One Corridor – One Strategy

Joint Regional Development for the North-South Corridor

Dipl.-Kfm. René Föhring • Dipl.-Kff. Adina Silvia Bruns

Implementierung des Software-Prototyps einer Online-Frachtenbörse

CODE24-Projektbericht Nr. 3

ISSN 1866-9255

UNIVERSITÄT DUISBURG ESSEN

Abstract

Zurzeit existieren auf dem Markt zahlreiche Lkw-Online-Frachtenbörsen, aber nur sehr wenige und in der betrieblichen Praxis nur rudimentär genutzte Online-Frachtenbörsen für den Schienengüterverkehr. Vor diesem Hintergrund wird im vorliegenden Projektbericht ein Softwareprototyp einer Online-Frachtenbörse entwickelt, der die Anforderungsanalyse, die im Projektbericht Nr. 2 (Klippert/Kowalski/Bruns (2010)) dargestellt wurde, auf den Schienengüterverkehr anwendet.

Inhaltverzeichnis

1 Das Projekt CODE24

Das Realproblem dieses Projektberichts[1] ergibt sich aus den Bemühungen der Europäischen Union, den paneuropäischen Schienengüterverkehr auf der Nord-Süd-Transversale zwischen Rotterdam und Genua zu stärken. Dieses Projektvorhaben wird als „Projekt CODE24“ bezeichnet.[2] Zur Koordination und Unterstützung grenzüberschreitender Güterverkehre für den Verkehrsträger Schiene ist im Rahmen dieses Projekts auch die Implementierung einer europaweiten Online-Frachtenbörse primär für den Schienengüterverkehr geplant.

1) Dieser Projektbericht basiert auf der Diplomarbeit von R. Föhring: Implementierung eines Softwareprototypen einer Online-Frachtenbörse zur Anforderungsanalyse im Schienengüterverkehr.

2) Weitere Informationen sind im Internet unter der URL http://www.code-24.eu/ verfügbar.

2 Wissenschaftliches Problem

In der einschlägigen Fachliteratur wird dem Thema „Online-Frachtenbörse für den Schienengüterverkehr" bisher kaum Beachtung geschenkt. Als wissenschaftliche Quelle zu den Anforderungen an eine Online-Frachtenbörse für den Schienengüterverkehr dient die empirische Forschungsarbeit von Frau Klippert, Herrn Kowalski und Frau Bruns.[1]

In der betrieblichen Praxis existiert derzeit keine europaweite Online-Frachtenbörse für den Schienengüterverkehr und auch auf nationaler Ebene lassen sich nur wenige Frachtenbörsen identifizieren, welche als Konkurrenz zu einer neu am Markt auftretenden europaweiten Online-Frachtenbörse für den Schienengüterverkehr anzusehen wären.[2]

Eine europaweit operierende Online-Frachtenbörse für den Schienengüterverkehr ist jedoch betriebswirtschaftlich wünschenswert, um so die Zusammenarbeit der verschiedenen nationalen Logistikdienstleister zu unterstützen und die durch das Projekt CODE24 angestrebte Stärkung des Schienengüterverkehrs in Europa zu fördern.

Aufgrund oben beschriebener Situation in der wissenschaftlichen Literatur und in der betrieblichen Praxis gestaltet es sich schwierig, auf Erfahrungswerte bei der Implementierung einer solchen Software zurückzugreifen. Es mangelt an Erkenntnissen darüber, welche Erfolgsfaktoren für die Planung, Realisierung und den Betrieb einer Online-Frachtenbörse entscheidend sind.

Zwischen dem Stand der Forschung, also der Tatsache, dass die theoretische und empirische Forschung dem Thema „Online-Frachtenbörse für den Schienengüterverkehr" bisher kaum Beachtung geschenkt hat, und der betriebswirtschaftlich wünschenswerten Realität, also der Existenz einer europaweit operierenden Online-Frachtenbörse für den Schienengüterverkehr, besteht somit eine Lücke. Der vorliegende Projektbericht dient dem Schließen dieser Lücke durch die teilweise Implementierung der vorliegenden Erkenntnisse über die Anforderungen an eine solche Software, um mithilfe des resultierenden Softwareprototyps die weitere Anforderungsanalyse zu unterstützen.[3]

Somit liegt in der Implementierung eines evolutionären[4] Softwareprototyps einer Online-Frachtenbörse zur Anforderungsanalyse im Schienengüterverkehr das wissenschaftliche Problem dieses Projektberichts.

---

1) Vgl. Klippert/Kowalski/Bruns (2010).

2) Die als gering eingeschätzte Konkurrenzfähigkeit der bestehenden Online-Frachtenbörsen folgt aus der Plausibilitätsüberlegung, dass diese entweder nur national agieren oder sich nicht ausschließlich auf den Verkehrsträger Schiene konzentrieren, wodurch am Markt eine Nische für eine europaweite Online-Frachtenbörse für den Schienengüterverkehr entsteht.

3) Zum Umfang der Implementierung vgl. S. 31.

4) Evolutionäre Prototypen werden, im Gegensatz zu Wegwerfprototypen, nicht dediziert für einen Einsatz entwickelt und anschließend „weggeworfen", sondern stetig (mithilfe der mit ihnen gewonnen Erkenntnisse) weiterentwickelt; vgl. S. 7. Zu den Gründen der Implementierung eines evolutionären Prototyps anstelle eines Wegwerfprototyps vgl. S. 7. Zum Einsatz von Prototypen zur Anforderungsanalyse vgl. Berenbach et al. (2009), S. 248 ff., Pohl/Rupp (2009), S. 39, Sommerville (2007), S. 409 ff., Hull/Jackson/Dick (2005), S. 103 sowie S. 167, Aurum/Wohlin (2005), S. 174 f., sowie Rupp (2004), S. 61 f.

3 Begriffsabgrenzungen

3.1 Online-Frachtenbörse

Der Begriff der Online-Frachtenbörse ist keineswegs eindeutig besetzt. Es existiert aktuell im europäischen Raum eine Vielzahl von Marktplätzen, welche sich als Marktführer präsentieren und Transporte sowie komplementäre Dienstleistungen aller Art für einen oder mehrere Verkehrsträger anbieten.[1] Aus der Tatsache, dass all diese Marktplätze ähnlichen, aber nicht gleichen Charakters sind und in ihren Bezeichnungen Begriffe wie „Fracht", „Laderaum", „Frachten", „Transport" und „Börse", „Marktplatz" sowie die Zusätze „elektronisch", „online" oder „net" scheinbar willkürlich kombinieren, folgt die Notwendigkeit, den Begriff der Online-Frachtenbörse im Rahmen dieses Projektberichts näher zu präzisieren.

Der Begriff „Online-Frachtenbörse für den Schienengüterverkehr" steht im Rahmen dieses Projektberichts für einen elektronischen Marktplatz, der Angebot und Nachfrage zusammenbringt und es seinen Teilnehmern so ermöglicht, an einem zentralen, aber virtuellen Ort unter Verwendung von Internettechnologien schienengebundene Transporte und komplementäre Dienstleistungen untereinander zu handeln und zu koordinieren.[2] Es können über den jedem frei zugänglichen Marktplatz Fracht und Laderaum auf dem Kontraktmarkt gehandelt werden.[3] Explizit gefördert werden soll auch das Auffinden neuer Geschäftspartner. Ausgeschlossen werden in diesem Projektberichts alle Marktplätze, die Kurier-, Express- und Paketdienste anbieten, sowie sämtliche Angebote, die sich nicht primär an den Verkehrsträger Schiene wenden.

3.2 Softwareprototyp

Ein Softwareprototyp ist im Rahmen dieses Projektberichts ein Instrument der Anforderungsanalyse und dient der Erhebung neuer Anforderungen, der Identifizierung fehlender Anforderungen sowie der Validierung existierender Anforderungen durch Überprüfung ihrer Korrektheit und Realisierbarkeit sowie der Identifizierung von Fehlkonzeptionen, Unvollständigkeit und Inkonsistenzen.[4] Darüber hinaus ermöglicht ein Softwareprototyp verschiedene Alternativen gegeneinander abzuwägen, bspw. bei der softwaretechnischen Abbildung einzelner Geschäftsprozesse und der Gestaltung von Benutzeroberflächen und Benutzerführung.[5] Zudem kann ein Softwareprototyp helfen, Schwächen in bestehenden Erkenntnissen der Anforderungsanalyse zu entdecken, da manche Funktionen in einer gedruckten Spezifikation sinnhaft erscheinen, sich im Zusammenspiel mit anderen Funktionen jedoch als fehlerhaft oder unvollständig erweisen.[6] Daraus ergibt sich der primäre Nutzen eines Softwareprototyps: Er ist von zentraler Bedeutung beim Schließen der Lücke zwischen Beschrei-

---

1) Beispiele für derartige Marktplätze sind zum Zeitpunkt der Erstellung dieses Projektberichts die Plattformen benelog (http://www.benelog.com/), LKWonline (http://www.lkwonline.de/), kurierportal.com (http://www.kurierportal.com/), World Transport Networking (http://www.de.wtransnet.com/) und TranspoLog (http://www.stoffstrom.de/transpolog.htm).

2) Vgl. Pankratz (2003), S. 3 f.

3) Zu den grundlegenden Überlegungen bzgl. des Zugangs und Funktionsumfangs der Online-Frachtenbörse vgl. Klippert/Kowalski/Bruns (2010), S. 44 ff.

4) Vgl. Sommerville (2007), S. 409, sowie Aurum/Wohlin (2005), S. 174.

5) Vgl. Sommerville (2007), S. 409.

6) Vgl. Sommerville (2007), S. 410.

bung und Implementierung einer Software.[1] Insbesondere bei Webapplikationen kann die Implementierung eines funktionsfähigen Prototyps weniger aufwendig sein, als einen ebenfalls visuellen, aber nicht funktionsfähigen Prototypen zu realisieren.[2]

3.3 System der Online-Frachtenbörse

Der Begriff „System der Online-Frachtenbörse" steht für das aus Softwareprototyp und Benutzern gebildete interaktive, soziale System. So stellt der Softwareprototyp viele Funktionen nicht direkt zur Verfügung. Der Softwareprototyp erlaubt vielmehr bspw. das Einstellen, Filtern und Auffinden von Inseraten, Anlegen von Anbieterprofilen und Bewerten von Geschäftspartnern. Erst durch die soziale Interaktion der Benutzer (durch Nutzung der durch den Softwareprototyp bereitgestellten Funktionen) werden durch das System der Online-Frachtenbörse Funktionen wie das „Zusammenbringen von Angebot und Nachfrage" oder das „Auffinden neuer Geschäftspartner" bereitgestellt.

Abbildung 1 stellt das System der Online-Frachtenbörse und seine Komponenten dar.[3]

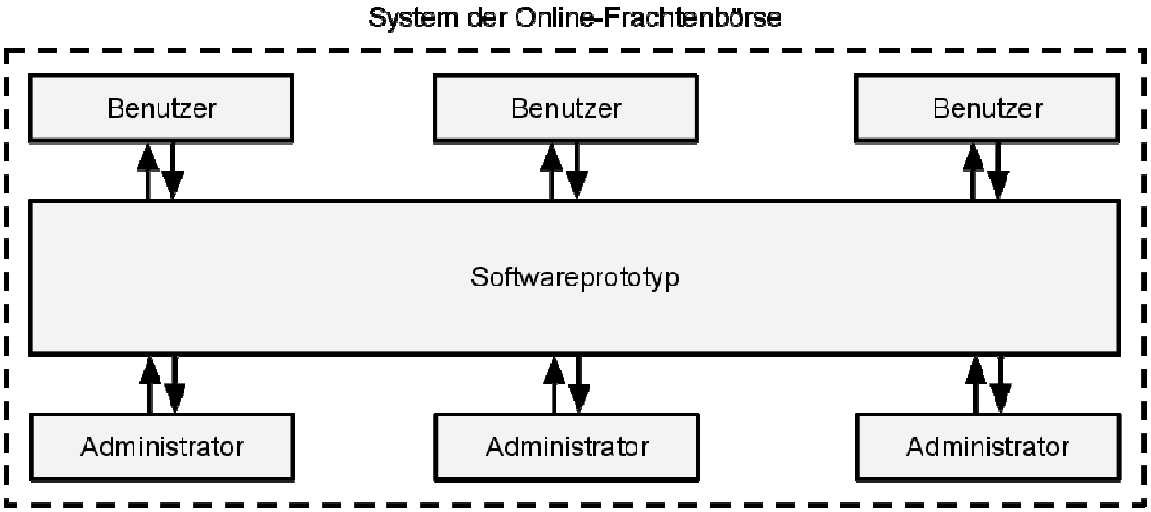

Abbildung 1: Darstellung der Interaktion zwischen Benutzern und Softwareprototyp

3.4 Benutzer, Administratoren, Frontend und Backend

Ein Benutzer[4] im weiteren Sinne ist ein Mensch, der mit der Software direkt interagiert. Benutzer im engeren Sinne interagieren mit der Software nur über das Frontend. Ein Administrator ist ein Benutzer, der mit der Software auch über das Backend interagiert.

Das Frontend ist der Teil des Softwareprototyps, welcher für alle Benutzer zugänglich ist. Einige Teile des Frontends, wie bspw. die Startseite oder das Registrierungsformular, sind allen Internetnutzern zugänglich, andere nur im System der Online-Frachtenbörse registrierten Benutzern.

---

1) Vgl. Aurum/Wohlin (2005), S. 174.

2) Vgl. Berenbach et al. (2009), S. 249.

3) Diese und alle weiteren Abbildungen in diesem Projektbericht sind eigene Darstellungen der Autoren.

4) Aus Gründen der Lesbarkeit wird in diesem Projektbericht der Begriff „Benutzer" verwendet. Er bezieht sich sowohl auf eine weibliche Benutzerin als auch auf einen männlichen Benutzer. Weitere männliche Personenbezeichnungen in diesem Projektbericht gelten sinngemäß ebenso für Personen weiblichen Geschlechts.

Das Backend ist der Teil des Softwareprototyps, der funktional lediglich den Betreibern der Online-Frachtenbörse zugänglich ist und über den die Online-Frachtenbörse von ihnen administriert werden kann.

3.5 Anbieter, Anbieterprofile und Geschäftspartner

Anbieter sind im System der Online-Frachtenbörse registrierte Unternehmen.[1]

Alle Anbieter haben ein Anbieterprofil, auf welchem sie Informationen, wie bspw. Kontaktdaten, Ansprechpartner und Bewertungen, veröffentlichen können.

Geschäftspartner sind Anbieter, die bereits miteinander in geschäftlicher Beziehung standen.

3.6 Applikation, Datenbank und Benutzeroberfläche

Der Softwareprototyp besteht wie jede Software aus mehreren Komponenten, die ihrerseits wiederum aus einer Vielzahl von Modulen und Klassen bestehen. Zur Begriffsabgrenzung und genaueren Beschreibung der Struktur und der Prozesse unterteilt dieser Projektbericht[2] die vorliegende Software in folgende Hauptkomponenten: Applikation, Datenbank und Benutzeroberfläche.

Die Applikation ist der Teil des Softwareprototyps, der entscheidet, welche Prozesse zu welchem Zeitpunkt in welcher Weise ausgeführt und welche Daten zu diesem Zweck geladen und gespeichert werden. Sie stellt somit die Geschäftslogik bereit und bestimmt die ablaufenden Prozesse und auszugebenden Inhalte.

Die Datenbank ist für die persistente Speicherung aller für die Applikation relevanten Daten verantwortlich.

Die Benutzeroberfläche ist der Teil des Softwareprototyps, mit dem ein Benutzer direkt interagiert. Sie stellt die visuelle Präsentation der Software für den Benutzer, also im vorliegenden Fall die Webseite, dar.

Abbildung 2 stellt dar, welche Komponente des Softwareprototyps mit dem Benutzer oder anderen Komponenten interagiert.

---

1) Genauer betrachtet sind Mitarbeiter eines Unternehmens als Benutzer im System der Online-Frachtenbörse registriert und erstellen für ihr Unternehmen ein Anbieterprofil. Der Begriff Anbieter bezeichnet also in der Regel das jeweilige Unternehmen eines Benutzers.

2) Eine weitere Möglichkeit zur Beschreibung des Softwareprototyps wäre das MVC-Modell, das eine oberflächenorientierte Software ebenfalls in drei Komponenten, hier „Schichten" genannt, unterteilt: Model-, View- und Controller-Schicht. Die Model-Schicht ist für die Datenmodellierung, die View-Schicht für die Datenpräsentation und die Controller-Schicht für die Programmsteuerung zuständig, vgl. Ammelburger/Scherer (2008), S. 6 ff. Die bewusste Abweichung vom MVC-Modell zur Beschreibung der Komponenten des Softwareprototyps soll eine trennschärfere Zuordnung einzelner Funktionen zu ihren Komponenten im Rahmen dieses Projektberichts ermöglichen.
Die Applikation wäre im MVC-Modell die Model- und Controller-Schicht, die Datenbank das Speichermedium der Model-Schicht und die Benutzeroberfläche die View-Schicht. Allerdings stellt sich hier die Frage, ob bspw. eine XML-Schnittstelle zur Anbindung fremder Software zu der View- oder Controller-Schicht gehört, da die ausgelieferten XML-Daten für das fremde System quasi die Benutzeroberfläche, also die View-Schicht darstellen, andererseits aber von der Controller-Schicht direkt ausgeliefert werden. In der vorliegenden Unterteilung gehört eine XML-Schnittstelle zur Anbindung von Drittanbietersoftware eindeutig zu der Applikation, da sie ein abstraktes Modell der Geschäftslogik ist und kein Benutzer direkt mit ihr interagiert. Denn interagiert ein Benutzer über die XML-Schnittstelle mit dem Softwareprototyp, so geschieht dies über die Benutzeroberfläche eines fremden Systems, welches als separate Schicht auftritt. Da der Benutzer also nicht direkt mit dem Softwareprototyp interagiert, gehört die Schnittstelle definitionsgemäß nicht zu der Benutzeroberfläche.

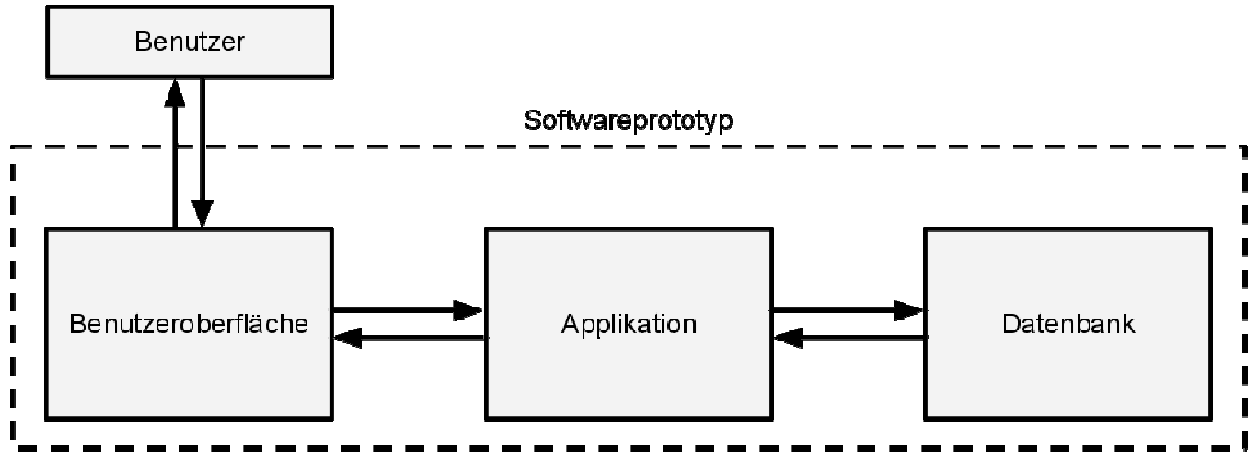

Abbildung 2: Darstellung der Interaktion zwischen Benutzern, Benutzeroberfläche, Applikation und Datenbank

3.7 Datenbanktabellen und Objekte

Die eingesetzte Datenbank verwaltet Informationen mithilfe von Tabellen. Diese Datenbanktabellen beinhalten eine Reihe von Datensätzen. Wird ein solcher Datensatz von der Applikation abgefragt, wird er gelesen und in ein Objekt verwandelt, bevor er innerhalb der Applikation weiter übergeben wird.

Diese Objekte haben die Felder der Datenbanktabelle als Attribute. Darüber hinaus werden sie jedoch durch die Applikation um weitere Attribute, Methoden und Unterklassen erweitert.

Ein Objekt ist immer eine Instanz einer Klasse, also ein Objekt im Sinne der objektorientierten Programmierung, das mit Daten aus einer Datenbanktabelle initialisiert wurde.

3.8 Entwicklungssystem vs. Produktionssystem

Als Produktionssystem werden eine Instanz oder mehrere Instanzen der Software im produktiven, meist kommerziellen Einsatz bezeichnet.[1] Hiervon abzugrenzen ist das Entwicklungssystem, welches eine rein zu Entwicklungszwecken betriebene, meist lokal beim Entwickler installierte, singuläre Instanz der Software bezeichnet.[2] Während beim Betrieb des Entwicklungssystems vor allem die schnelle Bereitstellung getätigter Softwareänderungen im Vordergrund steht, wird bei Produktionssystemen vor allem auf hohe Ausführungs- und Zugriffsgeschwindigkeiten Wert gelegt.[3] Häufig lässt sich Software daher sowohl im Produktions- als auch im Entwicklungsmodus starten, so dass die gleiche Codebasis für beide Systeme verwendet wird und die Systeme lediglich anders konfiguriert werden.[4]

1) Vgl. Henderson (2006), S. 48.

2) Vgl. Henderson (2006), S. 47.

3) Vgl. Tate/Carlson/Hibbs (2009), S. 127, sowie Mornini/Loy (2006), S. 25 f.

4) Vgl. Laurent/Dumbill (2009), S. 309, sowie Tate/Carlson/Hibbs (2009), S. 127 f.

4 Art des Softwareprototyps

Prinzipiell werden zwei Arten von Softwareprototypen unterschieden: Wegwerfprototypen und evolutionäre Prototypen.[1]

Wegwerfprototypen werden zu einem einmaligen, dedizierten Einsatz entwickelt und anschließend nicht wieder verwendet.[2]

Ein typisches Problem von Wegwerfprototypen ist, dass sie aufgrund ihrer temporären Natur sowohl hinsichtlich der Implementierung als auch hinsichtlich der Nutzung nicht mit dem endgültigen Produktionssystem zu vergleichen sind.[3]

Evolutionäre Prototypen werden schrittweise zur Anforderungsanalyse eingesetzt und mit dem Ziel entwickelt, nach der erstmaligen Verwendung mit den hierbei gewonnenen Erkenntnissen überarbeitet zu werden, bis die resultierende Software als Produktionssystem eingesetzt werden kann.[4]

Während in der betrieblichen Realität Wegwerfprototypen aus Zeit-, Kosten- und teilweise Sicherheitsgründen bevorzugt werden können,[5] liegt das Ziel des im Rahmen dieses Projektberichts zu implementierenden Softwareprototyps nicht in der Fertigstellung eines Produktionssystems oder kommerzialisierbaren Produkts, sondern in der Implementierung einer soliden softwaretechnischen Grundlage zur Unterstützung der weiteren Erhebung im Bereich der Anforderungen an eine Online-Frachtenbörse im Schienengüterverkehr. Der entwickelte Softwareprototyp kann so nicht nur einmalig aktuelle Forschungserkenntnisse umsetzen, sondern auch in zukünftigen Iterationen der Anforderungsanalyse aktualisiert verwendet werden.

