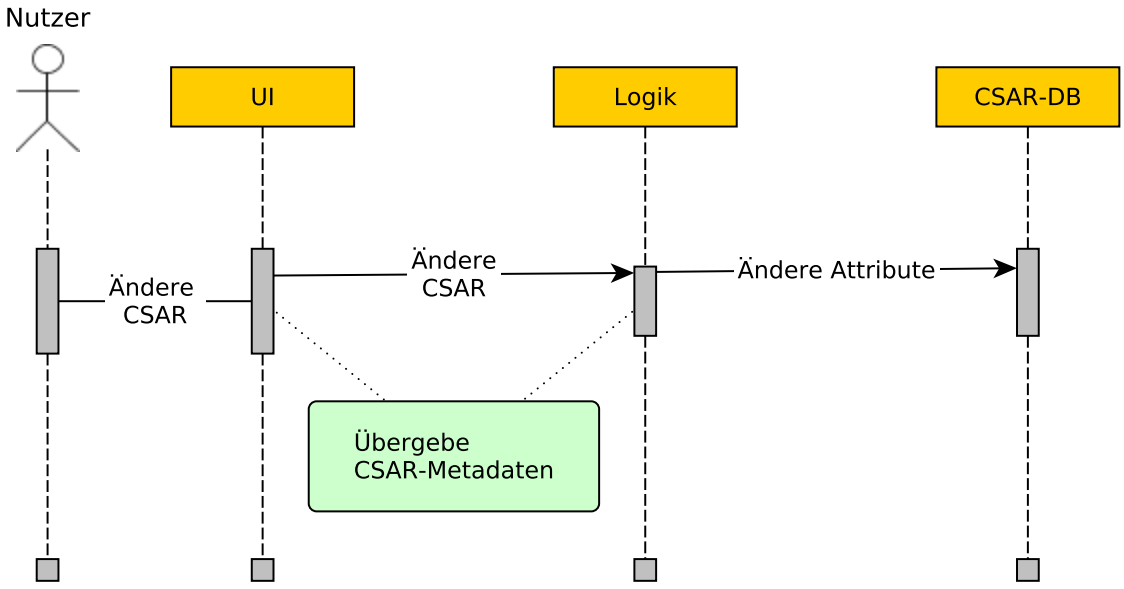

Abbildung 6.11: Sequenzdiagramm: Eine CSAR-Datei ändern.

6.7.7 Sequenzdiagramm: Eine CSAR-Datei löschen

Abbildung 6.12 beschreibt den Anwendungsfall aus Abschnitt 5.2.7 als Sequenzdiagramm. Der Ablauf kann wie folgt beschrieben werden:

1. Der Nutzer möchte eine CSAR löschen.
2. Die UI übergibt den Löschbefehl und die CsarId an die Logik.
3. Die Marktplatz-Logik holt mittels der CsarId die Metadaten der CSAR und löscht diese anschließend aus der CSAR-DB.
4. Mittels der Metadaten (CsarFileId, IconFileId, ImageFileId) löscht die Marktplatz-Logik die Dateien aus der File-DB.

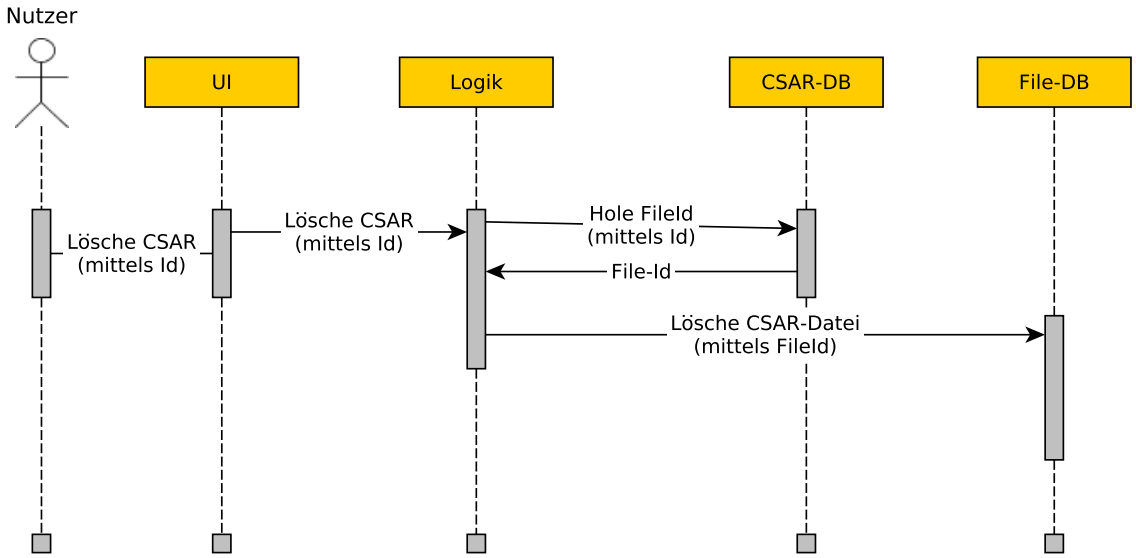

Abbildung 6.12: Sequenzdiagramm: Eine CSAR-Datei löschen.

7 Zusammenfassung und Ausblick

Ziel dieser Bachelorarbeit war es, grundlegende Konzepte und Anwendungsfälle eines Marktplatzes für TOSCA zu erarbeiten und zu implementieren. Zuerst wurden die zugrunde liegenden Konzepte eines Marktplatzes im TOSCA-Umfeld beschrieben, sowie die darin agierenden Rollen definiert. Unter Berücksichtigung dieser agierenden Rollen und der Konzepte wurde ein möglicher Entwurf und eine Architektur eines Marktplatzes geschaffen, sowie die darin enthaltenen Komponenten definiert (Kapitel 5).