Im vorliegenden Fall des Softwareprototyps einer Online-Frachtenbörse zur Anforderungsanalyse im Schienengüterverkehr erscheint die Implementierung eines evolutionären Softwareprototyps dem wissenschaftlichen Problem dieses Projektberichts angemessener als die Implementierung eines Wegwerfprototyps.[6]

---

1) Vgl. Pohl/Rupp (2009), S. 112, Berenbach et al. (2009), S. 249, sowie Sommerville (2007), S. 395, S. 409 sowie S. 412. Teilweise werden die Begriffe "evolutionärer Prototyp", "explorativer Prototyp" und "definitiver Prototyp" verwandt, um ein System zu beschreiben, das der Anforderungsanalyse dient und im Zeitverlauf sukzessive zu einem Produktionssystem weiterentwickelt werden soll, vgl. Bray (2002), S. 395, S. 396 sowie S. 398, Sommerville (2007), S. 65, S. 68 f. sowie S. 71. Dies wird teilweise unter dem Begriff "evolutionäre Entwicklung" zusammengefasst, vgl. Sommerville (2007), S. 68 f.

2) Vgl. Sommerville (2007), S. 409.

3) Der Grund hierfür ist, dass die Entwickler eines derartigen Systems nicht den gleichen Aufwand in einen Wegwerfprototypen investieren werden, den sie in ein vergleichbares Projekt investieren würden, das später von ihrem Unternehmen als Produktionssystem eingesetzt werden soll. Hierdurch kann es vorkommen, dass Wegwerfprototypen in den Bereichen Ausführungsgeschwindigkeit, Benutzeroberfläche und Benutzerführung nicht dem endgültigen System entsprechen. Hierdurch würden die mit ihrer Hilfe erhobenen Anforderungen verzerrt, vgl. Sommerville (2007), S. 412.

4) Vgl. Suh (2005), S. 83.

5) Vgl. Jones (2007), S. 371 sowie S. 401.

6) Der Grund hierfür ist die Plausibilitätsüberlegung, dass der vorliegende Projektbericht mit der Implementierung eines evolutionären, erweiterbaren Softwareprototyps nachhaltiger dazu beitragen kann, die in Kapitel 0 beschriebene Lücke zwischen dem Stand der Forschung und der betriebswirtschaftlich wünschenswerten Realität langfristig zu schließen, als dies bei Implementierung eines Wegwerfprototyps der Fall wäre. Diese Entscheidung steht im Gegensatz zu der in der Praxis geläufigen Grundregel, dass, wenn die bisher erhobenen Anforderungen ungenau definiert sind, bevorzugt ein Wegwerfprototyp zu entwickeln ist, vgl. Berenbach et al. (2009), S. 249.

5 Ziele des Softwareprototyps

5.1 Zielkonzeption

Das Primärziel des Softwareprototyps und dieses Projektberichts ist die Unterstützung der weiteren Anforderungsanalyse im Bereich Online-Frachtenbörsen im Schienengüterverkehr.

Ausgehend von diesem Primärziel können verschiedene Sekundärziele identifiziert werden.

Diese Ziele des Softwareprototyps werden in drei Bereichen dargestellt:

- systemische Ziele,
- Ziele des Frontends sowie
- Ziele des Backends.

Mit „systemischem Zielen" sind Ziele gemeint, die durch das System der Online-Frachtenbörse erreicht werden sollen.

Mit „Zielen des Frontends" sind Ziele gemeint, die durch die Gestaltung der Benutzeroberfläche erreicht werden sollen.

Mit „Zielen des Backends" sind analog Ziele gemeint, die nur mit dem für die Administratoren zugänglichen Teil der Systemkomponenten erreicht werden sollen.

Teilweise entstehen durch diese Aufteilung Probleme bei der Zuordnung einzelner Ziele. So ist das „Zusammenbringen von Angebot und Nachfrage" eine durch das System der Online-Frachtenbörse bereitgestellte Funktion und damit auch ein Ziel des Softwareprototyps. Der Softwareprototyp unterstützt dieses übergeordnete Ziel durch das systemische Ziel schlanker Prozessgestaltung, durch das Ziel des Frontends, den Benutzer nie zu überfordern, und durch das Ziel des Backends, möglichst gute Aussagen[1] über das Benutzerverhalten ableiten zu können.

5.2 Systemische Ziele

Der Softwareprototyp soll bestehende interne und externe Geschäftsprozesse unterstützen, ohne jedoch eine Änderung bestehender Geschäftsprozesse zu erfordern.[2] Dennoch sollen die Benutzer von der Software geführt werden, um einerseits ein bestimmtes, gewünschtes Verhalten innerhalb des Systems der Online-Frachtenbörse zu verstärken und andererseits im Rahmen der Anforderungsanalyse bestimmte Verhaltensweisen der Benutzer testen zu können.[3]

Die Anpassbarkeit des Softwareprototyps ist ein weiteres Ziel. Die in der Software abgebildeten Geschäftsprozesse werden möglichst schlank gehalten und die Geschäftslogik wird so einfach wie möglich strukturiert, damit sie später leicht erweiterbar ist.[4]

---

1) Dies bedeutet, dass die gewonnenen Erkenntnisse zu Aussagen führen sollen, die den drei Kriterien Objektivität, Reliabilität und Validität genügen.

2) Vgl. Klippert/Kowalski/Bruns (2010), S. 42.

3) So soll für die Betreiber der Online-Frachtenbörse nachvollziehbar sein, ob entwickelte Funktionen von den Benutzern auch genutzt werden.

4) Vgl. Klippert/Kowalski/Bruns (2010), S. 110.

Durch das Ziel der Anpassbarkeit erscheint die Implementierung von Schnittstellen zur einfachen Weiterentwicklung und Anbindung von Drittanbietersoftware sinnvoll.[1] Alle implementierten Schnittstellen sollen gut strukturiert sein, um eine spätere Anpassung der zugrundeliegenden Algorithmen zu ermöglichen. Dabei soll auch eine Schnittstelle implementiert werden, die eine Steuerung des Softwareprototyps durch Drittanbietersoftware und so die Integration in bestehende Planungs- und Steuerungssysteme ermöglicht.

5.3 Ziele des Frontends

Eine der Stärken des Einsatzes von Softwareprototypen liegt darin, dass sie interaktiv sind und dem Benutzer so ein Gefühl für die Funktionalitäten des geplanten Produkts geben.[2] Diese Tatsache soll das Frontend nutzen, indem es eine ansprechende Oberfläche bietet, die dem Benutzer möglichst wenig Vorstellungskraft in Bezug auf eine Online-Frachtenbörse im Schienengüterverkehr abverlangt, so dass er sich ganz auf das „Erlebnis", eine reale Software zu bedienen und zu testen, einlassen kann.

Aus diesem Grund muss das Frontend ein hohes Maß an Usability bieten.[3] Ein einfaches, durchdachtes Bedienkonzept bildet die Grundlage der Akzeptanz seitens der Benutzer.[4] Hierzu ist es im späteren Betrieb zur Anforderungsanalyse auch erforderlich, dass der Softwareprototyp eine angemessene Ausführungsgeschwindigkeit bietet.[5]

Ein weiteres Ziel besteht darin, den Benutzer möglichst nie zu überfordern.[6] So können zu komplexe Prozesse und Eingabemasken sowie eine zu aufwendige oder schlecht strukturierte Datenpräsentation zu Überforderung führen und die Akzeptanz seitens der Benutzer senken.[7]

5.4 Ziele des Backends

Das Backend muss die Betreiber der Online-Frachtenbörse in die Lage versetzen, über Berichtssysteme die Aktivitäten aller Benutzer zu verfolgen, die Entwicklung der Benutzerzahlen anzuzeigen und die Daten sowie Berechtigungen aller Benutzer zu ändern.[8] Ein weiteres Ziel des Backends ist es, den Administratoren ein hohes Maß an Usability zu bieten.[9] Die Online-Frachtenbörse muss durch die Administratoren über das Backend gewartet und kontrolliert werden können.[10]

---

1) Schnittstellen bieten einige Vorteile gegenüber fest implementierten Algorithmen.

2) Vgl. Bell (2005), S. 310.

3) Vgl. Klippert/Kowalski/Bruns (2010), S. 42 sowie S. 82.

4) Vgl. Klippert/Kowalski/Bruns (2010), S. 42 sowie S. 73.

5) Vgl. Bray (2002), S. 16 f. sowie S. 58.

6) Dieses Ziel ergibt sich aus folgender Plausibilitätsüberlegung: Eine komplexe Eingabemaske mag die Möglichkeiten der Datenanalyse durch den Softwareprototyp bereichern und korrekt ausgefüllt bessere Ergebnisse liefern als eine weniger komplexe Eingabemaske. Dies ist allerdings sowohl im Rahmen der Anforderungsanalyse als auch im Rahmen der betrieblichen Praxis dann irrelevant, wenn die komplexe Natur der Eingabemaske dazu führt, dass die betreffende Funktion von der Benutzern gemieden wird.

7) Vgl. Klippert/Kowalski/Bruns (2010), S. 57.

8) Vgl. Klippert/Kowalski/Bruns (2010), S. 83 f.

9) Vgl. Klippert/Kowalski/Bruns (2010), S. 82.

10) Vgl. Klippert/Kowalski/Bruns (2010), S. 84.

6 Arbeitstechniken zur Implementierung des Softwareprototyps

6.1 Auszuwählende Arbeitstechniken

Im Folgenden werden Arbeitstechniken ausgewählt, die zur Implementierung des Softwareprototyps eingesetzt werden sollen. Dabei sollen die Arbeitstechniken einigen strengen, noch zu identifizierenden Kriterien genügen, um die Implementierung des Softwareprototyps im Rahmen dieses Projektberichts zu ermöglichen.

Die ausgewählten Arbeitstechniken sind:

- Programmiersprache,
- Webframework,
- Datenbanksystem,
- Webserver sowie
- Versionsverwaltung.

Die zentrale Rolle nehmen die Auswahl der Programmiersprache und die hieraus resultierende Wahl des Webframeworks ein. Eine weniger endgültige Entscheidung stellt die Wahl der verwendeten Datenbank- und Webservertechnologie sowie der verwendeten Versionsverwaltung dar, da diese Technologien mit vertretbarem Aufwand in Zukunft angepasst werden könnten.[1] Es ist sogar davon auszugehen, dass diese Technologien im Laufe der Evolution des Softwareprototyps neu evaluiert werden müssen.[2]

Ebenfalls ausgewählt werden folgende Arbeitstechniken:[3]

- Browsertechnologien[4] sowie
- Entwicklungsumgebung[5].

Abbildung 3 stellt dar, welche Arbeitstechniken direkt und indirekt ausgewählt werden und die Wahl welcher Arbeitstechnik dabei die Wahl welcher anderen Arbeitstechnik beeinflusst.[6]

---

1) Der Grund hierfür ist, dass die Webserver- und Datenbanktechnologien lediglich unterstützende Technologien zum Betrieb des Softwareprototyps sind. Die Versionsverwaltung ist ebenfalls lediglich eine unterstützende Technologie zur Entwicklung und Verwaltung des Quelltextes des Softwareprototyps.

2) Der Grund hierfür ist, dass im Rahmen der weiteren Anforderungsanalyse Anforderungen an das System der Online-Frachtenbörse identifiziert werden können, die mit den gewählten Arbeitstechniken nicht oder nur unzureichend zu bewältigen sind.

3) Die Wahl dieser Arbeitstechniken geschieht indirekt über die Wahl der zuvor genannten Arbeitstechniken.

4) Mit dem Begriff "Browsertechnologien" werden im Folgenden jene Beschreibungs- und Skriptsprachen zusammengefasst, die lokal auf dem Rechner des Benutzers ihren Einsatz finden, um die Webseite darzustellen.

5) Gemeint ist hier die zur Entwicklung eingesetzte Betriebssystem/Quelltexteditor-Kombination.

6) Zur Reihenfolge der Auswahl vgl. S. 27.

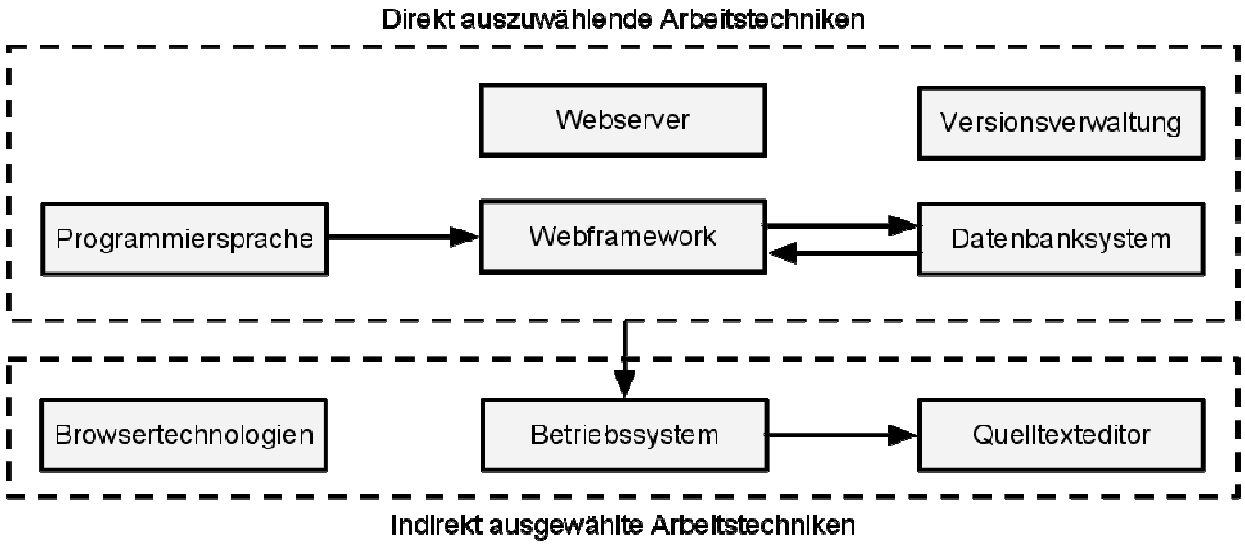

Abbildung 3: Darstellung der direkt und indirekt auszuwählenden Arbeitstechniken

Zu den Browsertechnologien HTML, CSS und JavaScript gibt es im Internet keine wirtschaftlich einsetzbaren Alternativen.[1]

Bei der Wahl der Entwicklungsumgebung kann es theoretisch, je nach Programmiersprache, Webframework, Datenbanksystem, Webserver und Versionsverwaltung zu Einschränkungen kommen, wenn nur einzelne Betriebssysteme oder Betriebssystemfamilien[2] von der gewählten Technologiekombination unterstützt werden.[3]

Wie gezeigt wurde, existiert bei den Browsertechnologien und der Entwicklungsumgebung keine direkte Auswahlproblematik. Die Wahl dieser Arbeitstechniken ergibt sich indirekt aus der Wahl der direkt ausgewählten Arbeitstechniken. Ihre Auflistung dient lediglich einer vollständigeren Darstellung der eingesetzten Arbeitstechniken.

6.2 Kriterien zur Auswahl der Arbeitstechniken

Da zur Implementierung des Softwareprototyps lediglich der finanzielle, zeitliche und personelle Rahmen eines Projektberichts zur Verfügung steht, empfiehlt es sich bei der Implementierung auf Technologien zurückzugreifen, die eine schnelle, praxis- und problemorientierte Implementierung geplanter Funktionalitäten unterstützen.

Wichtige Eigenschaften der zu evaluierenden Technologien sind daher:

---

1) Anstatt HTML könnte theoretisch auch ein beliebiges XML-Format verwendet werden, das per CSS darstellbar gemacht wird, vgl. Rothfuss/Ried (2003), S. 82 f. Anstatt CSS könnten Attribute nach dem Standard "HTML 4" verwendet werden, um die Farbe und Form einzelner Elemente zu beeinflussen, allerdings würde diese Vorgehensweise wesentlich größere HTML-Dateien verursachen und die Flexibilität stark einschränken. Zu JavaScript gäbe es zumindest theoretisch die Alternative VBScript einzusetzen, die allerdings nur von Browsern des Herstellers Microsoft interpretiert und von Microsoft auch nicht mehr weiterentwickelt wird (vgl. Lippert (2004), o.S.) und so den Anwenderkreis der Benutzeroberfläche originär stark einschränken würde.

2) Gemeint sind hier bspw. Microsoft Windows, Linux oder Apple Mac OS X.

3) Umgekehrt wird es durch die Wahl entsprechender Technologien möglich, während der Implementierung des Softwareprototyps simultan auf drei verschiedenen Betriebssystemen mit drei verschiedenen Quelltexteditoren zu arbeiten und dabei niemals durch die am jeweiligen Ort verfügbare technische Ausstattung eingeschränkt zu sein.

- geringe Kosten,[1]
- geringe Entwicklungszeit[2] sowie
- hohe Flexibilität[3].

Als weniger wichtig werden hingegen folgende Eigenschaften erachtet:

- hohe Ausführungsgeschwindigkeit des Softwareprototyps[4],
- hohe Sicherheit[5] sowie
- Qualifikation, Verfügbarkeit und Kosten des zur Weiterentwicklung des Softwareprototyps nötigen Fachpersonals[6].

Die aufgeführten Eigenschaften bilden die Kriterien zur Auswahl der Arbeitstechniken zur Implementierung des Softwareprototyps.

Abbildung 4 stellt dar, bezüglich welcher Kriterien sich die auszuwählenden Arbeitstechniken primär bewähren müssen, um eine schnelle, praxis- und problemorientierte Implementierung des Softwareprototyps zu gewährleisten.

1) Alle im Folgenden beschriebenen Technologien sind "Open Source". Sie sind somit quelloffen verfügbar und entgeltfrei nutzbar. Die Lizenzen der hier beschriebenen Technologien erlauben zudem explizit die kommerzielle Nutzung.

2) Eine geringe Entwicklungszeit wird immer dann begünstigt, wenn zur Implementierung geplanter Funktionalitäten möglichst wenig Vorarbeit, bspw. in Form von Installation und Konfiguration zusätzlicher Komponenten, nötig ist. Neue Ideen sollten im Idealfall direkt implementiert und zusätzliche Funktionen so schnell auf ihre Praxisrelevanz getestet werden können.

3) Mit Flexibilität ist vor allem gemeint, dass keine strengen Paradigmen existieren sollen, die den Entwickler bei der Verwendung der Technologie einschränken. Der Entwickler sollte möglichst frei in seinen Entscheidungen sein, bspw. auf welchem Betriebssystem und mit welchem Quelltexteditor entwickelt wird.

4) Für einen Softwareprototyp ist die Ausführungsgeschwindigkeit zunächst sekundär, da wenige Benutzer den Softwareprototyp gleichzeitig nutzen werden, da er lediglich der weiteren Anforderungsanalyse dient und nicht als Produktionssystem eingesetzt wird.

5) Fragen bzgl. der Systemsicherheit, Typsicherheit oder Threadsicherheit fließen nicht in die Bewertung ein, da der Softwareprototyp nicht als Produktionssystem eingesetzt wird.

6) Insbesondere Plausibilitätsüberlegungen, wie bspw. die, dass das Arbeitskräfteangebot bei Open-Source-Technologien (aufgrund des offenen Zugangs) tendenziell größer sein dürfte als bei geschlossenen Technologien oder dass Entwickler, die sich geschlossener Technologien bedienen, aufgrund strukturierter Ausbildung tendenziell höher qualifiziert sein dürften, fließen nicht in die Bewertung ein.

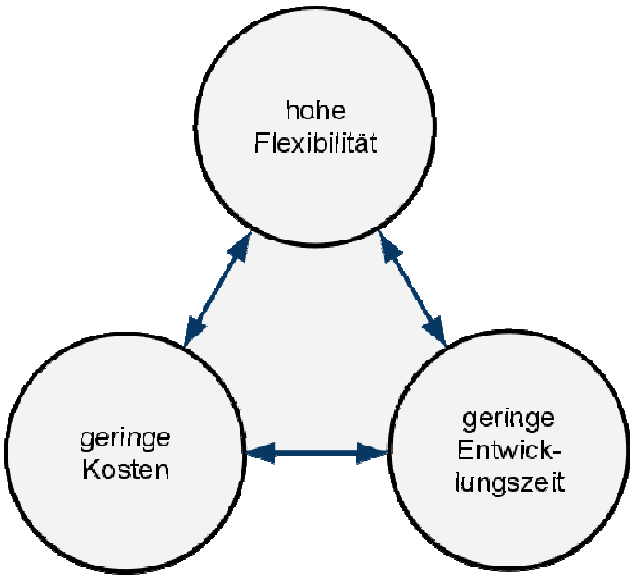

Abbildung 4: Darstellung der primären Kriterien zur Auswahl der Arbeitstechniken

6.3 Beschreibung der Arbeitstechniken

6.3.1 Programmiersprache

Es muss eine Programmiersprache gewählt werden, die den zuvor genannten Kriterien besonders streng genügt, da ihre Wahl bereits Auswirkungen auf die Eigenschaften des späteren Softwareprototyps haben kann.[1]

Folgende Programmiersprachen stehen zur Auswahl:

6.3.1.1 C#

C# ist eine Programmiersprache zum strukturierten, komponentenbasierten und objektorientierten Entwickeln großer Applikationen.[2] Das Ziel von C# ist es, eine einfache, sichere, hoch performante Programmiersprache zu sein.[3] Die Konzeption und Syntax von C# lehnt sich an Programmiersprachen wie C++ und Java an.[4]

C# ist über das Internet kostenlos verfügbar.[5]

1) So könnte die Wahl einer plattformabhängigen Programmiersprache dazu führen, dass der Softwareprototyp bspw. nur unter Windows-Betriebssystemen mit vertretbarem Aufwand weiterentwickelt werden könnte.

2) Vgl. Liberty/MacDonald (2009), S. 5.

3) Vgl. Liberty/MacDonald (2009), S. 5.

4) Vgl. Liberty/MacDonald (2009), S. 2 sowie S. 5, Nash (2010), S. 17, sowie Niemeyer/Knudsen (2005), S. 4.

5) Alle nötigen Informationen sind unter der URL http://www.microsoft.com/express/ verfügbar.

Mit C# lassen sich Applikationen kompilieren,[1] allerdings wird hierbei kein nativ von der Prozessor-Architektur ausführbarer Maschinencode erzeugt.[2] Vielmehr übersetzt der Compiler den in C# verfassten Quelltext zunächst in eine intermediäre Sprache, namens Microsoft Intermediate Language (MSIL), und speichert diese Informationen zusammen mit anderen nötigen Daten in eine ausführbare Datei.[3] Wird diese Datei ausgeführt, liest der .NET-JIT-Compiler den MSIL-Code, übersetzt ihn in Maschinencode und erzeugt hieraus zur Laufzeit eine ausführbare Applikation im Arbeitsspeicher.[4]

C# ist als Programmiersprache Teil der .NET-Plattform.[5] Über die .NET-Plattform hat C# somit Zugriff auf umfangreiche Klassenbibliotheken.[6]

C# eignet sich zur Entwicklung komplexer Webapplikationen, da es als objektorientierte Sprache für den Einsatz in Microsofts .NET-Plattform konzipiert wurde.[7]

6.3.1.2 Java

Java ist eine plattformunabhängige, schnelle und sichere Programmiersprache.[8] Das Ziel von Java ist, eine einfache Programmiersprache zu sein, auf deren Grundlage komplexere Konstrukte entwickelt werden können.[9] Die Syntax von Java lehnt sich an die Programmiersprache C an.[10] Java ist dabei sowohl im Design als auch in der Implementierung sicher.[11]

Java ist über das Internet kostenlos verfügbar.[12]

Mit Java lassen sich Applikationen kompilieren, allerdings wird hierbei kein nativ von der Prozessorarchitektur ausführbarer Maschinencode erzeugt.[13] Der Java-Compiler erzeugt aus dem Quelltext

---

1) Kompilierbare Programmiersprachen sind nicht dynamisch, aber die resultierenden Applikationen sind häufig schneller und sicherer in der Ausführung als eine analog in einer Skriptsprache geschriebene Applikation, vgl. Niemeyer/Knudsen (2005), S. 8. Skriptsprachen sind dagegen dynamisch und mächtige Werkzeuge zur schnellen Entwicklung von Softwareprototypen, vgl. Niemeyer/Knudsen (2005), S. 8.

2) Vgl. Liberty/MacDonald (2009), S. 10, sowie Solis (2008), S. 7 ff.

3) Vgl. Liberty/MacDonald (2009), S. 10, Solis (2008), S. 7 ff., sowie Nash (2010), S.10 ff.

4) Vgl. Liberty/MacDonald (2009), S. 10, Solis (2008), S. 7 ff., sowie Nash (2010), S.10 ff.

5) Die .NET-Plattform ist eine von dem Unternehmen Microsoft entwickelte Plattform zur Enwicklung und Ausführung von Applikationen. .NET eignet sich zum Entwickeln von Windows- und Webapplikationen sowie Webservices, vgl. Liberty/MacDonald (2009), S. 5 sowie S. 17. Die Plattform unterstützt mehrere Sprachen und steht im vollen Umfang nur für Windows zur Verfügung, wenngleich Open-Source-Compiler für C# auch für unixoide Systeme existieren, bspw. das Mono-Projekt oder #develop, vgl. Liberty/MacDonald (2009), S. 16. Die Sprachneutralität wird durch die CLR-Komponente bereitgestellt, vgl. Liberty/MacDonald (2009), S. 17, Solis (2008), S. 5, sowie Nash (2010), S. 15. Die CLR-Komponente ist neben der Code-Ausführung auch für das Speichermanagement, die Sicherheitsverifikation und die automatische Speicherbereinigung zuständig, vgl. Solis (2008) S. 3 f. sowie S. 6.

6) Vgl. Liberty/MacDonald (2009), S. 17.

7) Vgl. Liberty/MacDonald (2009), S. 5.

8) Vgl. Niemeyer/Knudsen (2005), S. 1.

9) Vgl. Niemeyer/Knudsen (2005), S. 10.

10) Vgl. Niemeyer/Knudsen (2005), S. 83.

11) Vgl. Niemeyer/Knudsen (2005), S. 9 ff. sowie S. 15 ff.

12) Alle nötigen Informationen sind unter der URL http://www.java.com/ verfügbar.

13) Vgl. Niemeyer/Knudsen (2005), S. 4 f.

zuerst einen Bytecode, der anschließend von einer Java Virtual Machine (JVM) ausgeführt werden kann.[1] Die Ausführungsgeschwindigkeit dieses Bytecodes ist aufgrund eines JIT-Compilers mit der Ausführungsgeschwindigkeit nativ kompilierter Applikationen zu vergleichen.[2]

Java eignet sich zur Entwicklung komplexer Webapplikationen, da Java-Applikationen eine höhere Ausführungsgeschwindigkeit haben als Applikationen, die in Skriptsprachen geschrieben sind.[3] Durch Javas Platformunabhängigkeit können Webapplikationen schnell und einfach auf andere Hardwarekonfigurationen migriert werden. Zudem existieren weitere Technologien wie Java Server Pages, Java Server Faces und XML, die das Entwickeln komplexer Webapplikationen zusätzlich vereinfachen und unterstützen.[4]

6.3.1.3 PHP

PHP ist eine laufzeitinterpretierte, objektorientierte Programmiersprache, die originär zur Webentwicklung implementiert wurde. PHP ist dabei plattformunabhängig, schnell und flexibel.[5] Zudem ist PHP stark von Perl und C beeinflusst.[6]

PHP ist über das Internet kostenlos verfügbar.[7]

PHP zeichnet sich durch eine hohe Ausführungsgeschwindigkeit, eine breite Datenbankunterstützung und die Verfügbarkeit zahlreicher Funktionsbibliotheken sowie durch eine gute Internetprotokolleinbindung aus.[8]

Theoretisch ist PHP durch die originäre Unterstützung von Datenbanken und Internetprotokollen gut geeignet, den Softwareprototyp einer Webapplikation zu implementieren.

6.3.1.4 Ruby

Ruby ist eine laufzeitinterpretierte, objektorientierte Programmiersprache, die Funktionalitäten zur Metaprogrammierung und damit eine gute Grundlage zur Programmierung eigener DSL[9] bietet. Ruby lehnt sich an Konzepte von Smalltalk und Perl an, wurde aber so konzipiert, dass es Entwicklern, die mit C und Java vertraut sind, leicht fällt, Ruby zu erlernen.[10]

Ruby ist über das Internet kostenlos verfügbar.[11]

---

1) Vgl. Niemeyer/Knudsen (2005), S. 4 f.

2) Beispiele für JIT-Compiler sind die JVM von Java und die CLR von .NET. Zum JIT-Compiler von .NET.

3) Vgl. Niemeyer/Knudsen (2005), S. 8.

4) Vgl. Niemeyer/Knudsen (2005), S. 502 f.

5) Vgl. Lerdorf/Tatroe/MacIntyre (2006), S. 1 f., sowie Hudson (2006), S. 1 ff.

6) Vgl. Lerdorf/Tatroe/MacIntyre (2006), S. 18.

7) Alle nötigen Informationen sind unter der URL http://www.php.net/ verfügbar.

8) Vgl. Lerdorf/Tatroe/MacIntyre (2006), S. 2, sowie Hudson (2006), S. 2 ff.

9) Eine DSL ist eine domänenspezifische Sprache für ein spezielles Problemfeld, vgl. Carneiro/Barazi (2010), S. 4.

10) Vgl. Flanagan/Matsumoto (2008) S. 2.

11) Alle nötigen Informationen sind unter der URL http://ruby-lang.org/ verfügbar.

Ruby ist wie PHP eine Skriptsprache, die während der Laufzeit interpretiert wird und, im Gegensatz zu C# und Java, zur Ausführung nicht kompiliert werden muss. Allein dieser Faktor beschleunigt die Entwicklungszeit[1] zulasten der Ausführungsgeschwindigkeit.[2]

Ruby ist komplett objektorientiert - jeder Wert ist ein Objekt. Dennoch eignet sich Ruby auch für prozedurale und funktionale Programmierstile.[3]

Die Programmiersprache Ruby eignet sich besonders gut zur schnellen Implementierung von Prototypen, da sie als Sprache konzipiert ist, die es dem Programmierer erleichtern soll, sich auf sein Problem zu konzentrieren, ohne dabei permanent auf die Besonderheiten der Programmiersprache achten zu müssen.[4]

6.3.2 Webframework

Es muss ein Webframework ausgewählt werden, welches die grundlegenden Funktionalitäten der Client/Server-Kommunikation bereitstellt und dabei den genannten Kriterien ebenso genügt wie die zuvor ausgewählte Programmiersprache. Die beschriebenen Frameworks kreieren hierbei eine DSL für die Domäne „Entwicklung einer Webapplikation".

Ausgehend von der Wahl von Ruby als Programmiersprache werden im Folgenden ausschließlich Webframeworks betrachtet, die in dieser Sprache implementiert sind.[5]

Zudem erscheint es sinnvoll, nur etablierte, mächtige Webframeworks zu evaluieren.[6]

Folgende Webframeworks stehen zur Auswahl:

6.3.2.1 Ruby on Rails

Ruby on Rails wurde von dem Dänen David Heinemeier Hansson als Nebenprodukt seiner Arbeit an der Projektmanagementsoftware „Basecamp" entwickelt.[7]

Hansson entschied sich, Ruby anstatt Java oder PHP zu verwenden, und es gelang ihm, das Projekt „basecamp" in nur zwei Monaten zu realisieren, da Ruby ihm das Programmieren so stark vereinfachte, dass er sich nur auf die Aspekte seines Problems anstatt auf die Aspekte der Programmiersprache konzentrieren musste.[8] Er entschloss sich daraufhin, die grundlegende Architektur des Pro-

---

1) Der Grund hierfür ist, dass der Quelltext nicht nach jeder kleinen Änderung (bspw. der Berichtigung eines Tippfehlers im Quelltext) neu kompiliert werden muss, was je nach Ausstattung des Rechners, der Komplexität des Projekts und der Programmiersprache schnell Minuten in Anspruch nehmen kann.

2) Vgl. Carlson/Richardson (2006), S. 817.

3) Selbst primitive Typen, die boolschen Werte "true" und "false" sowie das die Abwesenheit eines Wertes repräsentierende "nil" sind in Ruby Objekte, vgl. Flanagan/Matsumoto (2008) S. 2.

4) Der Schöpfer von Ruby sagt, er habe stets sich selbst als Zielgruppe vor Augen gehabt: einen einfachen Programmierer, der eine Sprache nutzen möchte, die ihm die Arbeit erleichtert ("Ruby is designed to make programmers happy."), vgl. Flanagan/Matsumoto (2008), S. 2.

5) Zu den Gründen der Wahl von Ruby vgl. Kapitel S. 28. Hierdurch nicht betrachtete Alternativen in anderen Sprachen wären bspw. ASP.NET MVC für C#, Struts für Java und CakePHP für PHP, vgl. Ammelburger/Scherer (2008), S. 6.

6) Als "mächtige Webframeworks" konnten die im Folgenden beschriebenen Frameworks "Merb", "Sinatra" und "Ruby on Rails" identifiziert werden, vgl. Berube (2008), S. 180.

7) Vgl. Carneiro/Barazi (2010), S. 3, Cooper (2009), S. 104 sowie Williams (2007), S. 7.

8) Vgl. Cooper (2009), S. 104 sowie Williams (2007), S. 7.

jekts in ein neues Webframework namens „Ruby on Rails" zu adaptieren, welches die Entwicklung von datenbankgestützten Webapplikationen vereinfachen sollte.[1]

Ruby on Rails folgt dem MVC-Modell, es abstrahiert die Verwendung der Datenbanktechnologie, des JavaScript-Frameworks, der Session-Verwaltung, stellt einen Migrationsansatz zur Population der Datenbank zur Verfügung und übernimmt die Verwaltung und Implementierung vieler wiederkehrender Abläufe für den Entwickler.[2]

Ruby on Rails hat über die Jahre Entwickler aus den Bereichen PHP, Java und C# durch seine einfachere Art der Webentwicklung überzeugt.[3] Primär liegt dies am MVC-Modell, den vorsichtig gewählten Voreinstellungen und der Programmiersprache Ruby.[4]

Merb

Merb ist ein kleines, leichtgewichtiges Webframework.[5] Merb wurde mit dem Ziel entwickelt, ein Komplement zu Ruby on Rails zu werden, aber im Gegensatz zu diesem sollte Merb im Kern vorsätzlich minimal ausgestattet sein.[6]

Merb ist über das Internet kostenlos verfügbar.[7]

Viele Funktionalitäten eines vollwertigen Webframeworks sind für Merb als Plugins verfügbar, jedoch nicht im Kernpaket enthalten, um dieses sowohl schlank als auch technologieagnostisch[8] zu halten. So übernimmt Merb bewusst nicht die Namenskonventionen von Ruby on Rails, was es erschwert, möglichst schnell funktionsfähige Prototypen zu entwickeln, jedoch den Vorteil bietet, dass der Entwickler mehr direkten Einfluss auf die Funktionsweise seiner Software hat.[9]

Merb eignet sich gut, um komplexe Webapplikationen zu entwickeln, die ein hohes Maß an Anpassungsfähigkeit erfordern. Die Flexibilität von Merb ist sehr hoch, da das Kernpaket keinerlei Vorgaben umfasst welche Technologie in welchem Bereich einzusetzen ist.[10] Allerdings leidet die Entwicklungszeit unter dieser Flexibilität, was die Eignung zum schnellen Entwickeln eines Softwareprotoyps einschränkt.

---

1) Vgl. Cooper (2009), S. 104 sowie Williams (2007), S. 7.

2) Die Reduktion der Wiederholung wiederkehrender Abläufe wird primär durch die Paradigmen "convention over configuration" und "don’t repeat yourself" gewährleistet, vgl. Chak (2009), S. 239, Ediger (2008), S. 1 sowie S. 2 f., sowie Williams (2007), S. 7.

3) Vgl. Carneiro/Barazi (2010), S. 4.

4) Bei den Voreinstellungen ist vor allem das Paradigma "convention over configuration" zu nennen, vgl. Ediger (2008), S. 1.

5) Merb steht für "Mongrel plus ERb", vgl. Laurent/Dumbill (2009), S. 325. Mongrel ist ein Webserver, vgl. S. 22. ERb eine Templatesprache des Ruby-Pakets, vgl. Carlson/Richardson (2006), S. 8 ff.

6) Vgl. Marshall/Pytel/Yurek (2007), S. 195, sowie Laurent/Dumbill (2009), S. 325 f.

7) Alle nötigen Informationen sind unter der URL http://merbivore.org/ verfügbar.

8) Technologieagnostisch bedeutet hier, dass es mit Merb nicht nur irrelevant ist, welches Datenbanksystem man als Entwickler einsetzt, sondern auch die die Datenbank abstrahierende Schnittstelle (der sogenannte ORM-Layer) kann gewechselt werden, vgl. Cooper (2009), S. 350 f.

9) Vgl. Laurent/Dumbill (2009), S. 325 f.

10) Vgl. Cooper (2009), S. 350 f.

Das Kernteam von Merb entschloss sich nach der Veröffentlichung der Version 1.0.0, sich dem Kernteam von Ruby on Rails anzuschließen und für die mittlerweile veröffentlichte Version Ruby on Rails 3.0 die Grundgedanken von Merb in den Kern von Ruby on Rails zu integrieren.[1]

Daher ist die Zukunft von Merb als eigenständiger Technologie ungewiss.[2] Das Projekt „Ramaze"[3] gilt als möglicher Nachfolger von Merb, da es auf den gleichen Idealen und Grundideen aufbaut.[4] Gleichzeitig versucht Ramaze jedoch, die Simplizität von Sinatra mit der komplexen MVC-Architektur von Ruby on Rails und Merb zu verbinden.[5] Aufgrund der vergleichsweise geringen Dokumentation, der jungen Entwicklungsgeschichte und der starken Ähnlichkeit zu Sinatra, Merb und Ruby on Rails wird Ramaze im Rahmen dieses Projektberichts nicht weiter evaluiert.

6.3.2.2 Sinatra

Sinatra ist ein leichtgewichtiges Webframework, das HTTP-Funktionen für existierende Applikationen bereitstellt und es erlaubt, von Grund auf neue Applikationen so einfach wie möglich zu entwickeln.[6]

Daher wird Sinatra von seinen Entwicklern auch häufig nicht als Webframework im klassischen Sinn beschrieben, sondern als „DSL zum Erstellen von Webapplikationen" bezeichnet.[7]

Sinatra ist über das Internet kostenlos verfügbar.[8]

Im Gegensatz zu Ruby on Rails forciert Sinatra keine Konzepte, wie bspw. MVC oder REST.[9]

6.3.3 Datenbanksystem

Ein Datenbanksystem ist nötig, damit der Softwareprototyp Daten bei Bedarf persistent speichern und wieder abfragen kann.[10] Moderne Webframeworks wie Ruby on Rails abstrahieren dabei den Datenbankzugriff und schaffen eine Schnittstelle zur Speicherung der Daten, welche die Wahl des Datenbanksystems zumindest theoretisch relativiert (da das Webframework mit allen gängigen Datenbanksystemen interagieren kann).[11]

Dennoch lohnt sich die Analyse einiger Alternativen in Bezug auf ihre Erfüllung der genannten Kriterien.

Folgende Datenbanksysteme stehen zur Auswahl:

---

1) Vgl. Carneiro/Barazi (2010), S. 4, Cooper (2009), S. 350, Katz (2008), o.S., sowie Hansson (2008), o.S.

2) Vgl. Cooper (2009), S. 350 f.

3) Ramaze ist über das Internet kostenlos verfügbar. Alle nötigen Informationen sind unter der URL http://ramaze.net/ verfügbar.

4) Vgl. Cooper (2009), S. 351.

5) Vgl. Cooper (2009), S. 397.

6) Vgl. Cooper (2009), S. 387, sowie Watson (2009), S. 133.

7) Vgl. Cooper (2009), S. 387, sowie Watson (2009), S. 133.

8) Alle nötigen Informationen sind unter der URL http://www.sinatrarb.com/ verfügbar.

9) Vgl. Cooper (2009), S. 388.

10) Für ein einfaches Beispiel, warum Datenbanksysteme für Webapplikationen wichtig sind, vgl. Reese et al. (2002), S. 10.

11) Vgl. Carneiro/Barazi (2010), S. 3.

6.3.3.1 MySQL

MySQL ist ein relationales Datenbankmanagementsystem (RDBMS).[1] Das Ziel von MySQL ist die strukturierte Speicherung von großen Datenmengen in Tabellen. MySQL ist plattformunabhängig, schnell, sicher und stabil.[2]

MySQL ist über das Internet kostenlos verfügbar.[3]

MySQL wird seit über 10 Jahren weiterentwickelt, ausgiebig getestet und dokumentiert.[4] MySQL hat sich zu einer der führenden Systeme im Bereich der RDBMS entwickelt.[5]

Das Konzept von MySQL basiert auf einer Client/Server-Architektur.[6] MySQL besitzt einen dedizierten Datenbankserver, der von Clients angesteuert werden kann, bspw. um Daten zu manipulieren und abzufragen.[7] Zudem können mehrere MySQL-Server distribuiert betrieben werden, um so die Sicherheit der Daten oder die Ausführungsgeschwindigkeit der Abfragen durch Redundanz zu steigern.[8]

Während MySQL nicht immer die beste Wahl als Datenbanktechnologie sein muss, spricht für die Verwendung die breite Anwendbarkeit, da es sowohl eine Reihe an Datentypen unterstützt als auch als hochgradig redundantes System aufgesetzt werden und auf einer breiten Palette an Hardwarekonfigurationen eingesetzt werden kann.[9] Insbesondere die letzten beiden Aspekte sprechen für den Einsatz im Bereich von Webapplikationen.[10]

6.3.3.2 SQLite

SQLite ist, wie auch MySQL, ein relationales Datenbankmanagementsystem (RDBMS).[11] Das Ziel von SQLite ist es, die Konfiguration und den Betrieb eines dedizierten Datenbankservers überflüssig zu machen und das Datenbanksystem stattdessen in die Applikation, die es benutzt, zu integrie-

---

1) RDBMS bedeutet "relational database management system". Es beschreibt ein System von Applikationen zur Verwaltung relationaler Datenbanken. Ziel eines RDBMS ist die sichere Speicherung von Daten und das Interpretieren von Kommandos zur Analyse, Manipulation, Sortierung und Selektierung bestehender und neuer Datensätze. Beispiele für solche Systeme sind MySQL, Oracle und Microsoft SQL Server, vgl. Kofler/Kramer (2005), S. 3 f.

2) Vgl. Kofler/Kramer (2005), S. 7 sowie S. 16.

3) Alle nötigen Informationen sind unter der URL http://mysql.com/ verfügbar.

4) Vgl. Kofler/Kramer (2005), S. 16.

5) Folgende Quellen behaupten, MySQL sei die meistverwendete Open-Source-Datenbanktechnologie: Morsy/Otto (2008), S. 429, sowie Kofler/Kramer (2005), S. 16. Generell sind solche Aussagen jedoch schwierig zu verifizieren oder zu falsifizieren, wie das Beispiel SQLite zeigt, vgl. Kreibich (2010), S. 15 f.

6) Vgl. Kofler/Kramer (2005), S. 4.

7) Vgl. Kofler/Kramer (2005), S. 4 f.

8) Vgl. Kofler/Kramer (2005), S. 6. Zur Architektur von Client/Server- sowie distribuierten MySQL-Systemen vgl. Reese et al. (2002), S. 137 ff.

9) Vgl. Schwartz et al. (2008), S. 1.

10) Vgl. Schwartz et al. (2008), S. 1.

11) Vgl. Kreibich (2010), S. 1 ff., sowie Owens (2006), S. 1.

ren.[1] Zu diesem Zweck ist SQLite plattformunabhängig, kompakt, verlässlich und muss vor der Inbetriebnahme nicht konfiguriert werden.[2]

SQLite ist über das Internet kostenlos verfügbar.[3]

Auf der Webseite von SQLite wird behauptet, SQLite sei das Datenbanksystem mit den meisten Installationen weltweit. Diese Aussage darf als mutig gelten, wenngleich sie nicht von vorneherein verworfen werden kann. Aufgrund der Tatsache, dass keinerlei Lizenzabkommen zur Nutzung von SQLite nötig sind, kann nicht genau erhoben werden, wieviele installierte SQLite-Versionen es weltweit gibt.[4]

Eine Stärke von SQLite liegt in der hohen Flexibilität: Der Entwickler muss keinen Datenbankserver konfigurieren, sich keine Sorgen über Netzwerkkonnektivität machen, keine Lizenzabkommen beachten oder Gebühren entrichten.[5] Bei der Ausführungsgeschwindigkeit kann SQLite mit anderen RDBMS mithalten, außer bei sehr komplexen Abfragen („Queries") auf großen Datenmengen.[6]

Aufgrund der im Vergleich zu anderen RDBMS einfachen Installation und entfallenden Konfiguration ist SQLite gut geeignet, die Entwicklungszeit einer Webapplikation zu verkürzen. Auch können die erwähnten Nachteile von SQLite im Rahmen eines Entwicklungssystems vernachlässigt werden.[7] Es ist allerdings zweifelhaft, ob ein späteres Produktionssystem mit SQLite zu betreiben wäre.[8]

6.3.3.3 CouchDB

CouchDB ist ein noch junges, dokumentorientiertes Datenbanksystem. Das Ziel von CouchDB ist, den Anforderungen moderner, webbasierter, dokumentorientierter und distribuierter Applikationen Rechnung zu tragen. Da CouchDB distribuiert betrieben werden kann und in Erlang implementiert ist, ist es ein hochgradig verlässlich verfügbares, redundantes und skalierbares Datenbanksystem. CouchDB ist zurzeit nur für unixoide Systeme verfügbar; eine inoffizielle Implementierung für Windows befindet sich in der Entwicklung.[9]

CouchDB ist über das Internet kostenlos verfügbar.[10]

Im Gegensatz zu RDBMS ist CouchDB nicht schemabasiert, sondern dokumentorientiert.[11] Dies bedeutet grundsätzlich, dass CouchDB Daten nicht in Tabellen, Zeilen und Spalten speichert, son-

---

1) Vgl. Kreibich (2010), S. 1 ff., sowie Owens (2006), S. 1.

2) Vgl. Kreibich (2010), S. 1 ff., sowie Owens (2006), S. 1.

3) Alle nötigen Informationen sind unter der URL http://sqlite.org/ verfügbar.

4) Zur weiteren Diskussion der Installationsbreite, vgl. Kreibich (2010), S. 15 f.

5) Vgl. Owens (2006), S. 9.

6) Vgl. Owens (2006), S. 11.

7) Vgl. Kreibich (2010), S. 11 f., sowie Kofler/Kramer (2005), S. 7.

8) Für den produktiven Einsatz wäre ein RDBMS mit dediziertem Datenbankserver, wie bspw. MySQL, besser geeignet, vgl. Tate/Carlson/Hibbs (2009), S. 153.