Nach der Evaluierung möglicher Ansätze wurde eine konkrete Implementierung eines Marktplatzes, sowie einer in OpenTOSCA integrierten Benutzeroberfläche vorgestellt. Die zuvor abstrakten Konzepte wurden konkret umgesetzt, sowie das Zusammenspiel dieser Komponenten beschrieben (Kapitel 6).

In Zukunft kann der, in dieser Arbeit vorgestellte, Marktplatz noch weiter erweitert werden. So ist die Integration von Zahlungsdienstleistern (wie bspw. PayPal[1] oder Google Checkout [2]) in den Marktplatz denkbar.

Weiterhin ist eine Weiter- oder Neuentwicklung der Standalone UI (siehe Abschnitt 5.4.1) denkbar.

[1] http://www.paypal.com
[2] http://checkout.google.com/

Literaturverzeichnis

[Apa12] Apache Software Foundation, 2012. URL http://www.apache.org/licenses/GPL-compatibility.html. (Zitiert auf Seite 19)

[Arh13] A. Arhipov. Java Build Tools Survey, 2013. URL http://arhipov.blogspot.de/2013/08/java-build-tools-survey-results.html. (Zitiert auf Seite 46)

[BBH+13] T. Binz, U. Breitenbücher, F. Haupt, O. Kopp, F. Leymann, A. Nowak, S. Wagner. OpenTOSCA – A Runtime for TOSCA-based Cloud Applications. In *11th International Conference on Service-Oriented Computing (ICSOC 2013)*, LNCS. Springer, 2013. (Zitiert auf Seite 13)

[BBK+12] U. Breitenbücher, T. Binz, O. Kopp, F. Leymann, D. Schumm. Vino4TOSCA: A Visual Notation for Application Topologies based on TOSCA. In *Proceedings of the 20th International Conference on Cooperative Information Systems (CoopIS 2012)*, LNCS. Springer-Verlag, 2012. (Zitiert auf den Seiten 12 und 14)

[BBKL14] T. Binz, U. Breitenbücher, O. Kopp, F. Leymann. *TOSCA: Portable Automated Deployment and Management of Cloud Applications*, S. 527–549. Advanced Web Services. Springer, New York, 2014. doi:10.1007/978-1-4614-7535-4_22. (Zitiert auf Seite 11)

[FKW13] M. Fischer, K. Kepes, A. Wassiljew. Vergleich von Frameworks zur Implementierung von REST basierten Anwendungen. Fachstudie Softwaretechnik: Universität Stuttgart, Institut für Architektur von Anwendungssystemen, 2013. URL http://www2.informatik.uni-stuttgart.de/cgi-bin/NCSTRL/NCSTRL_view.pl?id=FACH-0170&engl=0. (Zitiert auf Seite 46)

[Gab13] Gabler Wirtschaftslexikon Online, 2013. URL http://wirtschaftslexikon.gabler.de/. (Zitiert auf Seite 14)

[KBBL13] O. Kopp, T. Binz, U. Breitenbücher, F. Leymann. Winery – A Modeling Tool for TOSCA-based Cloud Applications. In *11th International Conference on Service-Oriented Computing (ICSOC 2013)*, LNCS. Springer, 2013. (Zitiert auf Seite 14)

[NKP13] P. Niehues, T. Kunz, L. Posiadlo. Das CloudCycle-Ökosystem. Technischer Bericht, CloudCycle, 2013. (Zitiert auf den Seiten 14 und 23)

[OAS13] OASIS. *Topology and Orchestration Specification for Cloud Applications (TOSCA) Primer Version 1.0*, 2013. (Zitiert auf den Seiten 11, 13 und 23)

[OTS13] OpenTOSCA Website, 2013. URL http://www.opentosca.org. (Zitiert auf den Seiten 6, 13 und 64)

[PKW07] PKWARE. ZIP File Format Specification. Technischer Bericht version 6.3.2, PKWARE, 2007. URL http://www.pkware.com/documents/casestudies/APPNOTE.TXT. (Zitiert auf Seite 13)

[PM10] F. Pavel, A. Mattes. Cloud-Computing: Großes Wachstumspotenzial. *Wochenbericht des DIW Berlin Nr. 48/2010*, 2010. (Zitiert auf Seite 9)

[Tor12] J. Torke, 2012. URL http://pixeltuner.de/12-open-source-shopsysteme-im-vergleich/. (Zitiert auf Seite 17)

[TYK+12] J. Trammel, Yalçinalp, A. Kalfas, J. Boag, D. Brotsky. Device Token Protocol for Persistent Authentication Shared across Applications. In F. Paoli, E. Pimentel, G. Zavattaro, Herausgeber, *Service-Oriented and Cloud Computing*, Band 7592 von *Lecture Notes in Computer Science*, S. 230–243. Springer Berlin Heidelberg, 2012. doi:10.1007/978-3-642-33427-6_20. (Zitiert auf Seite 47)

Alle URLs wurden zuletzt am 04. 10. 2013 geprüft.

Erklärung

Ich versichere, diese Arbeit selbstständig verfasst zu haben. Ich habe keine anderen als die angegebenen Quellen benutzt und alle wörtlich oder sinngemäß aus anderen Werken übernommene Aussagen als solche gekennzeichnet. Weder diese Arbeit noch wesentliche Teile daraus waren bisher Gegenstand eines anderen Prüfungsverfahrens. Ich habe diese Arbeit bisher weder teilweise noch vollständig veröffentlicht. Das elektronische Exemplar stimmt mit allen eingereichten Exemplaren überein.

Ort, Datum, Unterschrift