9) Vgl. Lennon (2009), S. 3 f., sowie Watson (2009), S. 255 sowie S. 260.

10) Alle nötigen Informationen sind unter der URL http://couchdb.apache.org/ verfügbar.

11) Vgl. Lennon (2009), S. 3, sowie Watson (2009), S. 255.

dern in Form eigenständiger, voneinander unabhängiger Dokumente.[1] Eine schemafreie Datenbank hat den Vorteil, dass der Entwickler sich nicht bereits vor der Entwicklung Gedanken zum Aufbau des Tabellenschemas und der damit verbundenen Interdependenzen und Integritätsbedingungen machen muss. Somit begünstigt die Schemafreiheit die Entwicklungszeit einer datenbankbasierten Applikation erheblich. Nachteilig an dieser Konzeption ist aber, dass sie sich in hoch komplexe Applikationen, deren Geschäftslogik auf die Existenz eindeutig definierter Interdepenzen und eines konstanten Integritätsniveaus angewiesen ist, negativ auswirken kann, da die gespeicherten Daten hochgradig unstrukturiert sind.[2] Für diese Fälle ist ein dokumentorientiertes Datenbanksystem weniger geeignet.[3]

Ein weiterer Unterschied zu bspw. SQL-Datenbanksystemen ist, dass CouchDB Revisionen der gespeicherten Dokumente archiviert (ähnlich einer Versionsverwaltung, nur dass das Datenbanksystem selbst die Versionierung vornimmt).[4] Im Gegensatz zu SQL-Datenbanksystemen werden Reports nicht mithilfe von SQL-Queries, sondern mithilfe von JavaScript-Funktionen erstellt, die über die in der Datenbank gespeicherten JSON-Dokumente iterieren.[5] Zur Kommunikation mit dem Datenbankserver stellt CouchDB eine HTTP-API bereit.[6] Dies hat den Vorteil, dass jede Plattform, die HTTP-Requests ausführen und JSON interpretieren kann, in der Lage ist, CouchDB als Datenbanksystem einzusetzen.[7]

Nicht zuletzt aufgrund der Implementierung in Erlang, der HTTP-API und des Einsatzes von JSON ist CouchDB gut geeignet, im Rahmen einer komplexen Webapplikation, die große Mengen unstrukturierter Daten handhaben muss, als Datenbanksystem eingesetzt zu werden.

6.3.4 Webserver

Ein Webserver wird benötigt, damit der Softwareprototyp seine Benutzeroberfläche und Schnittstellen über ein Netzwerk, bspw. das Internet, bereitstellen kann.[8] Zu diesem Zweck übersetzt ein Webserver eine URL in einen Datei- oder Applikationsnamen und überträgt entweder die betreffende Datei oder die Ausgabe der entsprechenden Applikation.[9]

Folgende Webserver stehen zur Auswahl:

---

1) Vgl. Kofler/Kramer (2005), S. 4. CouchDB präsentiert die Informationen im JSON-Format, vgl. Andeson/Lehnhardt/Slater (2010), S. 120, Watson (2009), S. 255, sowie Lennon (2009), S. 4, S. 50 sowie S. 87 ff.

2) Vgl. Lennon (2009), S. 7.

3) Vgl. Lennon (2009), S. 5 f.

4) Vgl. Lennon (2009), S. 6.

5) Vgl. Lennon (2009), S. 7, Anderson/Lehnhardt/Slater (2010), S. 13 f., sowie Watson (2009), S. 255 f.

6) Vgl. Anderson/Lehnhardt/Slater (2010), S. 33 ff.

7) Vgl. Lennon (2009), S. 7.

8) Vgl. Kersken (2007), S. 273.

9) URL bedeutet "uniform resource locator", vgl. Laurie/Laurie (2003), S. 1.

6.3.4.1 Apache

Apache ist ein sicherer, schneller Webserver und von den hier beschriebenen Webservern im Funktionsumfang der komplexeste.[1]

Apache ist über das Internet kostenlos verfügbar.[2]

Apache ist modular aufgebaut und kann durch entsprechende Erweiterungen komplexe Manipulationen von HTTP-Kopfdaten und URLs durchführen, als Proxyserver eingesetzt werden und die Kommunikation zwischen Browser und Webserver verschlüsseln.[3]

Apache eignet sich prinzipiell zur Entwicklung des Softwareprototyps, wenngleich der Einsatz eines im Funktionsumfang so komplexen Webservers ungerechtfertigt erscheint.[4]

6.3.4.2 Mongrel

Mongrel ist ein Webserver, der nativ über HTTP mit Ruby on Rails kommunizieren kann, was eine höhere Ausführungsgeschwindigkeit zur Folge hat.[5] Mongrel ist in Ruby und C implementiert, plattformunabhängig und frameworkagnostisch.[6]

Mongrel ist über das Internet kostenlos verfügbar.[7]

Mongrel wurde als Alternative zu WEBrick entwickelt und ist heute einer der bevorzugten Webserver zum Entwickeln und Betreiben von Rails-Applikationen.[8] Mongrel ist für den Einsatz in einem Produktionssystem geeignet.[9]

6.3.4.3 WEBrick

WEBrick ist der Standard-Webserver des Ruby-Kernpakets.[10] WEBrick unterstützt alle Funktionen, die ein Webframework zum Betrieb benötigt, wie bspw. HTTP, HTML und ERb, ist sehr einfach zu konfigurieren und für die Entwicklung eines Softwareprototyps als ausreichend performant zu erachten.[11]

WEBrick ist als Teil des Ruby-Kernpakets kostenlos verfügbar.[12]

---

1) Apache ist die gebräuchliche Kurzform für das Produkt "Apache HTTP Server" der Apache Foundation, vgl. Laurie/Laurie (2003), S. 1, sowie Ediger (2008), S. 318.

2) Alle nötigen Informationen sind unter der URL http://www.apache.org/ verfügbar.

3) Vgl. Laurie/Laurie (2003), S. 84, S. 179, S. 188 sowie S. 230.

4) Wenn sie die Wahl haben, entscheiden sich Administratoren häufig für leichtgewichtigere Lösungen, vgl. Ediger (2008), S. 318.

5) Vgl. Ediger (2008), S. 320 sowie Dobbs-Sciortino (2006), S 2.

6) Vgl. Dobbs-Sciortino (2006), S 2.

7) Alle nötigen Informationen sind unter der URL http://mongrel.rubyforge.org/ verfügbar.

8) Vgl. Ediger (2008), S. 78 sowie S. 319, sowie Dobbs-Sciortino (2006) S 2.

9) Vgl. Tate/Carlson/Hibbs (2009), S. 152. Es ist allerdings empfehlenswert, Mongrel nicht als singuläre Instanz laufen zu lassen, die alle Anfragen beantwortet, sondern einen "Load Balancer" vorzuschalten, vgl. Dobbs-Sciortino (2006) S 8.

10) Vgl. Carlson/Richardson (2006), S. 546.

11) Vgl. Carlson/Richardson (2006), S. 547.

12) Vgl. S. 16.

WEBrick wird in neuerer Zeit auch bei Entwicklungssystemen zunehmend von Mongrel verdrängt.[1] WEBrick eignet sich dennoch zur Entwicklung des Softwareprototyps durch den geringeren Konfigurationsaufwand und die damit verbundene positive Wirkung auf die Entwicklungszeit.

6.3.5 Versionsverwaltung

Im Rahmen der Versionsverwaltung geht es darum, Änderungen an einem Quelltext zu erfassen und eine mit Zeitstempeln archivierte Historie aller hierfür vorgesehenen Dateien zu führen.[2] So können Änderungen, die vom Entwickler im Nachhinein als falsch, schadhaft oder anderweitig unvorteilhaft identifiziert werden, zurückgenommen werden, indem aus dem Archiv der Versionsverwaltung eine ältere, als besser erachtete Version der betreffenden Dateien wiederhergestellt wird.[3]

Folgende Versionsverwaltungen stehen zur Auswahl:

6.3.5.1 Git

Git ist eine dezentralisierte Versionsverwaltung zur Verwaltung der Änderungen an Verzeichnissen.[4] Ziel von Git ist es, die distribuierte, unabhängige und gleichzeitige Entwicklung in privaten Repositories zu ermöglichen und zu fördern.[5] Git lehnt sich dabei an Konzepte von Bitkeeper und Mercurial an.[6]

Git ist über das Internet kostenlos verfügbar.[7]

In Git sind Transaktionen atomar und einmal eingecheckte Daten unveränderbar.[8] Git ist sowohl beim „branching" und „merging" als auch beim Übertragen der eingecheckten Änderungen über das Netzwerk sehr schnell und effizient.[9] Hierdurch können bspw. Ideen für neue Funktionalitäten schnell und unabhängig voneinander (und unabhängig vom Hauptentwicklungszweig) ausprobiert, evaluiert und in andere Entwicklungszweige übernommen oder verworfen werden.[10]

Git ist gut geeignet, die Entwicklung eines Softwareprototyps zu unterstützen, da Git gegenüber anderen gängigen und kostenlosen Versionsverwaltungen entscheidende Vorteile bei der schnellen, flexiblen Entwicklung von Software bietet.[11]

---

1) Vgl. Tate/Carlson/Hibbs (2009), S. 8.

2) Der Vorgang des Hinzufügens von Änderungen wird im Folgenden als "einchecken" bezeichnet. Das Archiv, das alle eingecheckten Änderungen enthält, wird im Folgenden als "Repository" bezeichnet, vgl. Pilato/Collins-Sussman/Fitzpatrick (2008), S. 1.

3) Vgl. Chacon (2009), S. 1, sowie Berlin/Rooney (2006), S. 1.

4) Vgl. Loeliger (2009), S. 1.

5) Vgl. Loeliger (2009), S. 2.

6) Vgl. Loeliger (2009), S. 2 f.

7) Alle nötigen Informationen sind unter der URL http://git-scm.com/ verfügbar.

8) Vgl. Loeliger (2009), S. 3.

9) Vgl. Loeliger (2009), S. 2 f.

10) Vgl. Loeliger (2009), S. 89 f.

11) Die entscheidenden Vorteile sind: die Erleichterung des distribuierten Arbeitens durch einfaches und schnelles „branching" und „merging" sowie die Tatsache, dass jede Arbeitskopie eine komplette Historie aller Entwicklungszweige enthält, vgl. Loeliger (2009), S. 3 f., sowie Chacon (2009), S. 5.

6.3.5.2 SVN

SVN ist eine zentralisierte Versionsverwaltung zur Verwaltung der Änderungen an einzelnen Dateien.[1] Das Konzept von SVN lehnt sich stark an das von CVS an.[2] Das Ziel von SVN ist es, die Fehler, die bei der Konzeption von CVS gemacht wurden, nicht zu wiederholen.[3] Aus diesem Grund ähneln sich Arbeitsabläufe unter CVS und SVN stark, was nicht zuletzt dazu beigetragen hat, dass viele unzufriedene CVS-Benutzer zu SVN wechselten.[4]

SVN ist über das Internet kostenlos verfügbar.[5]

SVN speichert Änderungen atomar, bietet Unterstützung von binären Dateien und erlaubt das Abspalten einzelner Entwicklungszweige und -versionen mittels „branching" und „tagging".[6] SVN speichert die eingecheckten Änderungen dabei zentral auf einem Server, auf den alle Entwickler eines Projekts Zugriff haben können.[7] SVN ist kein SCM[8], also keine Versionsverwaltung, die speziell für das Archivieren von Quelltexten entwickelt wurde.

SVN ist durchaus geeignet, die Entwicklung eines Softwareprototyps zu unterstützen. Allerdings wirkt sich der, insbesondere im Vergleich zu Git, höhere Konfigurationsaufwand negativ auf die Entwicklungszeit aus.[9]

6.3.6 Browsertechnologien

Da bei den Browsertechnologien keine direkte Auswahlproblematik besteht, werden im Folgenden die eingesetzten Browsertechnologien beschrieben.

6.3.6.1 HTML

HTML ist eine Auszeichnungssprache, mit deren Hilfe die Logik und Struktur von Webseiten beschrieben und so visuell darstellbar gemacht wird.[10] Sie dient der Strukturierung von Informationen in Dokumenten und wird vom W3C weiterentwickelt.[11] HTML-Dokumente enthalten sowohl sicht-

---

1) SVN bedeutet "Subversion", vgl. Pilato/Collins-Sussman/Fitzpatrick (2009), S. 318.

2) CVS bedeutet "Concurrent Versions System" und ist ebenfalls eine Versionsverwaltung, vgl. Pilato/Collins-Sussman/Fitzpatrick (2009), S. 19.

3) Vgl. Pilato/Collins-Sussman/Fitzpatrick (2009), S. 19.

4) Vgl. Berlin/Rooney (2006), S. 5.

5) Alle nötigen Informationen sind unter der URL http://svn.apache.org/ verfügbar.

6) Vgl. Berlin/Rooney (2006), S. 4 f.

7) Vgl. Berlin/Rooney (2006), S. 4.

8) SCM bedeutet "software configuration management". SCM bietet einige Vorteile gegenüber reiner Versionsverwaltung, da sie den eingecheckten Quelltext auch syntaktisch versteht und Werkzeuge, bspw. zu dessen Kompilierung und Distribution, bereitstellt, vgl. Pilato/Collins-Sussman/Fitzpatrick (2008), S. xiii.

9) Der Grund hierfür ist, dass für SVN ein dedizierter Server konfiguriert und betrieben werden muss, vgl. Berlin/Rooney (2006), S. 4, sowie Pilato/Collins-Sussman/Fitzpatrick (2008), S. 191 ff.

10) Vgl. Musciano/Kennedy (2007), S. 8.

11) W3C bedeutet "World Wide Web Consortium". Das W3C ist eine der Standardisierungsorganisationen des Internets, vgl. Musciano/Kennedy (2007), S. 7 f.

bare Informationen, wie Bildverweise, Texte und Links, als auch unsichtbare Metainformationen, welche für den Browser oder Suchmaschinen wie bspw. Google relevant sind.[1]

HTML-Dateien enthalten dabei idealerweise nur die logische Struktur eines Dokuments und stellen qualitativ hochwertige Informationen zur Verfügung, während die Präsentation dieser Struktur und Informationen der CSS-Technologie überlassen wird.[2]

Die Anwendung von HTML geht heute jedoch über die Darstellung klassischer Webseiten hinaus. So werden immer häufiger Programmkomponenten durch eine Kombination von HTML und CSS dargestellt,[3] und auch im Bereich der Mobilgeräte gewinnt HTML zunehmend an Bedeutung.[4]

6.3.6.2 CSS

CSS ist eine Technologie, um Informationen, bspw. in HTML-Seiten, zu visualisieren.[5] Dabei geht es primär um die ansprechende visuelle Aufbereitung der Inhalte durch Farben, Formen und Grafiken, aber auch um die Barrierefreiheit angebotener Inhalte, wie bspw. die Lesbarkeit von Webseiten durch sogenannte Screenreader.[6]

Bevor das W3C 1996 das erste Konzept zu CSS veröffentlichte,[7] mussten alle visuellen Eigenschaften von HTML-Elementen in den Elementen selbst definiert werden.[8] Mit der Einführung von CSS wurde es möglich, generelle Regeln zur Darstellung der HTML-Elemente zu definieren.[9]

Ein entscheidender Grundgedanke von CSS ist es, die Präsentationsinformationen zu den in HTML-Dateien vorhandenen Daten von den HTML-Dateien selbst zu trennen und an eine andere Stelle auszulagern.[10] Das Konzept von CSS erreicht dabei reichhaltigere Gestaltungsmöglichkeiten und geringere Dateigrößen durch die kaskadierende Natur von CSS.[11]

Die Anwendung von CSS geht heute weit über den Einsatz zur Visualisierung von Webseiten hinaus. Teilweise stellen auch Applikationen, wie bspw. der Mozilla-Browser, ihre eigenen Programmkomponenten mithilfe von CSS dar.[12]

---

1) Vgl. W3C (2010), o.S., sowie Musciano/Kennedy (2007), S. 8 sowie S. 208 ff.

2) Vgl. Musciano/Kennedy (2007), S. 9 sowie S. 230.

3) Vgl. Meyer (2007), S. 1.

4) Vgl. Musciano/Kennedy (2007), S. 457 ff.

5) Vgl. Andrew/Shafer (2006), S. 3 ff., sowie S. 23 ff. sowie W3C (2010), o.S.

6) Screenreader sind Programme, die Webseiten vorlesen (bspw. Menschen mit Sehschwäche).

7) Vgl. Musciano/Kennedy (2007), S. 230.

8) Bspw. musste in jeder einzelnen Überschrift explizit definiert werden, welche Farbe die jeweilige Überschrift haben sollte.

9) Bspw. konnte nun festgelegt werden, dass alle Überschriften grau dargestellt werden sollen.

10) Vgl. Meyer (2007), S. 3 ff.

11) Vgl. Meyer (2007), S. 3 ff.

12) Vgl. Meyer (2007), S. 1.

6.3.6.3 JavaScript

JavaScript ist eine Skriptsprache, welche dem Benutzer die Interaktion mit sonst statischen Webseiten ermöglicht und als „Sprache des Internets" bezeichnet werden kann.[1] JavaScript vereinfacht die Interaktion mit Webseiten, indem es Manipulationen des sonst statischen Inhalts ermöglicht.[2] So wird es möglich, auf Benutzerinteraktionen zu reagieren (bspw. durch Öffnen eines Popup-Fensters),[3] Informationen auf dem Rechner des Benutzers zu speichern[4] und mit Formularen zu interagieren.[5] Es existieren dabei keine Alternativen zum Einsatz von JavaScript, da JavaScript als einzige Sprache in allen modernen Webbrowsern implementiert ist.[6]

Das Einsatzgebiet von JavaScript geht heute, ähnlich dem von HTML und CSS, weit über die Implementierung klassischer Webseiten hinaus.[7]

6.4 Auswahl der Arbeitstechniken

6.4.1 Konzeption der Auswahl

Es werden zunächst sukzessiv die Arbeitstechniken Programmiersprache, Webframework, Datenbanksystem und Webserver gewählt. Dies bedeutet, dass auf jeder Entscheidungsstufe die Entscheidung über die vorangegangene Arbeitstechnik bereits getroffen wurde und so bei der Auswahl der nächsten Arbeitstechnik bekannt ist und berücksichtigt wird.

Abbildung 5 stellt diesen Ablauf verdeutlichend dar.

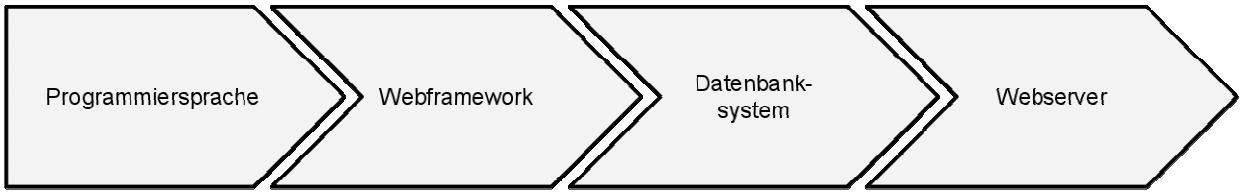

Abbildung 5: Darstellung der sukzessiven Auswahl der Arbeitstechniken

Die Arbeitstechniken Versionsverwaltung und Browsertechnologien werden im Anschluss gewählt.[8]

---

1) Vgl. Crockford (2008), S. 1.

2) Vgl. Flanagan (2006), S. 1 sowie S. 4 ff.

3) Vgl. Flanagan (2006), S. 280 ff.

4) Vgl. Flanagan (2006), S. 459 ff.

5) Vgl. Flanagan (2006), S. 438 ff.

6) Vgl. Crockford (2008), S. 4.

7) So lassen sich mittlerweile (dank stetig leistungsfähiger werdender VMs) vollwertige Applikationen, wie bspw. Webserver, mit JavaScript realisieren, vgl. Stefanov (2010), S. 1.

8) Der Grund hierfür ist, dass sowohl die Browsertechnologien als auch die Versionsverwaltung komplett unabhängig von den zuvor gewählten Arbeitstechniken sind. Ihre Auswahl hat keinerlei Auswirkung auf die Auswahl anderer Arbeitstechniken. Sie könnten auch vor den anderen Arbeitstechniken gewählt werden.

6.4.2 Programmiersprache

Tabelle 1 stellt eine Übersicht und Beurteilung der vorgestellten Programmiersprachen dar.[1]

Arbeitstechnik	geringe Kosten	geringe Entwicklungszeit	hohe Flexibilität	hohe Ausführungs-geschwindigkeit	hohe Sicherheit
Ruby	+ +	+ +	+ +	-	-
C\#	+ +	- -	+	+ +	+
Java	+ +	- -	-	+ +	+
PHP	+ +	-	+	+ +	- -

Tabelle 1: Beurteilung von Programmiersprachen bezüglich ihrer Eignung zur Implementierung des Softwareprototyps

+ +	sehr gute Eignung
+	gute Eignung
-	geringe Eignung
- -	schlechte Eignung

Tabelle 2: Legende zur Beurteilung der Arbeitstechniken

Ruby ist die flexibelste der beschriebenen Programmiersprachen, wodurch eine schnelle, elegante und vor allem problemorientierte Programmierung unterstützt wird und Konzepte der Metaprogrammierung umgesetzt werden können.

1) Die Beurteilung dieser und aller weiterer Arbeitstechniken erfolgt auf Basis der in Kapitel 6.3 dargelegten Eigenschaften der jeweiligen Arbeitstechnik sowie den sich ergebenden Vor- und Nachteilen für die schnelle Implementierung eines Softwareprototyps. Die Beurteilung wird subjektiv durch den Autor auf Basis seiner professionellen Erfahrungen vorgenommen und dient der groben Einordnung der Arbeitstechniken in Bezug auf die in Kapitel 6.2 identifizierten Kriterien. Aus Gründen der Vollständigkeit werden sowohl primäre als auch sekundäre Kriterien aufgeführt. Das Kriterium der "geringen Kosten" wird aufgeführt, obwohl seine Ausprägung konstant und damit für die Beurteilung irrelevant ist. Das Kriterium "Qualifikation, Verfügbarkeit und Kosten des zur Weiterentwicklung des Softwareprototyps nötigen Fachpersonals" wird nicht aufgeführt, da die hierfür nötige Recherche den zeitlichen und fachlichen Rahmen dieses Projektberichts überschritten hätte. Die erstellten Beurteilungen erheben keinerlei Anspruch auf Objektivität, Reliabilität oder Validität.

Ruby ist quelloffen und wird für Windows und unixoide Systeme bereitgestellt. C# wird dagegen offiziell nur für Windows zur Verfügung gestellt, wodurch die Plattformunabhängigkeit eingeschränkt ist.[1]

Als Nachteil von Skriptsprachen kann angeführt werden, dass die resultierende Applikation häufig langsamer in der Ausführung ist als eine analog in einer kompilierbaren Sprache geschriebene Applikation. Dieser Umstand ist jedoch vernachlässigbar.[2]

Ruby ist in der Ausführung bestimmter Operationen nicht so schnell wie Java, C# oder PHP. Dieses Problem ist bei der Entwicklung eines Softwareprototyps jedoch weniger relevant als bei der Konzeption und Entwicklung eines Produktionssystems.[3]

Im Gegensatz zu Java, C# und Ruby ist die Objektorientierung in PHP nachträglich implementiert worden, um so sowohl die prozedurale als auch die objektorientierte Programmierung zu unterstützen.[4] Viele PHP-Entwickler versuchen, diese Tatsache als besonders wertvolle Eigenschaft der Progammiersprache darzustellen.[5] Die aus dieser Abwärtskompatibilität resultierende Konsequenz, dass die meisten Standardbibliotheken noch prozedural angelegt sind, wirkt sich jedoch negativ auf die Entwicklungszeit aus.[6] Somit wirkt sich die Möglichkeitsvielfalt der Programmiersprache PHP negativ auf die reale Flexibilität ihres Einsatzes aus, da stets eruiert werden muss, welche Bibliotheken in welcher Kombination wie gut zusammenspielen.

Ruby bietet somit alle benötigten Funktionalitäten zur schnellen Implementierung eines Softwareprototyps einer Online-Frachtenbörse und wird daher als Programmiersprache gewählt.

---

1) Es existieren Open-Source-Projekte, die C# auch für unixoide Systeme zur Verfügung stellen. Allerdings werden diese Projekte nicht von Microsoft unterstützt.

2) Begründet wird dies durch folgende Plausibilitätsüberlegung: Die immer schneller werdenden Laufzeitinterpreter moderner Skriptsprachen und die stetig zunehmende Rechenleistung moderner Computer führen dazu, dass die Lücke zwischen der Ausführungsgeschwindigkeit einer interpretierten und einer kompilierten Applikation stetig kleiner wird. Insbesondere vernachlässigbar wird dieser Unterschied bei der Implementierung eines Softwareprototyps.

3) Ruby bietet zudem Möglichkeiten, besonders lastsensible Teile einer Applikation durch die Programmierung sogenannter "C-Extensions" oder den Einsatz von "JRuby" zu beschleunigen, vgl. Carlson/Richardson (2006), S. 817 ff.

4) Vgl. Lerdorf/Tatroe/MacIntyre (2006), S. 30 sowie S. 143.

5) Vgl. Gilmore (2008), S. 9.

6) Es erscheint plausibel, dass eine prozedurale Programmiersprache, die nachträglich um Objektorientierung erweitert wurde, nicht im gleichen Maße beide Programmierstile unterstützen kann, wie eine objektorientierte Programmiersprache, die lediglich Paradigmen des prozeduralen Programmierstils zulässt.

6.4.3 Webframework

Tabelle 3 stellt eine Übersicht und Beurteilung der vorgestellten Webframeworks dar.

Arbeitstechnik	geringe Kosten	geringe Entwicklungszeit	hohe Flexibilität	hohe Ausführungs-geschwindigkeit	hohe Sicherheit
Ruby on Rails	+ +	+ +	+	-	+
Merb	+ +	-	+ +	+	+
Sinatra	+ +	- -	+ +	+ +	+

Tabelle 3: Beurteilung von Webframeworks
bezüglich ihrer Eignung zur Implementierung des Softwareprototyps

Merb und Sinatra sind mächtige und im Kern schlanke Webframeworks, deren produktiver Einsatz jedoch Vorarbeit erfordert, um alle benötigten Zusatzkomponenten zu identifizieren und zu installieren.

Seit der Zusammenlegung der Open-Source-Projekte Merb und Ruby on Rails ist die Zukunft von Merb als eigenständige Technologie ungewiss. Auch Merbs potentieller Nachfolger „Ramaze" erscheint noch zu unausgereift, um die schnelle Entwicklung eines Softwareprototyps einer Online-Frachtenbörse zu unterstützen.

Sinatra eignet sich eher zur Integration von Webservices als zur schnellen Umsetzung komplexer Prototypen.

Ruby on Rails führt Grundideen von Ruby konsequent fort und bietet eine Vielzahl an vorgefertigten Lösungen für standardisierte Probleme, was eine schnelle Entwicklungszeit begünstigt.[1] Es ist weniger flexibel als Merb oder Sinatra, allerdings wird dieser Nachteil durch die sonst bessere Eignung zur schnellen Implementierung des Softwareprototyps vernachlässigbar.[2]

Ruby on Rails bietet somit alle benötigten Funktionalitäten zur schnellen Implementierung eines Softwareprototyps einer Online-Frachtenbörse und wird daher als Webframework gewählt.

1) Vgl. Carneiro/Barazi (2010), S. 4 ff.

2) Ruby on Rails gibt einige Standardverfahren und Paradigmen vor, die als inflexibel charakterisiert werden können. Allerdings begünstigen diese Paradigmen die schnelle Implementierung einer komplexen, datenbankgestützten Webapplikation, vgl. Carneiro/Barazi (2010), S. 4 ff.

6.4.4 Datenbanksystem

Tabelle 4 stellt eine Übersicht und Beurteilung der vorgestellten Datenbanksysteme dar.

Arbeitstechnik	geringe Kosten	geringe Entwicklungszeit	hohe Flexibilität	hohe Ausführungs-geschwindigkeit	hohe Sicherheit
SQLite	+ +	+ +	+	+	+
MySQL	+ +	+	+	+ +	+
CouchDB	+ +	-	+ +	-	+

Tabelle 4: Beurteilung von Datenbanksystemen bezüglich ihrer Eignung zur Implementierung des Softwareprototyps

Die beschriebenen RDBMS MySQL und SQLite bieten für die Entwicklung des Softwareprototyps die gleichen Basisfunktionen. MySQL ist in bestimmten Fällen schneller, allerdings auch aufwendiger zu konfigurieren als SQLite und erfordert zusätzlich den Betrieb eines dedizierten Servers.

Das dokumentorientierte Datenbanksystem CouchDB ist für die Entwicklung des Softwareprototyps nicht so gut geeignet wie ein RDBMS, da das dokumentorientierte Konzept weniger gut für eine Online-Frachtenbörse geeignet ist als ein tabellenorientiertes. Zudem abstrahiert Ruby on Rails alle Anfragen an das Datenbanksystem, wodurch das Konzept des Datenbanksystems und die mit ihm verbundenen Vorteile bei sachgerechter Implementierung im Rahmen der Implementierung des Softwareprototyps unwichtig werden.

Für ein Produktionssystem würde MySQL aufgrund seiner flexiblen Einsetzbarkeit und weiten Verbreitung bevorzugt. Jedoch ist die Konfiguration des MySQL-Servers im Vergleich zu SQLite aufwendig, während SQLite für die Implementierung eines Softwareprototyps die gleichen Basisfunktionen bereitstellt wie MySQL und damit eine schnelle Entwicklungszeit begünstigt.

SQLite bietet somit alle benötigten Funktionalitäten zur schnellen Implementierung eines Softwareprototyps einer Online-Frachtenbörse und wird daher als Datenbanksystem gewählt.

6.4.5 Webserver

Tabelle 5 stellt eine Übersicht und Beurteilung der vorgestellten Webserver dar.

Arbeitstechnik	geringe Kosten	geringe Entwicklungszeit	hohe Flexibilität	hohe Ausführungs-geschwindigkeit	hohe Sicherheit
WEBrick	+ +	+ +	-	-	-
Mongrel	+ +	+	+	+	-
Apache	+ +	-	+ +	+ +	+

Tabelle 5: Beurteilung von Webservern bezüglich ihrer Eignung zur Implementierung des Softwareprototyps

Alle Webserver bieten für die Entwicklung des Softwareprototyps die gleichen Basisfunktionen. Apache ist schneller als WEBrick und Mongrel, erfordert allerdings eine umfangreiche Konfiguration des Servers.

WEBrick und Mongrel hingegen sind weniger komplex und sehr einfach zu konfigurieren, was die möglichst schnelle Implementierung eines Softwareprototyps unterstützt.

Beim Vergleich von Mongrel und WEBrick ist festzuhalten, dass WEBrick zwar langsamer ist als Mongrel, dafür allerdings ohne weitere Konfiguration sofort mit Ruby on Rails genutzt werden kann. WEBrick bietet somit alle benötigten Funktionalitäten zur schnellen Implementierung eines Softwareprototyps einer Online-Frachtenbörse und wird daher als Webserver gewählt.

6.4.6 Versionsverwaltung

Tabelle 6 stellt eine Übersicht und Beurteilung der vorgestellten Software zur Versionsverwaltung dar.

Arbeitstechnik	geringe Kosten	geringe Entwicklungszeit	hohe Flexibilität	hohe Ausführungs-geschwindigkeit	hohe Sicherheit
Git	+ +	+ +	+	+ +	+ +
SVN	+ +	+	+	-	-

Tabelle 6: Beurteilung von Software zur Versionsverwaltung bezüglich ihrer Eignung zur Implementierung des Softwareprototyps

Die hohe Flexibilität von Git im Bereich „branching and merging" und die Schnelligkeit bei der Ausführung dieser Operationen begünstigen eine schnelle Entwicklungszeit, da so während der Entwicklung leicht Ideen ausprobiert und anschließend mühelos verworfen oder integriert werden können. SVN bietet diese Funktionalitäten ebenfalls, allerdings ist die Nutzung aufwendiger und teilweise weniger performant.[1]

Ebenfalls für Git spricht die Tatsache, dass kein Server betrieben werden muss, um ein Repository anzulegen und Änderungen einzuchecken. So kann Git auch ohne Netzwerkkonnektivität voll genutzt werden. Auch die distribuierte Natur der Repositories und die Tatsache, dass jede Arbeitskopie die komplette Historie aller Entwicklungszweige eines Repositories enthält, sprechen für Git.

Git bietet somit alle benötigten Funktionalitäten zur schnellen Implementierung eines Softwareprototyps einer Online-Frachtenbörse und wird daher als Versionsverwaltung gewählt.

6.4.7 Browsertechnologien

Wie bereits beschrieben, gibt es zu den vorgestellten Browsertechnologien keine wirtschaftlich einsetzbaren Alternativen.

Daher werden diese Arbeitstechniken keiner Beurteilung unterzogen.

Als Browsertechnologien werden HTML, CSS sowie JavaScript gewählt.

1) Es gibt in SVN kein natives "branching". "Merging" ist ein umständlicher, langsamer Prozess, der häufig manuell durchgeführt werden muss, vgl. Pilato/Collins-Sussman/Fitzpatrick (2008), S. 48 ff. sowie S. 59 ff.

7 Leistungsumfang des Softwareprototyps

7.1 Konzeption des Leistungsumfangs

Der Softwareprototyp wird als erweiterbares „Schwarzes Brett"-System konzipiert.[1]

Es können im System der Online-Frachtenbörse also Inserate für Fracht- und Laderaum eingestellt werden. Es werden jedoch keine monetären Informationen erfasst.[2] Deshalb arbeitet die Matching-Funktionalität nicht als Auktionsmechanismus, bspw. über Preisagenten, sondern die Inserate für Fracht- und Laderaum werden auf ihre Kompatibilität zueinander analysiert.[3] Die Applikation schlägt Anbietern auf Basis dieser Kompatibilitätsanalyse passende Inserate zu den eigenen Inseraten vor und versucht so, Angebot und Nachfrage zusammenzubringen.[4]

Die Leistungen des Softwareprototyps umfassen:[5]

- die Registrierung von Unternehmen im System der Online-Frachtenbörse,
- das Erstellen und Verwalten mehrerer Benutzer innerhalb eines Unternehmens,
- die Authentifizierung verschiedener Benutzergruppen durch ein Rechtesystem,
- das Hinterlegen von Kontaktinformationen für Unternehmen und Benutzer,
- das Einstellen von eigenen Inseraten,
- die Suche nach bestehenden Inseraten mittels einer Schnittstelle (Such-API),
- das Matching von Inseraten mittels einer Schnittstelle (Matching-API),
- die Steuerung des Softwareprototyps mittels einer Schnittstelle (XML/JSON-API),
- die beispielhafte Ansteuerung dieser Schnittstelle durch Roboter,
- die Bewertung von Geschäftspartnern innerhalb des Systems der Online-Frachtenbörse,
- angepasste Benutzeroberflächen für die Betreiber der Online-Frachtenbörse,
- die Protokollierung des Nutzungsverhaltens der Benutzer sowie
- die Ausgabe von Statistiken über das Nutzungsverhalten der Benutzer.

Dieser Leistungsumfang des Softwareprototyps wird in drei Komponenten unterteilt:

- systemische Funktionen,
- Funktionen des Frontends sowie
- Funktionen des Backends.

---

1) Vgl. Klippert/Kowalski/Bruns (2010), S. 50.

2) Vgl. Merkel/Kromer (2002), S. 82.

3) Zur Funktionsweise der Matching-Funktionalität vgl. S. 39 ff.

4) Die eigentlichen Vertragsverhandlungen zwischen den Anbietern finden außerhalb des Systems der Online-Frachtenbörse statt.

5) Diese Aufzählung erhebt keinen Anspruch auf Vollständigkeit. So ergeben sich aus den aufgeführten Leistungen (und den sich daraus ergebenden, nötigen Funktionalitäten) wiederum weitere Anforderungen, wie bspw. die Möglichkeit zur generellen Administration der Online-Frachtenbörse, vgl. S. 54.

Mit „systemischen Funktionen" sind jene Funktionen gemeint, die nicht direkt Teil der Benutzeroberfläche sind, aber dennoch durch das System der Online-Frachtenbörse bereitgestellt werden.

Mit „Funktionen des Frontends" sind jene Funktionen gemeint, die Teil der für alle Benutzer zugänglichen Benutzeroberfläche sind.

Mit „Funktionen des Backends" sind analog jene Funktionen gemeint, die Teil der nur den Administratoren zugänglichen Systemkomponenten sind.

Abbildung 6 stellt die Komponenten des Leistungsumfangs mit ihren Komponenten und Akteuren zusammenfassend dar.

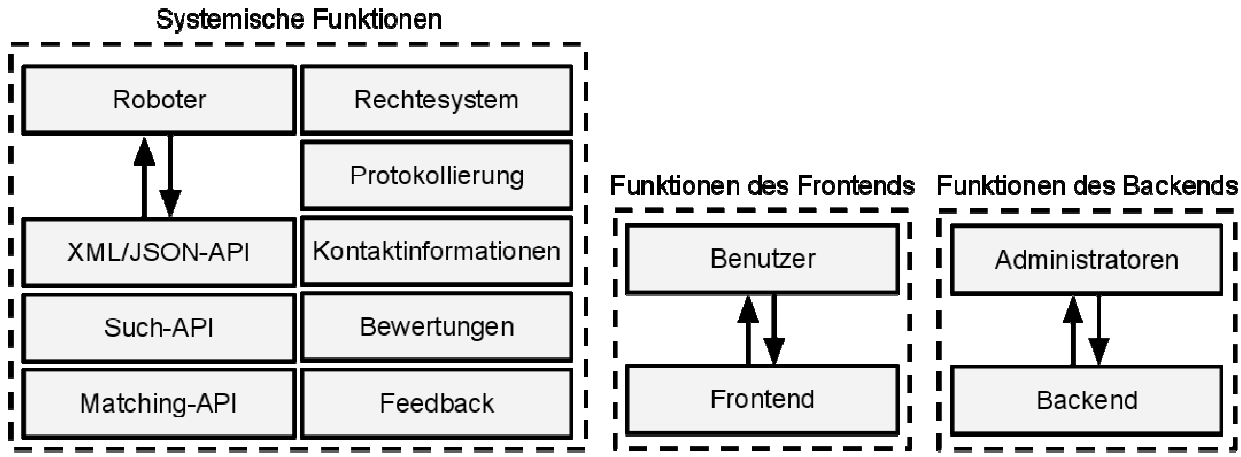

Abbildung 6: Darstellung der Komponenten des Leistungsumfangs

Teilweise entstehen durch diese Aufteilung Probleme bei der Zuordnung einzelner Funktionen der Online-Frachtenbörse. So ist die „Suche nach bestehenden Inseraten" eine durch den Softwareprototyp bereitgestellte Funktion. Hierbei ist die Such-API Teil der systemischen Funktionen, die Eingabemaske zur Suche und anschließenden visuellen Aufbereitung der Suchergebnisse eine Funktion des Frontends und das Auffinden von Benutzern im Rahmen der allgemeinen Benutzerverwaltung eine Funktion des Backends.

In derartigen Fällen werden die Teilaspekte der jeweiligen Funktionen möglichst trennscharf voneinander in den korrespondierenden Kapiteln erläutert.

In den folgenden Kapiteln wird dargestellt, wie die Funktionen des Systems, Frontends und Backends die Ziele des Softwareprototyps, wie bspw. das Auffinden neuer Geschäftspartner, unterstützen.

7.2 Systemische Funktionen

7.2.1 Modelle der Geschäftslogik

Modelle sind abstrakte, informationstechnische Datensammlungen der für den Betrieb einer Online-Frachtenbörse benötigten logischen und realen Systemkomponenten, wie bspw. eingestellte Inserate und teilnehmende Unternehmen.[1] Dabei abstrahieren Modelle von der Realität, um bspw. Objekte logisch besser voneinander trennen zu können.[2]

1) Vgl. Kannengiesser (2007), S. 419.

2) Bspw. bestehen Inserate aus mehreren, verschiedenartigen Objekten.

Es werden folgende Modelle implementiert:

- Unternehmen,
- Benutzer,
- Personen,
- Berechtigungen,
- Frachtrauminserate,
- Laderauminserate,
- Bewertungen sowie
- Einstellungen.

Unternehmen können sich im System der Online-Frachtenbörse registrieren und mehrere Benutzer anlegen.[1] Jedem Benutzer ist ein Personenobjekt zugeordnet, welches relevante Informationen über die hinter dem Benutzer stehende natürliche Person beinhaltet.[2] Benutzer können mit Berechtigungen ausgestattet werden. Je nach Berechtigung können die Benutzer des Unternehmens die Daten zu ihrer Person anpassen, das Anbieterprofil des Unternehmens ändern, Inserate einstellen, andere Anbieter bewerten und Bewertungen anderer Anbieter annehmen oder verwerfen.

Einstellungen bestehen aus einem eindeutigen Bezeichner und einem Wert.[3] Sie werden vom System genutzt, um umgebungsspezifisch Variablen zu speichern und abzufragen.[4] Für alle Einstellungen existieren dabei Standardwerte, die ausgegeben werden, falls keine umgebungsspezifische Version der betreffenden Variable existiert.

Abbildung 7 stellt ausgehend von einem Benutzerobjekt dar, welches Modell der Geschäftslogik welches andere Modell erzeugt oder während dessen Erzeugung beeinflusst.[5]

1) Der genaue Registrierungsprozess wird in diesem Projektbericht nicht gesondert dargestellt. Es handelt sich um eine internettypische Prozedur, in deren Verlauf eine E-Mail-Adresse hinterlegt und eine Benutzername/Passwort-Kombination gewählt werden muss.

2) Gemeint sind hier bspw. Vor- und Nachname sowie Kontaktdaten.

3) Zur Struktur der Datenbanktabelle für Einstellungen vgl. Anhang, S. 85.

4) Umgebungsspezifisch bedeutet, dass die Werte der Einstellungen für jede Installation des Softwareprototyps individuell konfiguriert werden können, da sie in der jeweiligen Datenbank gespeichert werden.

5) Einstellungen werden in dieser Abbildung nicht aufgeführt, da sie unabhängig von allen anderen Modellen der Geschäftslogik existieren.

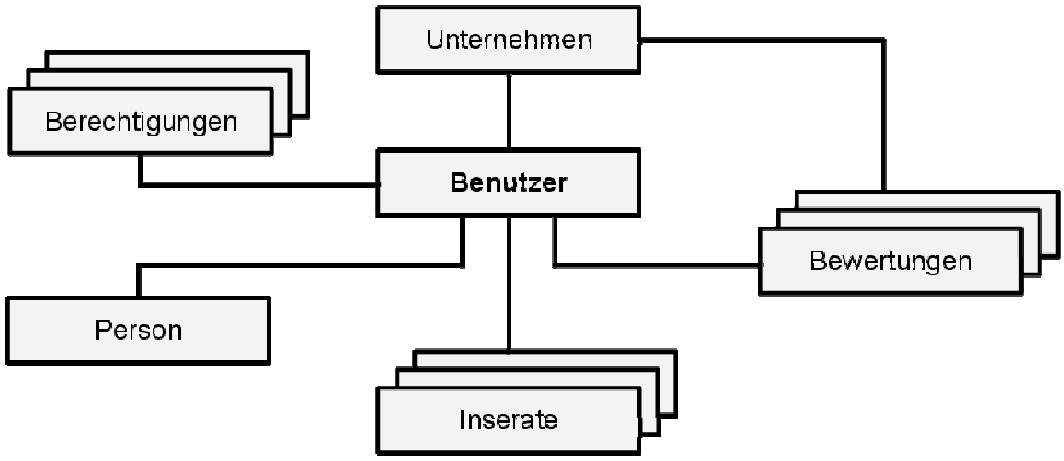

Abbildung 7: Darstellung der Modelle der Geschäftslogik

7.2.2 Verwaltung von Inseraten

Im System der Online-Frachtenbörse haben registrierte Benutzer die Möglichkeit, Fracht- und Laderaum für Einzelwagen-, Wagengruppen- und Ganzzugverkehre zu inserieren.

Das Einstellen von Inseraten[1] im System der Online-Frachtenbörse verfolgt das Ziel, Anbieter verschiedener Dienstleistungen zusammenzubringen und das Auffinden neuer Geschäftspartner im System der Online-Frachtenbörse zu ermöglichen.

Um dieses Ziel zu unterstützen, sind folgende Anforderungen zu erfüllen:

- Inserate müssen einfach zu erstellen, zu verwalten, aufzufinden und wiederzuverwenden sein. [2]
- Inserate müssen vom jeweiligen Anbieter gelöscht werden können.[3]
- Nicht mehr aktuelle Inserate müssen vom jeweiligen Anbieter identifiziert und gelöscht werden können.[4]
- Inserate müssen die angebotene Dienstleistung möglichst vollständig und universell beschreiben.[5]
- Inserate müssen mit korrespondierenden Inseraten anderer Anbieter vergleichbar sein.[6]
- Inserate müssen die Preisverhandlungen für die Dienstleistungen außerhalb des Systems

---

1) Unter dem Begriff "Inserat" sind sowohl Fracht- als auch Laderauminserate zusammengefasst, da trotz der separaten Speicherung der jeweiligen Objekte in der Datenbank die Struktur und Geschäftslogik dieser Objekte sehr ähnlich ist. Ausführungen zu Inseraten gelten, sofern nicht anders angegeben, stets sowohl für Fracht- als auch für Laderauminserate.

2) Vgl. Klippert/Kowalski/Bruns (2010), S. 73 f.

3) Vgl. Klippert/Kowalski/Bruns (2010), S. 80.

4) Vgl. Klippert/Kowalski/Bruns (2010), S. 50 sowie S. 75.

5) Dies bedeutet, dass Inserate möglichst alle relevanten Informationen atomar erfassen sollten, um sie so leichter maschinell vergleichbar zu machen, vgl. Klippert/Kowalski/Bruns (2010), S. 74. Zudem sollte die Möglichkeit bestehen, Marktbarrieren, wie bspw. Sprachbarrieren, zu senken, indem bspw. natürlichsprachliche Informationen in mehreren Sprachen hinterlegt werden können.

6) Dies bedeutet, dass zu einem Inserat für Frachtraum alle Laderauminserate gelistet werden können, welche möglichst gut zu diesem Inserat passen.

ermöglichen.[1]

Inserate werden von Benutzern über das Frontend erstellt und verwaltet. Alle aktuellen Inserate können von den Benutzern über die Suchfunktion des Frontends gefunden werden.[2]

Abbildung 8 stellt am Beispiel eines Frachtrauminserats dar, wie Inserate aus mehreren verschiedenartigen Objekten bestehen. Jedes Inserat gehört zu dem Benutzer, der es angelegt hat (und damit auch implizit zu dem Unternehmen dieses Benutzers).

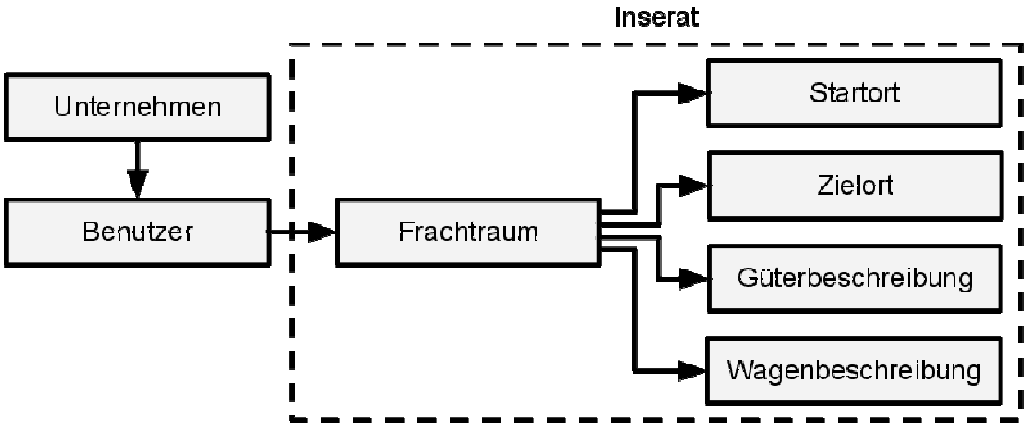

Abbildung 8: Darstellung der an einem Inserat beteiligten Objekte

Inserate erlauben die individuelle Angabe eines Ansprechpartners, so dass bspw. je nach Region oder Angebotstyp ein spezieller Ansprechpartner eingestellt werden kann, um so potentiellen Geschäftspartnern einen direkten Ansprechpartner zu präsentieren.

Die Informationen zu den Eigenschaften des Fracht- oder Laderaums werden direkt im Inserat, also der jeweiligen Datenbanktabelle gespeichert.[3]

Zusätzlich werden die qualitativen und damit natürlichsprachlich zu beschreibenden Attribute des Fracht- oder Laderaums ebenfalls in einer separaten Datenbanktabelle gespeichert. Hierdurch können die gleichen Informationen mehrmals in verschiedenen Sprachen hinterlegt werden.[4]

1) Vgl. Klippert/Kowalski/Bruns (2010), S. 77.

2) Gemeint sind hier Inserate, deren Startzeitpunkt in der Zukunft liegt. Nicht mehr aktuelle Inserate werden in den Suchergebnissen nicht aufgeführt.

3) Zur Struktur der Datenbanktabellen für Fracht- sowie Laderauminserate vgl. Anhang, S. 86 f.

4) Vgl. Klippert/Kowalski/Bruns (2010), S. 81.

Tabelle 7 zeigt die Struktur der zugehörigen Datenbanktabelle.

Name	Typ	Beschreibung	Beispiel
item_type	String	Verweis auf den Typ des Objekts, zu dem die Information gehört	„Freight"
item_id	Integer	Verweis auf die ID des Objekts, zu dem die Information gehört	1
name	String	Name	„misc_text"
lang	String	Sprache	„en"
text	Text	Text	„We are proud to..."

Tabelle 7: Struktur der „localized_infos"-Datenbanktabelle

Die Angaben zu Start- und Zielort sind strukturell identisch. Daher werden Informationen zu Start- und Zielort in separaten Objekten und in einer separaten Datenbanktabelle hinterlegt.[1] Neben dem Zeitpunkt, der Adresse und dem Namen des Dienstleisters erfassen diese Objekte auch Informationen über das Vorhandensein eines Gleisanschlusses.

Tabelle 8 zeigt die Struktur der zugehörigen Datenbanktabelle.

Name	Typ	Beschreibung	Beispiel
contractor	String	Name des Dienstleisters	„Mustermann GmbH"
name	String	Name des Ortes	„Musterstadt Bf"
address	String	Addresse	„Musterstr. 21"
address2	String	Addresse (Fortsetzung)	„3. Etage"
zip	String	Postleitzahl	„12345"
city	String	Ort	„Musterstadt"
country	String	Land	„Deutschland"
side_track_available	Boolean	Gleisanschluss vorhanden?	false
track_number	String	Gleisnummer	5

Tabelle 8: Struktur der „site_infos"-Datenbanktabelle

1) Der Grund hierfür ist, dass die Strukturen für bspw. Adressdaten von Start- und Zielort redundant sind. Anstatt im Fracht- bzw. Laderauminserat mehrere redundante Datenbankfelder zu implementieren, die sich lediglich durch das Prefix „start_" bzw. „ziel_" unterscheiden, werden alle Ortsangaben in einer separaten Tabelle gespeichert und vom Fracht- bzw. Laderauminserat aus referenziert.

7.2.3 Schnittstellen

Im Zentrum der systemischen Funktionen des Softwareprototyps stehen drei Funktionalitäten, die jeweils als anwendungsprogrammierbare Schnittstelle (API) implementiert werden.[1]

Es werden implementiert:[2]

- die Matching-API zur Analyse von bestehenden Inseraten,
- die Such-API zum Auffinden von bestehenden Inseraten sowie
- die XML/JSON-API zur Steuerung durch Drittanbietersoftware.

Abbildung 9 stellt dar, wie die Schnittstellen mit den Benutzern und möglicher Drittanbietersoftware sowie untereinander interagieren.

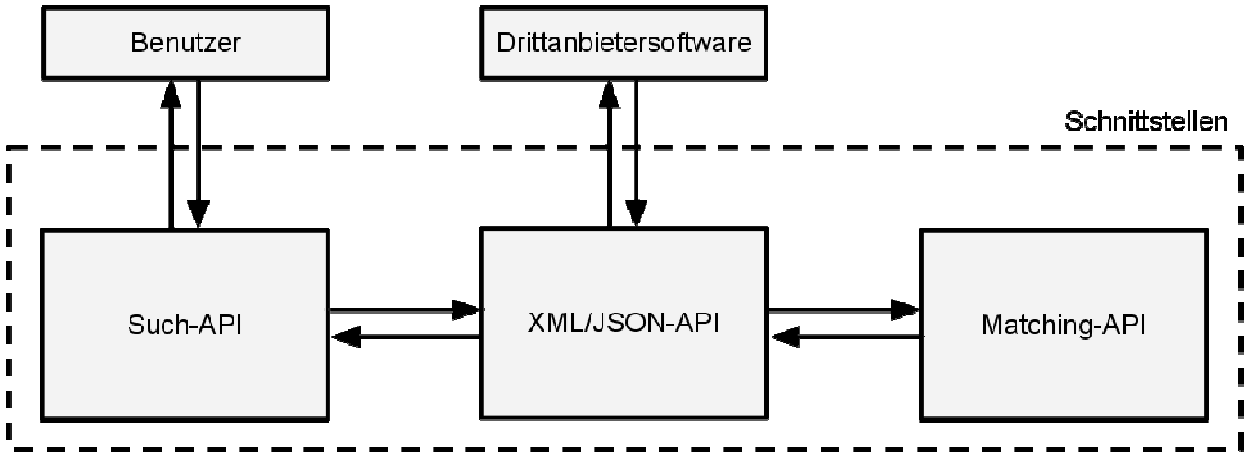

Abbildung 9: Darstellung der implementierten Schnittstellen

7.2.3.1 Matching-API zum Vergleich von Fracht- und Laderauminseraten

Die Matching-API untersucht Fracht- und Laderauminserate auf ihre Kompatibilität und stellt Strukturen zum Vergleich gleich- und verschiedenartiger Objekte bereit.

Die zentrale Aufgabe des Systems der Online-Frachtenbörse besteht im Zusammenbringen von Angebot und Nachfrage. Hierzu müssen Fracht- und Laderauminserate auf ihre Kompatibilität analysiert und anschließend, in Bezug auf den Grad dieser Kompatibilität geordnet, abgefragt werden können, um so auf Basis eines Inserats andere, passende Inserate ausgeben zu können. Die Matching-API implementiert die hierzu notwendigen Strukturen.

Die Matching-API des Softwareprototyps vergleicht Objekte auf ihre Kompatibilität. Im Falle zweier Inserate bedeutet dies, in welchem Maße das Angebot in Inserat A eine passende Nachfrage

---

1) API bedeutet "application programming interface", vgl. Carneiro/Barazi (2010), S. 136. Eine API bietet den Vorteil, dass kritische Teile einer Applikation modular entwickelt und gegebenenfalls ausgetauscht werden können. Das folgende Beispiel illustriert diesen Sachverhalt: Sollte sich herausstellen, dass die gewählten Arbeitstechniken zur Implementierung einer realen Online-Frachtenbörse im Allgemeinen ausreichend performant sind, jedoch die Vergleichsoperationen der Matching-API sich signifikant negativ auf die Ausführungsgeschwindigkeit auswirken, so könnte das für das Matching zuständige Modul mithilfe einer performanteren Arbeitstechnik implementiert werden. Da die Schnittstelle die Einbindung beliebiger Technologien ermöglicht, müsste an dem restlichen Softwareprototyp nichts verändert werden. Es würde lediglich ein anderes Matching-Modul an die Matching-API angebunden.

2) Streng genommen bildet die Benutzeroberfläche ebenfalls eine Schnittstelle (zur Interaktion zwischen Benutzer und Applikation). Diese wird jedoch in den Kapiteln 7.3 und 7.4 ausführlich behandelt und daher an dieser Stelle nicht aufgeführt.

zu dem Angebot in Inserat B darstellen könnte.
Da Objekte wie bspw. Inserate unterschiedlich strukturiert sein können (Laderauminserate unterscheiden sich in ihren Attributen und deren Ausprägungen von Frachtrauminseraten), ist die Matching-API nicht nur in der Lage, Regeln für zwei Vergleichsobjekte gleichen Typs zu definieren (um bspw. zwei Adressen oder Zeitpunkte miteinander zu vergleichen), sondern erlaubt auch das Definieren von Sonderregeln zum Vergleich zwei strukturell unterschiedlicher Objekte (so kann eine Sonderregel definiert werden, um ein numerisches Attribut in Objekt A mit einem alphanumerischen Attribut in Objekt B zu vergleichen).

Wie bereits angedeutet, basiert der Vergleich zweier Objekte auf sukzessiven Paarvergleichen. Der resultierende Wert jedes Vergleichs bewegt sich im Intervall [0,1]. Ein Comparer-Objekt definiert die zu vergleichenden Attribute der Objekte und die anzustellenden Vergleiche.

```
class PersonComparer < Matching::Compare::Base
 compare :weight
end
```

In obigem Beispiel wird ein Comparer für zwei Personen-Objekte definiert. Er bestimmt, dass die Personen nur anhand ihres Gewichts verglichen werden. Wiegen beide Personen das gleiche, so ergibt der Vergleich 100%. Wiegt eine Person 10% mehr oder weniger als die andere, ergibt der Vergleich 90%.

```
class PersonComparer < Matching::Compare::Base
 compare :weight
 compare :height
end
```

Im zweiten Beispiel werden nun nicht nur das Gewicht, sondern auch die Größe der Person als Vergleichsindikator herangezogen. Beide Vergleiche werden durchgeführt und die Ergebnisse ungewichtet arithmetisch gemittelt. Ergeben die Vergleiche 100% und 60%, so wird der Gesamtvergleich mit 80% gewertet.

In obigen Beispielen werden nur Zahlen verglichen und hierzu wird der Standardvergleich verwendet (in diesem Fall die prozentuale Abweichung eines Wertes von einem anderen). Häufig sind Vergleiche jedoch komplexer und bedürfen individuell definierter Vergleichsoperationen sowie der Möglichkeit, den gesamten Vergleich bei Nichterfüllung kritischer Faktoren scheitern zu lassen.

```
class PersonComparer < Matching::Compare::Base
 compare :weight
 compare :height
 compare :first_name do |a, b|
   a == b
 end
end
```

Der obige Comparer vergleicht nun zusätzlich den Vornamen der Personen anhand einer individuellen Regel. Sie überprüft die Identität der Vornamen und liefert hierbei einen der logischen Werte wahr oder falsch zurück. Im Falle von wahr wird der Vornamensvergleich mit 100% gewertet, im

Falle von falsch wird der gesamte Vergleich mit 0% gewertet - der Vergleich ist an einer kritischen Bedingung gescheitert.

Ein realistischeres Beispiel für diese Art von Vergleich wäre ein Comparer-Objekt, dass feststellen soll, ob zwei Personen in der gleichen Organisation arbeiten:

```
class PersonInSameCompanyComparer < Matching::Compare::Base
 compare :email do |a, b|
   email_domain = /[^@]+$/
   a[email_domain] == b[email_domain]
 end
end
```

Es macht wenig Sinn, die ganze E-Mail-Adresse zu vergleichen, daher vergleicht obiges Beispiel nur den Teil nach dem @ und somit die Organisationszugehörigkeit des Benutzers. Haben beide verglichenen Personen die gleiche Endung (bspw. @pim.uni-due.de), so besteht der E-Mail-Vergleich mit 100%, haben sie verschiedene Endungen, so scheitert der Gesamtvergleich mit 0%.

Die Matching-API funktioniert dabei grundsätzlich über Paarvergleiche der Ausprägungen einzelner Attribute von zwei Objekten A und B. Jeder Paarvergleich liefert einen numerischen Wert zwischen 0 und 1, wobei 0 die komplette Verschiedenheit und 1 die perfekte Übereinstimmung der Ausprägung des verglichenen Attributs kennzeichnet. Über die so ermittelten Paarvergleichswerte wird das ungewichtete arithmetische Mittel gebildet, wodurch für die Kompatibilität der Objekte A und B ebenfalls ein Wert zwischen 0 und 1 resultiert.

Dabei ist zu beachten, dass in der vorliegenden Implementierung immer Objekt A in Bezug auf Objekt B verglichen wird. In einigen Fällen würde der Paarvergleich „B mit A" zu demselben Übereinstimmungsgrad kommen, dies ist jedoch nicht notwendigerweise der Fall.[1]

Im Fall von unterschiedlich strukturierten Objekten, wie bspw. Fracht- und Laderauminseraten, bedeutet „perfekte Kompatibilität" natürlich nicht (wie im Fall der Paarvergleiche) die Identität beider Inserate, sondern, dass die Inserate sich perfekt ergänzen. Um unterschiedlich strukturierte Objekte vergleichen zu können, lassen sich in der DSL der Matching-API nicht nur generelle Regeln für jeden Datentyp definieren, sondern zusätzlich individuelle Vergleichsoperationen für jedes Attribut angeben.[2]

Abbildung 10 stellt dar, wie die Matching-API die Ausprägungen einzelner Attribute vergleicht, um einen Gesamtwert für die Kompatibilität der Objekte zu bilden.

1) So ist in der DSL der Matching-API die Möglichkeit gegeben, obere und untere Schwellenwerte der Toleranz anzugeben. Darf die Ausprägung eines Attributs von Objekt A nicht kleiner als die Ausprägung des korrespondierenden Attributs von Objekt B sein, aber durchaus größer, dann würde für den Paarvergleiche "A mit B" (in Bezug auf das entsprechende Attribut) zwangsläufig ein anderer Wert resultieren als für "B mit A". Das folgende Beispiel illustriert diesen Sachverhalt: Das erste Objekt darf nicht schwerer sein als das zweite. Ist A leichter als B, so ordnet der Vergleich "A mit B" der Gewichtsübereinstimmung einen numerischen Wert zwischen 0 und 1 zu, während der Vergleich "B mit A" zu einem Scheitern des gesamten Vergleichs führt, da B schwerer ist als A.

2) Vgl. Anhang, S. 111 ff.

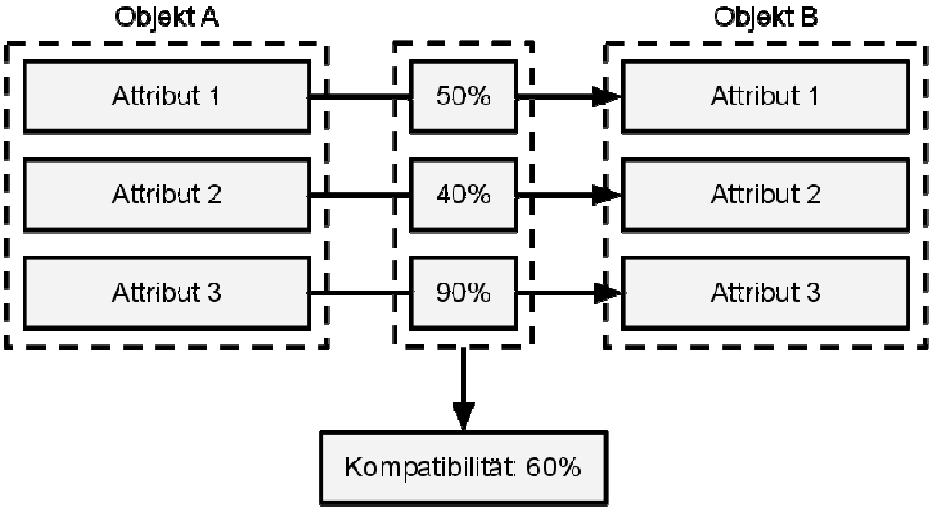

Abbildung 10: Darstellung der Matching-API

Die Matching-API wird genutzt, um unter einem betrachteten Frachtrauminserat kompatible Laderauminserate anzuzeigen.[1]

Die mithilfe der Matching-API ermittelten Kompatibilitätswerte werden dazu in einer dedizierten Datenbanktabelle vorgehalten, um diese nicht bei jedem Betrachten eines Inserats neu kalkulieren zu müssen.[2]

7.2.3.2 Such-API zum Abfragen des Datenbestands

Das Auffinden bestehender Angebote[3] soll durch eine Suche erleichtert werden.[4]

Es werden sowohl eine Volltextsuche als auch eine erweiterte Suche implementiert. Die Suchalgorithmen sind dabei nicht fest in die Applikation integriert, sondern werden über eine Schnittstelle – die Such-API – aufgerufen.

Abbildung 11 stellt dar, wie die Such-API hierzu mit anderen Komponenten interagiert.

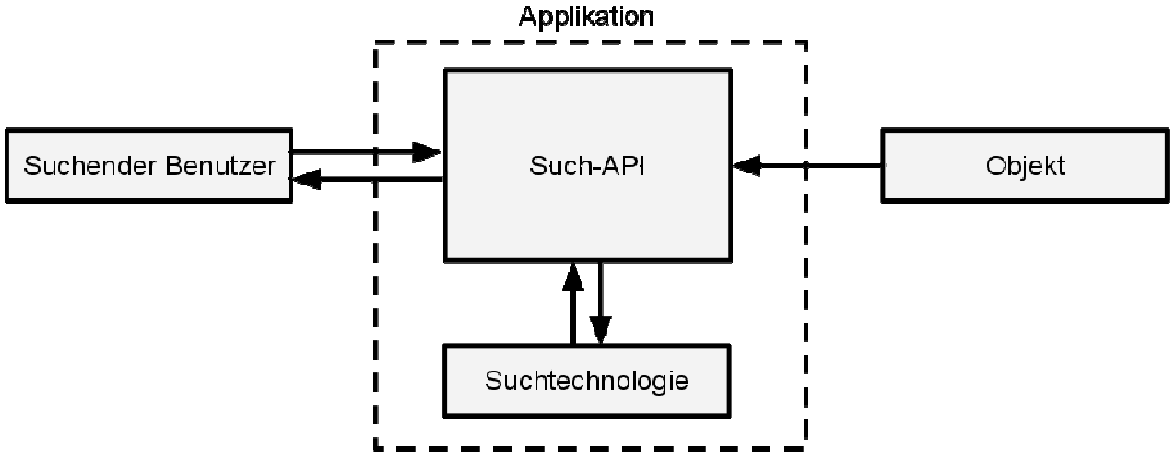

Abbildung 11: Darstellung der Such-API

---

1) Analog werden unter einem Laderauminserat kompatible Frachtrauminserate angezeigt.

2) Zur Struktur dieser Datenbanktabelle vgl. Anhang, S. 88.

3) Alle Inserate im System der Online-Frachtenbörse sind Angebote. Es wird nicht möglich sein, Inserate als Anfragen einzustellen.

4) Vgl. Klippert/Kowalski/Bruns (2010), S. 74.

Die Such-API definiert hierzu folgende abstrakte Operationen, die von der benutzten Suchtechnologie unterstützt werden müssen:[1]

- die Informationen über ein Objekt aus dem Suchindex zu entfernen,
- die Informationen eines Objekt im Suchindex zu aktualisieren,
- die Anzahl der Resultate einer Suchanfrage zurückzugeben (gegebenenfalls nach Objektarten gefiltert), sowie
- die Resultate einer Suchanfrage zurückzugeben (gegebenenfalls nach Objektarten gefiltert).

Sobald ein Objekt erstellt, bearbeitet oder gelöscht wird, wird der korrespondierende Eintrag im Suchindex erstellt, angepasst oder entfernt.[2] Die Applikation übermittelt der Such-API hierzu die Daten des zu indizierenden Objekts (Art und ID des Objekts sowie die zu dem Objekt gehörenden Schlagworte). Inserate übergeben zudem alle Informationen für alle Sprachen lokalisiert inklusive der vom Benutzer eingegebenen lokalisierten Informationen.[3] Dies führt dazu, dass eine Suche nach „Rotterdam hazmat single wagon" dieselben Ergebnisse liefert wie eine Suche nach „Rotterdam Gefahrengut Einzelwagen".

In der vorliegenden Implementierung des Suchalgorithmus wird das übergebene Objekt unverändert in eine Datenbanktabelle mit Suchobjekten gespeichert.

Tabelle 9 zeigt die Struktur der zugehörigen Datenbanktabelle.

Name	Typ	Beschreibung	Beispiel
item_type	String	Verweis auf den Typ des indizierten Objekts	„Freight"
item_id	Integer	Verweis auf die ID des indizierten Objekts	1
text	Text	Text mit allen für das Objekt relevanten Schlagwörtern	„Rotterdam Gefahrengut..."

Tabelle 9: Struktur der „simple_searches"-Datenbanktabelle

Der Suchalgorithmus vergleicht die Suchanfrage mit der Textspalte dieser Tabelle.[4]

7.2.3.3 XML/JSON-API zum Zugriff durch Drittanbieter

Ein Ziel des Softwareprototyps besteht darin, den Zugriff auf die Funktionalitäten des Systems der Online-Frachtenbörse auch über Drittanbietersoftware zu ermöglichen.[5]

1) Für eine technischere Spezifikation vgl. Anhang, S. 122 ff.

2) Dies geschieht nicht nur für Fracht- und Laderauminserate, sondern auch für diverse andere Objekte, wie bspw. Benutzer, Personen und Unternehmen.

3) Gemeint sind hier die natürlichsprachlichen lokalisierbaren Attribute von Inseraten.

4) Dieses Vorgehen ist gleichermaßen effizient und ineffektiv.

5) Vgl. Klippert/Kowalski/Bruns (2010), S. 74 sowie S. 84.

Aus diesem Grund wird eine Schnittstelle geschaffen, über welche die gesamte Applikation gesteuert werden kann. Sie kommuniziert nach dem REST-Ansatz über HTTP in den Formaten XML und JSON.[1]

Diese XML/JSON-API ist damit ein Webservice und dient als Schicht über den Funktionen der Applikation.[2] Sie stellt eine Schnittstelle für die Maschine-Maschine-Interaktion dar, analog zur Benutzeroberfläche, die eine Schnittstelle zur Mensch-Maschine-Interaktion darstellt.

Abbildung 12 stellt dar, wie eine Drittanbietersoftware über die XML/JSON-API (im Vergleich zu einem menschlichen Benutzer über die Benutzeroberfläche) mit der Applikation interagiert.

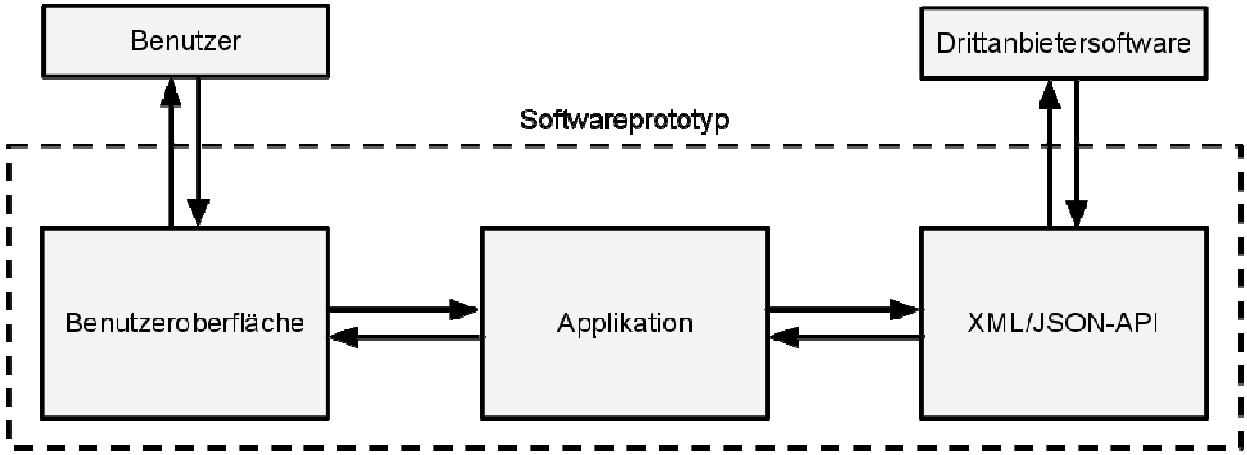

Abbildung 12: Darstellung der XML/JSON-API

Da die XML/JSON-API technisch gesehen nur eine Abstraktion der Interaktion über die Benutzeroberfläche darstellt, sind die zugrundeliegende Logik, bspw. zur Erstellung von Inseraten oder Bewertungen, das Rechtesystem und die Protokollierung des Benutzerverhaltens von der Implementierung dieser Schnittstelle unabhängig.

Es entsteht somit durch die Existenz der XML/JSON-API kein Mehraufwand bei der Implementierung neuer Funktionen in den Software-Prototyp.[3]

7.2.4 Rechtesystem

Das Rechtesystem definiert Benutzerrollen und verknüpft sie mit bestimmten Berechtigungen, um so den Zugriff auf Informationen und Funktionen zu beschränken und betriebliche Funktionen im System der Online-Frachtenbörse abzubilden.[4]

---

1) Zum REST-Ansatz und seiner Nutzung von HTTP vgl. Chak (2009), S. 267 ff. Der Einsatz von Standards wird in der Literatur als Anforderung identifiziert, vgl. Klippert/Kowalski/Bruns (2010), S. 57, S. 58 f. sowie S. 84. XML und JSON sind industrietypische Standards bei der Implementierung von Webservices und ermöglichen die Implementierung offener, vollkompatibler Schnittstellen, vgl. Richardson/Ruby (2007), S. 44 ff.

2) Vgl. Ferrara/MacDonald (2002), S. 2 f.

3) Würden bspw. die Bewertungen von Geschäftspartnern um eine quantitative Komponente in Form des Vergebens von 1 bis 5 Sternen erweitert, so müsste eine Drittanbietersoftware die Anzahl der Sterne in ihren Anfragen zur Erstellung einer neuen Bewertung hinzufügen. In der Benutzeroberfläche des Softwareprototyps müsste ein zusätzliches grafisches Kontrollelement aufgenommen werden, mit dem beim Erstellen einer neuen Bewertung die Anzahl der Sterne ausgewählt werden kann. Aber innerhalb der Applikation müsste die Anforderung "Sterne empfangen" und die Aktion "Sterne in Bewertung speichern" nur einmal implementiert werden.

4) Vgl. Klippert/Kowalski/Bruns (2010), S. 56.

Berechtigungen und Benutzerrollen spielen in jeder größeren Applikation, wie bspw. einer Online-Frachtenbörse, eine entscheidende Rolle.[1]

Wie in Abbildung 7 dargestellt wurde[2], werden bestimmte Aktionen der Benutzer innerhalb des Systems der Online-Frachtenbörse von den ihnen aufgrund ihrer Benutzerrolle erteilten Berechtigungen beeinflusst.

Benutzerrollen haben keine dichotome Ausprägung in Bezug auf die Berechtigung zur Interaktion mit bestimmen Teilen der Applikation, sondern definieren für jeden Benutzertyp explizit, welche Operationen auf welchen Objekten innerhalb der Applikation erlaubt und verboten sind.[3] Hierdurch lässt sich bspw. festlegen, dass ein Benutzer mit der Benutzerrolle „Praktikant" zwar Bewertungen lesen und anschließend veröffentlichen darf, jedoch keine bereits freigeschalteten Bewertungen löschen kann.[4] Darüber hinaus ermöglicht das Rechtesystem des Softwareprototyps, dass der zuständige Mitarbeiter eines Anbieters neue Benutzerkonten anlegen und diesen Benutzern wiederum Benutzerrollen zuordnen kann.[5] Jeder Benutzer kann beliebig viele Benutzerrollen besitzen. Die mit den Benutzerrollen einhergehenden Berechtigungen werden nicht in dem Benutzerrollenobjekt definiert, sondern es wird innerhalb der Applikation für alle Aktionen definiert, welche Benutzerrolle benötigt wird, um die jeweilige Aktion ausführen zu dürfen. Ein Benutzerrollenobjekt besteht daher nur aus dem Namen der Rolle. Eine weitere Datenbanktabelle bildet ab, welcher Benutzer welche Benutzerrollen besitzt.[6]

Des Weiteren ermöglicht das Vorhandensein von Benutzerrollen die Präsentation einer an die Tätigkeiten des Benutzers angepassten Benutzeroberfläche.[7]

Das Rechtesystem des Softwareprototyps bestimmt also die Zugriffsmöglichkeiten eines Benutzers auf Informationen sowie die Präsentation seiner Benutzeroberfläche.[8] Darüber hinaus wäre ein derartiges Rechtesystem auch geeignet, potentiellen Teilnehmern schrittweise Zugang zum System der Online-Frachtenbörse zu gewähren.[9]

---

1) Vgl. Mauthe/Thomas (2004), S. 248.

2) Vgl. Abbildung 7, S. 36.

3) Auch wenn in der Literatur teilweise nur von der Unterscheidung zwischen Schreib- und Leserechten ausgegangen wird (vgl. Mauthe/Thomas (2004), S. 248), sind hier Berechtigungen zum "Erstellen", "Lesen", "Verändern" und "Löschen" von Objekten gemeint.

4) Ein Rechtesystem könnte prinzipiell noch komplexer strukturiert sein und neben Benutzertypen auch Rechte für individuelle Benutzer verwalten, vgl. Mauthe/Thomas (2004), S. 248. Im Rahmen einer Online-Frachtenbörse erscheint die Implementierung eines typenbasierten Systems ausreichend, da die antizipierten homogenen Benutzergruppen gut voneinander abgegrenzt werden können (bspw. in die Typen "Benutzer" und "Administratoren").

5) Zur Benutzerverwaltung innerhalb eines Unternehmens vgl. S. 57.

6) Zur Struktur der Datenbanktabellen für Benutzerrollen sowie der Benutzerrollen-Benutzer-Zuordnung vgl. Anhang, S. 88 f.

7) Vgl. Mauthe/Thomas (2004), S. 248 sowie S. 55.

8) Vgl. Mauthe/Thomas (2004), S. 248.

9) Vgl. Klippert/Kowalski/Bruns (2010), S. 48 f. sowie S. 62.

Im Folgenden wird bei der Beschreibung von Funktionen des Softwareprototyps, sofern nicht anders angegeben, stets davon ausgegangen, dass der beschriebene Benutzer alle erforderlichen Benutzerrollen besitzt, um die dargestellte Funktion auszuführen.[1]

7.2.5 Protokollierung des Benutzerverhaltens

Im Rahmen der Protokollierung des Benutzerverhaltens werden alle Aktionen der Benutzer im System der Online-Frachtenbörse möglichst vollständig gespeichert.[2]

Die Protokollierung des Benutzerverhaltens ist eine zentrale Funktionalität des Softwareprototyps, da fundierte Aussagen über das Verhalten der Benutzer innerhalb des Systems der Online-Frachtenbörse nur durch die möglichst vollständige Speicherung aller Aktionen der Benutzer sowie deren nachträgliche statistische Auswertungen möglich sind. Des Weiteren können die zuständigen Mitarbeiter eines Unternehmens und Betreiber der Online-Frachtenbörse nur so das Verhalten ihrer Benutzer nachvollziehen.[3]

Abbildung 13 stellt dar, wie die Interaktion mit dem Softwareprototyp zur Speicherung eines Protokolleintrags führt. Dabei ist es irrelevant, ob der Benutzer mit dem Softwareprototyp über die Benutzeroberfläche oder die XML-JSON-API interagiert.

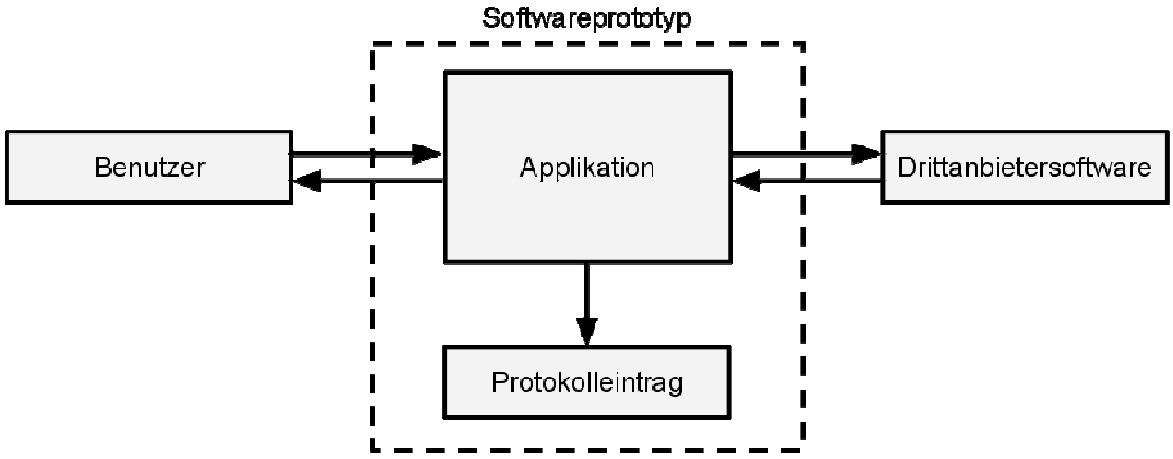

Abbildung 13: Darstellung des Ablaufs zur Erstellung eines Protokolleintrags

7.2.5.1 Protokollierung von Aktionen auf Objekten

Die Protokollierung der Benutzeraktionen auf Objekten geschieht mit dem Ziel, Antworten auf folgende Fragestellungen zu finden:

- Wieviele Benutzer sind im System der Online-Frachtenbörse aktiv?

---

1) Dies impliziert, dass der Benutzer im System der Online-Frachtenbörse registriert und angemeldet sein muss, um auf Informationen zugreifen zu können, die durch das System der Online-Frachtenbörse und seine Benutzer bereitgestellt werden, da nur so gewährleistet werden kann, dass diese Informationen keinem Dritten zugänglich sind, vgl. Klippert/Kowalski/Bruns (2010), S. 79.

2) "Möglichst vollständig" bedeutet hier, dass alle Veränderungen an Objekten sowie alle Suchanfragen berücksichtigt werden. Einige Benutzeraktionen werden zum gegenwärtigen Zeitpunkt jedoch nur implizit dokumentiert. So werden bspw. das Aufrufen des Kontaktformulars durch eingehende E-Mails der Benutzer und die Speicherung der Serveranfrage in den Server-Logs dokumentiert. Sowohl die Datenbank als auch die Server-Logs können aus technischer Sicht zur Datenanalyse herangezogen werden (vgl. Klippert/Kowalski/Bruns (2010), S. 56). Diese Quellen sind jedoch zum Zeitpunkt der Erstellung dieses Projektberichts nicht Gegenstand der vom System durchgeführten und im Backend abrufbaren Nutzungsstatistiken (vgl. S. 56).

3) Vgl. Klippert/Kowalski/Bruns (2010), S. 81.

- Welche Benutzer sind im System der Online-Frachtenbörse aktiv?
- Zu welchen Uhrzeiten sind die Benutzer im System der Online-Frachtenbörse aktiv?
- Welche Aktionen führen die Benutzer durch?
- Welcher Anteil an Interaktion im System der Online-Frachtenbörse entfällt auf welche Objekte?

Um Antworten auf diese Fragestellungen finden zu können, erzeugt die Applikation, wenn ein registrierter und im System der Online-Frachtenbörse angemeldeter Benutzer Objekte innerhalb der Online-Frachtenbörse erstellt, verändert oder löscht, für diese Aktion einen Protokolleintrag in der Datenbank. Der gespeicherte Protokolleintrag enthält Informationen über den Benutzer sowie den Zeitpunkt, die Art und die Auswirkungen der durchgeführten Aktion.

Tabelle 10 zeigt die Struktur der zugehörigen Datenbanktabelle.

Name	Typ	Beschreibung	Beispiel
item_type	String	Verweis auf den Typ des veränderten Objekts	„Freight"
item_id	Integer	Verweis auf die ID des veränderten Objekts	1
action	String	Art der ausgeführten Aktion	create
diff	Text	Zusammenfassung aller getätigten Änderungen in Form eines serialisierten Hash-Objekts	„{weight: [50, 60]}"
user_id	Integer	Verweis auf den Benutzer, dem das Objekt gehört	1
company_id	Integer	Verweis auf das Unternehmen, zu der das Objekt gehört	1

Tabelle 10: Struktur der „action_recordings"-Datenbanktabelle

Hierdurch werden die Abbildung der Aktivitäten aller Benutzer sowie detaillierte Analysen, bspw. über Benutzer, Objekte und Zeitpunkte, möglich.[1]

7.2.5.2 Protokollierung von Suchanfragen

Zusätzlich werden alle Suchanfragen der Benutzer protokolliert.

Die Protokollierung der Suchanfragen geschieht mit dem Ziel, Antworten auf folgende Fragestellungen zu finden:

- Wieviele Benutzer suchen im System der Online-Frachtenbörse?
- Welche Benutzer suchen im System der Online-Frachtenbörse?
- Zu welchen Uhrzeiten suchen die Benutzer?
- Welche Benutzer suchen nach welchen Schlüsselwörtern?
- Wieviele Resultate lieferte die jeweilige Suche?
- Wieviele Inserate betrachten die Benutzer nach einer Suche?
- Welche Inserate betrachten die Benutzer nach einer Suche?

1) Die Darstellungen der Benutzeraktivitäten im Frontend und Backend sowie die Nutzungsstatistiken basieren auf diesen Protokolleinträgen.

Um Antworten auf diese Fragestellungen finden zu können, erzeugt die Applikation, wenn ein registrierter und im System der Online-Frachtenbörse angemeldeter Benutzer eine Suchanfrage an den Server richtet, für diese Aktion einen Protokolleintrag in der Datenbank. Der gespeicherte Protokolleintrag enthält Informationen über den Benutzer sowie den Zeitpunkt, die Suchbegriffe und die Anzahl der Resultate.

Klickt ein Benutzer im Rahmen einer Suchanfrage auf ein Resultat, wird ebenfalls ein Protokolleintrag geschrieben. Dieser Protokolleintrag enthält Informationen über das angeklickte Resultat sowie den zu der durchgeführten Suchanfrage gehörenden Protokolleintrag. So wird festgehalten, wieviele Resultate eine Suchanfrage liefert und welche Resultate anschließend vom Benutzer angeklickt werden.

Mit den auf diese Art gespeicherten Informationen lassen sich die zuvor genannten Fragen im Rahmen der statistischen Auswertung beantworten.[1]

Tabelle 11 zeigt die Struktur der zugehörigen Datenbanktabelle.

Name	Typ	Beschreibung	Beispiel
user_id	Integer	Verweis auf den Benutzer, dem das Objekt gehört	1
query	String	Suchbegriffe	„Rotterdam Gefahrengut"
results	Integer	Anzahl der Resultate	12
parent_id	Integer	Verweis auf das Elternobjekt	1
result_type	String	Verweis auf den Typ des Resultats	„Freight"
result_id	Integer	Verweis auf die ID des Resultats	1

Tabelle 11: Struktur der „search_recordings"-Datenbanktabelle

7.2.6 Roboter

Ein Roboter ist eine Applikation, die dazu entwickelt wurde, ein Medium, bspw. eine Webapplikation, wie ein menschlicher Benutzer zu nutzen.[2]

Analog dazu ist ein Roboter im Rahmen der Online-Frachtenbörse ein Skript, das autonom mit dem Software-Prototyp interagiert und dabei versucht, das Verhalten eines menschlichen Benutzers zu imitieren. Der Roboter interagiert dabei wie eine Drittanbietersoftware über die XML/JSON-API mit der Applikation anstatt wie ein Benutzer über die Benutzeroberfläche.

Der Einsatz von Robotern dient dem Ziel, im Rahmen der Anforderungsanalyse bestimmte Verhaltensweisen der Benutzer zu testen.[3]

---

1) Die Auswertung findet durch die Administratoren der Online-Frachtenbörse über das Backend statt.

2) Vgl. Heaton (2008), S. 333.

3) Zudem erscheint es plausibel, dass sich die Suchfunktion mit dynamisch generierten Inseraten verlässlicher testen lässt als mit statisch hinterlegten Testdaten, da die Variabilität dynamischer Inhalte Probleme aufzeigen kann, die bei Verwendung von stets gleichen Daten nicht entdeckt würden.

Die Entwicklung und der Einsatz von Robotern verfolgen dazu folgende Teilziele:

- Es soll soziale Interaktion im System der Online-Frachtenbörse simuliert werden, um diese für menschliche Tester authentischer wirken zu lassen. Hierzu soll nicht ein vorbereiteter Satz statischer Inhalte geladen werden, sondern neue, dynamische Inhalte sollen kontinuierlich bereitgestellt werden können.
- Die Datenbank soll mit aktuellen Inseraten gefüllt werden, die eine Berechnungsgrundlage für die Such- und Matching-API bilden.
- Die Ansichten des Frontends sollen mit aktuellen Inseraten und Bewertungen gefüllt werden, um so die Verwaltung von Inseraten und Bewertungen testen zu können.
- Die Ansichten des Backends sollen mit Benutzern und Aktivitäten aller Art gefüllt werden, um so die Benutzerverwaltung und Auswertungsmöglichkeiten der Nutzungsstatistiken testen zu können.
- Der Softwareprototyp soll durch die zahlreichen asynchronen Anfragen von Robotern einem Lasttest unterzogen werden können, um so mögliche Defizite in der Implementierung aufzudecken, wie bspw. redundante oder besonders komplexe Datenbankabfragen.

Ein Roboter führt zur Erreichung dieser Teilziele folgende Aktionen durch:

- Er inseriert Fracht- oder Laderaum.[1]
- Er löscht Inserate.
- Er sucht passende Inserate zu seinen Inseraten.
- Er erzeugt Benutzer innerhalb seines Unternehmens.
- Er löscht Benutzer innerhalb seines Unternehmens.
- Er bewertet andere Teilnehmer in der Online-Frachtenbörse.
- Er schaltet Bewertungen von anderen Teilnehmern frei.

Bei der „Planung" dieser Aktionen greift der Roboter jedoch nicht auf komplexe Algorithmen, wie bspw. neuronale Netzwerke oder Methoden des Case-Based-Reasoning zurück, sondern wählt zufallsbedingt eine der genannten Aktionen und entscheidet anschließend anhand einfacher Heuristiken, ob und wie er diese Aktion ausführt.[2]

7.2.7 Hinterlegung von Kontaktinformationen

Benutzer können sowohl zu ihrer Person als auch zu ihrem Unternehmen Kontaktinformationen im System der Online-Frachtenbörse hinterlegen.[3]

Das Einstellen möglichst vollständiger Kontaktinformationen wird als erfolgskritischer Faktor beim Betrieb einer Frachtenbörse angesehen, da die tatsächliche Kontaktanbahnung außerhalb des Systems der Online-Frachtenbörse stattfinden muss.[4] Der Softwareprototyp unterstützt dieses Ziel, indem er die Hinterlegung dieser Informationen fördert.[5]

---

1) Das Inserieren von Fracht- bzw. Laderaum ist abhängig davon, welche Rolle dem Benutzerobjekt des Roboters über das Attribut "posting_type" zugewiesen ist.

2) Bspw. wird beim Erstellen neuer Benutzerkonten darauf geachtet, dass deren Anzahl innerhalb eines Unternehmens bestimmte Höchst- und Mindestgrenzen einhält. Ebenso achtet der Roboter darauf, dass er trotz der Aufgabe, Inserate zu löschen, immer eine bestimmte Mindestanzahl an Inseraten bestehen lässt.

3) Gemeint sind hier bspw. Adress- und Telefondaten sowie E-Mail- und Internetadressen.

4) Vgl. Klippert/Kowalski/Bruns (2010), S. 53, S. 55 sowie S. 77.

5) Vgl. S. 53.

Obwohl die Datenstrukturen zur Hinterlegung von Kontaktinformationen für Unternehmen und Personen teilweise identisch sind, wurden die betreffenden Datenbankfelder redundant in den jeweiligen Datenbanktabellen implementiert, anstatt eine separate Datenbanktabelle für Kontaktinformationen anzulegen.[1] Ebenso wichtig erscheint das Vorhandensein eines konkreten Ansprechpartners. Hierzu gibt es zum einen die Möglichkeit, einen allgemeinen Ansprechpartner auf dem Anbieterprofil zu hinterlegen, und zum anderen kann für jedes Inserat ein individueller Ansprechpartner angegeben werden.

7.2.8 Bewertung von Geschäftspartnern

Benutzer können im System der Online-Frachtenbörse ihre Geschäftspartner bewerten.[2]

Es können nur positive Bewertungen in Form eines „Daumen hoch" und eines Begleittextes abgegeben werden. Hierzu schreibt ein Benutzer einen bewertenden Text über einen Geschäftspartner und dieser erhält die Möglichkeit, diese Bewertung freizuschalten und den Text so auf seinem Anbieterprofil zu veröffentlichen.

Das Nichtzulassen negativer Bewertungen begründet sich in einem antizipiert objektiveren Gesamtbild zu der Qualität der Anbieter.[3]

Auf die Implementierung eines mehrstufigen quantitativen Bewertungssystems wird verzichtet, da im Rahmen der Implementierung eines Softwareprototyps die qualitative, natürlichsprachliche Bewertung ausreichend erscheint, um die grundlegenden Strukturen eines leicht erweiterbaren Bewertungssystems aufzuzeigen und zu implementieren.[4] Dadurch werden die Ziele der schlanken Prozessabbildung und leichten Erweiterbarkeit unterstützt.

Ein Bewertungsobjekt besteht somit nur aus einem Text sowie Informationen über den bewertenden Benutzer, den bewerteten Anbieter und den veröffentlichenden Benutzer.

Tabelle 12 zeigt die Struktur der zugehörigen Datenbanktabelle.

---

1) Der Grund hierfür ist, dass die Datenstrukturen, im Gegensatz zu den in Kapitel 7.2.2 besprochenen Ortsobjekten, nur teilweise identisch sind.

2) Vgl. Klippert/Kowalski/Bruns (2010), S. 80.

3) Begründet wird dies durch folgende Plausibilitätsüberlegung: In einem Vote-Up-Down-System, bei dem sowohl positive als auch negative Stimmen abgegeben werden können, würden diese gegeneinander aufgerechnet. Das heisst, hat ein Anbieter drei positive (+1) und drei negative Bewertungen (-1), so hat er im Schnitt eine neutrale Wertung (0). Es erscheint allerdings plausibel, dass von all seinen zufriedenen Kunden nur ein unbekannter Bruchteil eine Bewertung ins System eingetragen hat, während verärgerte Kunden sehr viel eher nach Möglichkeiten suchen, ihren Geschäftspartner für das aus ihrer Sicht misslungene Geschäft nachträglich zu sanktionieren. Es wäre also der Fall denkbar, dass ein Anbieter von 12 zufriedenen Kunden nur drei positive Bewertungen erhält, von seinen vier unzufriedenen Kunden sich jedoch ebenfalls drei zu einer negativen Bewertung entschließen. Lässt das System nur positive Bewertungen zu (Vote-Up-System), so können positive nicht von negativen Bewertungen "neutralisiert" werden. In einem Vote-Up-Down-System würde ein Anbieter mit neutraler Bewertung (0) bspw. 50 positive und 50 negative Bewertungen haben und damit nach außen genauso bewertet sein, wie ein Anbieter mit einer positiven und einer negativen Bewertung. In einem Vote-Up-System hätte der erstgenannte Anbieter 50 Bewertungen und der zweite nur eine. Es wäre bereits hieraus abzuleiten, dass der erste Anbieter scheinbar länger und aktiver an der Frachtenbörse teilnimmt als der zweite.

4) In späteren Iterationen wäre die Implementierung eines solchen mehrstufigen, quantitativen Bewertungssystems durchaus denkbar, um hierdurch zusätzlich zum Kontaktformular Benutzerfeedback über die Qualität der Anbieter im System der Online-Frachtenbörse zu generieren, vgl. Klippert/Kowalski/Bruns (2010), S. 41. Zusätzlich kann ein Bewertungssystem die Transparenz der Online-Frachtenbörse steigern, vgl. Klippert/Kowalski/Bruns (2010), S. 49.

Name	Typ	Beschreibung	Beispiel
author_user_id	Integer	Verweis auf den Benutzer, der die Bewertung erstellt hat	1
author_company_id	Integer	Verweis auf das Unternehmen, deren Benutzer die Bewertung erstellt hat	1
approved_by_id	Integer	Verweis auf den Benutzer, der die Bewertung freigegeben hat	2
company_id	Integer	Verweis auf das Unternehmen, zu der die Bewertung gehört	1
text	Text	Text der Bewertung	„Danke für die..."

Tabelle 12: Struktur der „reviews"-Datenbanktabelle

7.2.9 Feedback an die Betreiber

Die Benutzer sollen die Möglichkeit erhalten, sich gegenüber den Betreibern der Online-Frachtenbörse kritisch zu äußern.[1]

Neben den bereits vorgestellten Bewertungen, die ebenfalls zur Feedbackanalyse herangezogen werden können,[2] wird ein Kontaktformular implementiert, über das die Benutzer sich mit einem Freitext zur Online-Frachtenbörse äußern können.[3]

Der Benutzer erhält die Möglichkeit, seinen Text in einem Formular des Frontends einzugeben und abzuschicken.[4] Abbildung 14 zeigt dieses Formular.

1) Vgl. Klippert/Kowalski/Bruns (2010), S. 81.

2) Vgl. Klippert/Kowalski/Bruns (2010), S. 41.

3) Umfangreichere Feedbacksysteme in Form von Online-Fragebögen mit geschlossenen oder halboffenen Fragen erscheinen vor dem Hintergrund des Primärziels dieses Projektberichts zu umfangreich und vor allem zu zeitaufwendig in der Implementierung.

4) Da die hierfür benötigte Funktionalität des Frontends vergleichsweise trivial ist (es wird ein Textfeld zur Verfügung gestellt, über welches vom Benutzer ein Freitext an die Betreiber der Online-Frachtenbörse verschickt werden kann, und anschließend wird eine Erfolgsmeldung über den Versand angezeigt), wird diese Funktionalität bei den Funktionalitäten des Frontends nicht gesondert behandelt.

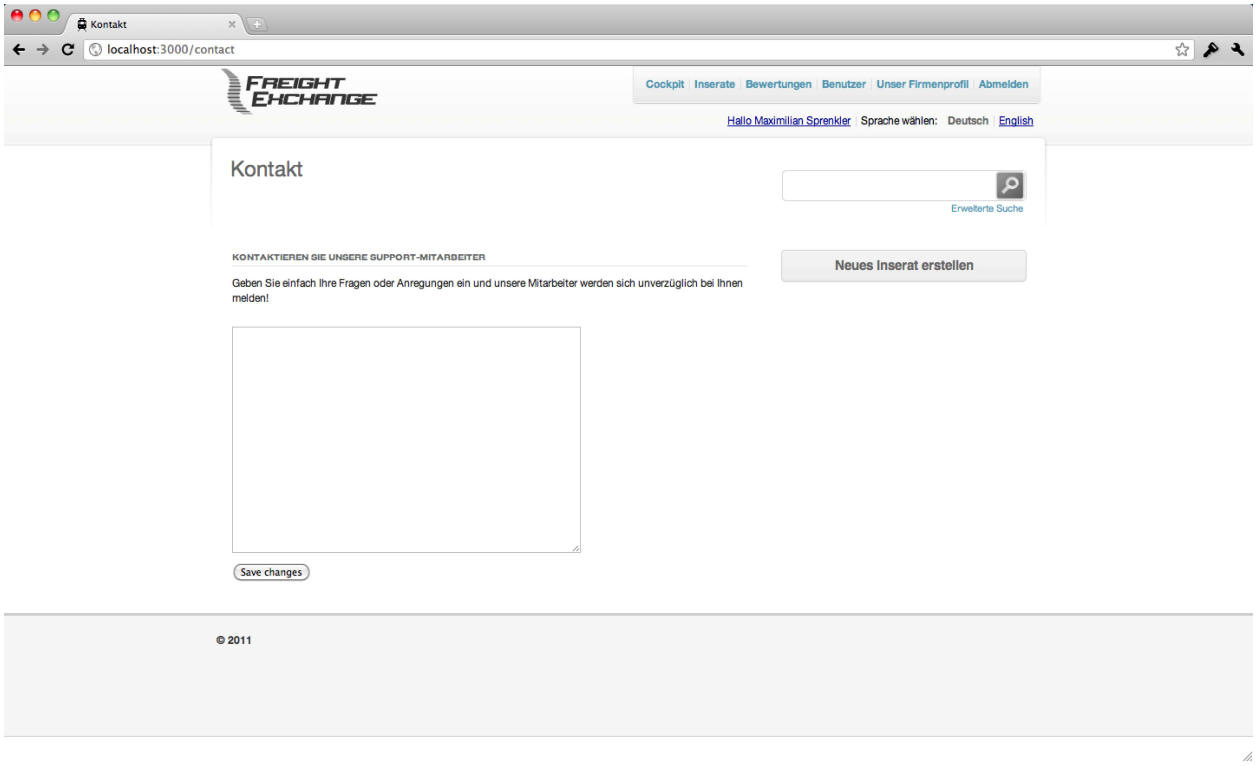

Abbildung 14: Screenshot des Kontaktformulars

Die Applikation generiert aus dem übermittelten Text eine E-Mail und sendet diese an eine über das Backend konfigurierbare E-Mail-Adresse.[1] In der E-Mail werden unter der Nachricht des Benutzers sowohl Links auf die Profile des Benutzers und seines Unternehmens als auch seine Kontaktinformationen angezeigt, um eine Kontaktaufnahme durch die Betreiber der Online-Frachtenbörse so einfach wie möglich zu gestalten, damit diese eine persönliche Betreuung der Benutzer gewährleisten können.[2]

Es wurde darauf verzichtet, einen vorgegebenen Satz an Betreffzeilen zu integrieren, wie bspw. „Frage an die Betreiber" oder „Technisches Problem", da es plausibel erscheint, dass diese antizipierten Themengebiete den später von realen Benutzern angesprochenen Themen nicht immer entsprechen werden. Des Weiteren scheint eine Vorabkategorisierung von Kontaktanfragen primär der Komplexitätsreduktion der Aufgaben von Supportmitarbeitern zu dienen. Es erscheint zu diesem Zeitpunkt zielführender, komplett auf die Kategorisierung von Kontaktanfragen zu verzichten und im späteren Verlauf der Anforderungsanalyse tatsächliche Kategorien zu identifizieren, in welche die Kontaktanfragen sinnvoll aufgeteilt werden können.

7.3 Funktionen des Frontends

7.3.1 Verwaltung von Inseraten

Das Frontend ermöglicht das Erstellen, Verwalten und Wiederverwenden von Inseraten.

---

1) Die E-Mail-Adresse kann über die Variablenkonfiguration des Backends geändert werden, vgl. S. 54.

2) Die persönliche Betreuung wird von der Literatur als potentieller Erfolgsfaktor einer Online-Frachtenbörse identifiziert, vgl. Klippert/Kowalski/Bruns (2010), S. 55. Es erscheint plausibel, dass die zeitnahe, adäquate, persönliche Betreuung der Benutzer auch im Rahmen der Anforderungsanalyse einen Erfolgsfaktor darstellt, insbesondere wenn bspw. projektfremde Tester nicht persönlich, sondern aus der Ferne betreut werden.

Eine Übersicht zeigt dem Benutzer alle Inserate seines Unternehmens und gibt ihm die Möglichkeit, neue Inserate zu erstellen sowie bestehende Inserate zu bearbeiten, zu löschen oder als Vorlage für neue Inserate zu verwenden. Abbildung 15 zeigt diese Übersicht.

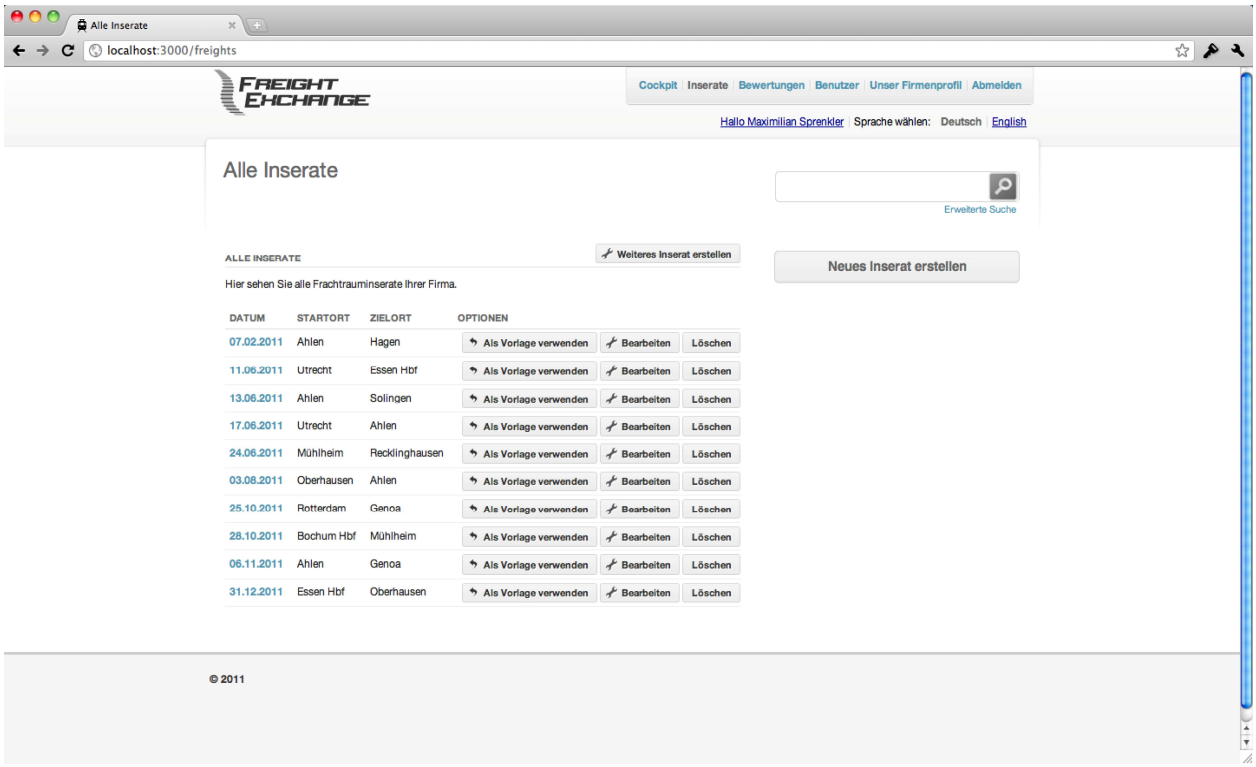

Abbildung 15: Screenshot der Inserateübersicht eines Unternehmens

Ein Inserat wird in der gleichen Eingabemaske bearbeitet, in der es auch angelegt wurde. Abbildung 16 zeigt diese Eingabemaske.

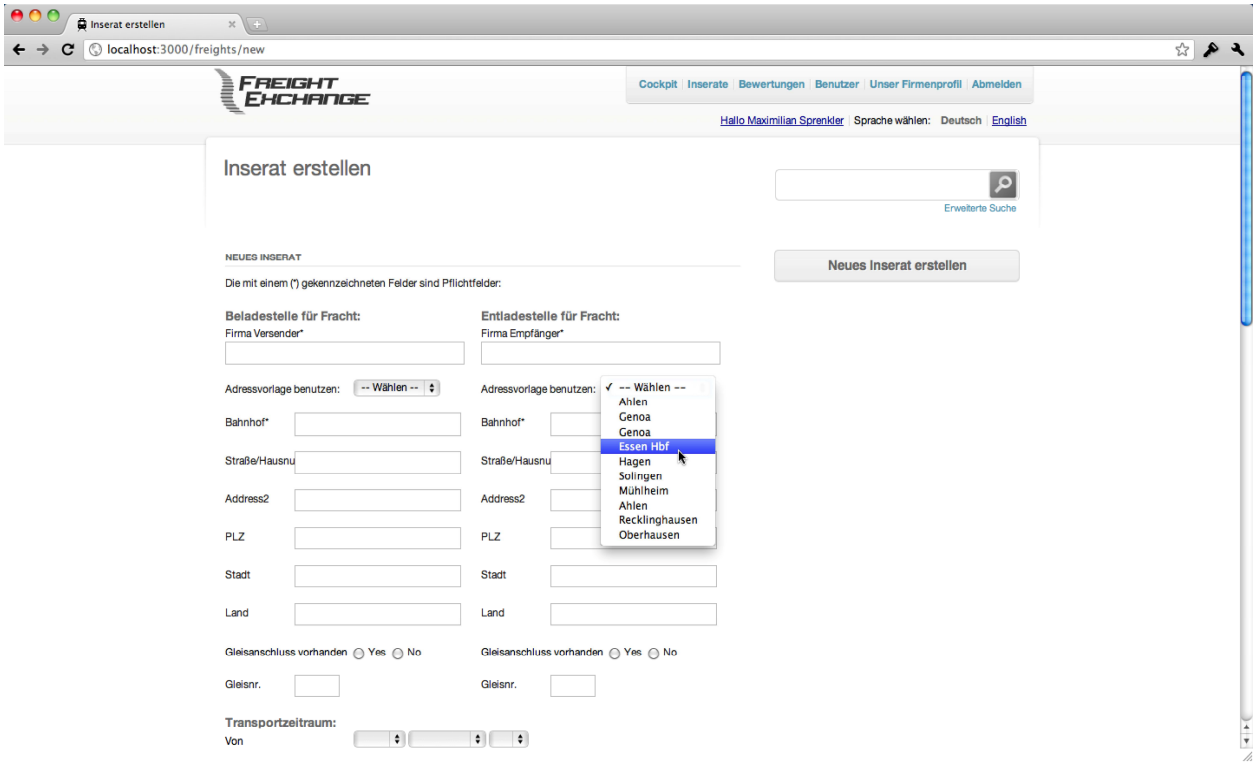

Abbildung 16: Screenshot der Eingabemaske für ein Inserat

Die Benutzeroberfläche ermöglicht dem Benutzer, Start- und Zielorte vorangegangener Inserate wiederzuverwenden, um so den Zeitaufwand redundanter Dateneingaben sowie die Gefahr von Eingabefehlern zu reduzieren.[1] Hierzu können die Orte vergangener Inserate für Start- und Zielort aus einer Liste ausgewählt werden.[2] Dies dient dem Ziel, die Komplexität von Eingabemasken und Prozessen zu reduzieren.

Während der Erstellung eines Inserats ermöglicht die Benutzeroberfläche die simultane Eingabe vom Benutzer lokalisierter Informationen.[3] Hierzu kann der Benutzer zu einem Eingabefeld für bspw. „Sonstige Informationen" mittels einer Auswahlliste bestimmen, in welchen Sprachen er den betreffenden Text hinterlegen möchte. So können auch die qualitativen und damit natürlichsprachlich zu beschreibenden Informationen eines Inserats leicht durch den Benutzer selbst lokalisiert werden.

Im Anschluss an die Erstellung oder Bearbeitung eines Inserats wird der Benutzer auf die Inseratsseite weitergeleitet, auf der er unter seinem eigenen Inserat passende Inserate anderer Anbieter findet, die ihm als mögliche Geschäftspartner vorgeschlagen werden.[4] Abbildung 17 zeigt eine solche Ansicht.

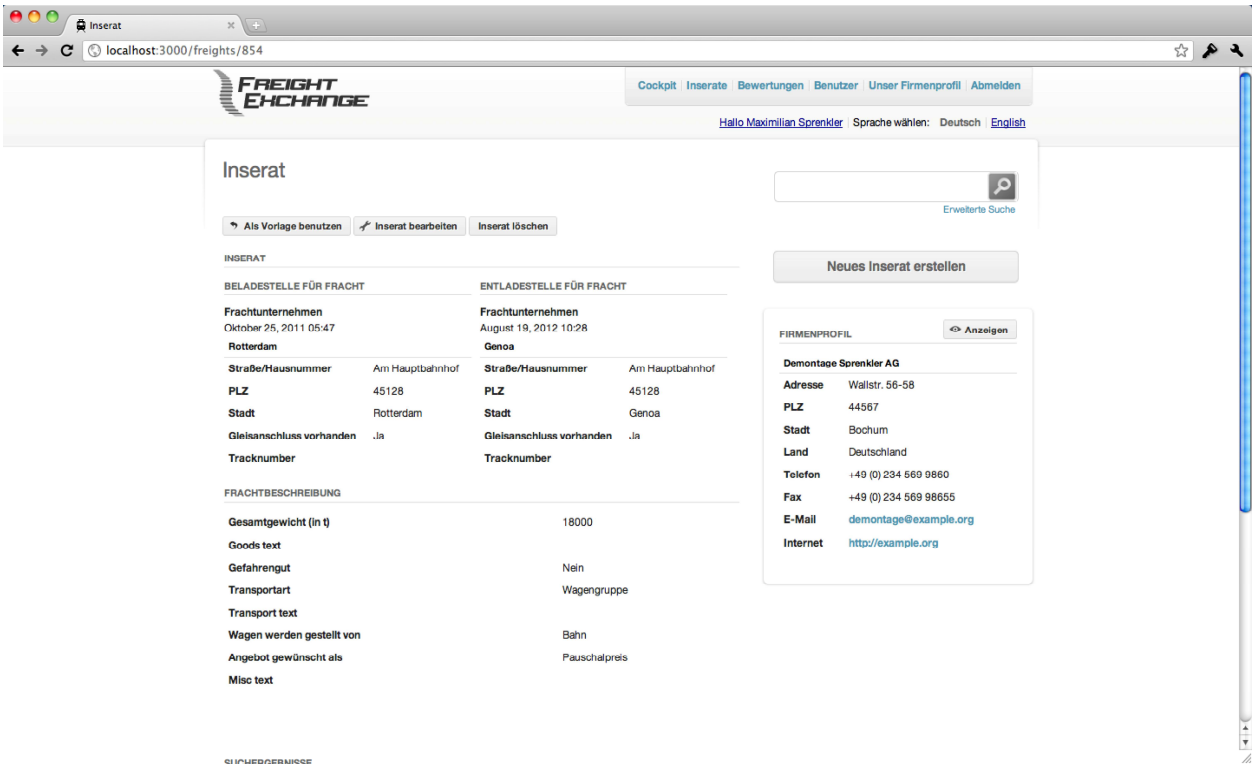

Abbildung 17: Screenshot eines Inserats

Zum Löschen eines Inserats muss der Benutzer den Löschvorgang nochmals bestätigen. Eine Löschung aus dem System ist permanent und irreversibel.

---

1) Vgl. Klippert/Kowalski/Bruns (2010), S. 73.

2) Hierdurch werden die korrespondierenden Eingabefelder mit den Adressdaten des Ortes ausgefüllt.

3) Gemeint sind hier die natürlichsprachlichen lokalisierbaren Attribute von Inseraten.

4) Diese Inserate werden mithilfe der Matching-API bestimmt. Es werden Laderaumresultate für Frachtrauminserate und umgekehrt gelistet.

Bei der Benutzung eines bestehenden Inserats als Vorlage werden alle Attribute des bestehenden Inserats als Initialwerte für die Attribute des neuen Inserats verwendet. Anschließend kann das neue Inserat noch angepasst werden, bevor es tatsächlich in der Datenbank gespeichert wird.

7.3.2 Suche nach Inseraten und potentiellen Geschäftspartnern

Die Suche nach Inseraten dient dem Auffinden potentieller Geschäftspartner im System der Online-Frachtenbörse.

Das Frontend unterstützt dieses Ziel durch folgende Funktionen:

- Benutzer, die selbst Laderaum inserieren, suchen automatisch nach Frachtraum.[1]
- Die Suchergebnisse werden nach optischen und informationstechnischen Aspekten ansprechend präsentiert.[2]
- Potentielle Geschäftspartner werden in den Suchergebnissen hervorgehoben.
- Nicht mehr aktuelle Inserate werden in den Suchergebnissen nicht aufgeführt.

Die primäre Suchfunktion des Softwareprototyps wird als Volltextsuche implementiert (mithilfe eines einfachen Textfelds, welches einen Freitext zur Suche entgegennimmt) anstatt über eine möglichst detailreiche und damit komplexe Eingabemaske.[3] So wird eine Benutzererfahrung erzeugt, die sich an moderne Suchmaschinen anlehnt und bei ausreichend guten Suchalgorithmen trotz des simplen Suchprozesses ebenfalls zufriedenstellende Ergebnisse liefert.

Abbildung 18 zeigt diese Suchmaske und die Darstellung der Suchergebnisse.

1) Analog suchen Benutzer, welche selbst Frachtraum inserieren, automatisch nach Laderaum.

2) Optische und informationstechnische Aspekte meinen hier, dass die für den suchenden Benutzer relevanten Informationen (bspw. Start- und Zielort oder Transportzeitraum) ansprechend aufbereitet werden und einfach zu lesen sowie zu verstehen sind. Eine zu hohe Informationsdichte und Datenkomplexität ist in den Suchergebnissen zu vermeiden.

3) Der Grund hierfür ist, dass komplexe Eingabemasken Benutzer unter Umständen überfordern, vgl. Klippert/Kowalski/Bruns (2010), S. 57 sowie S. 9.

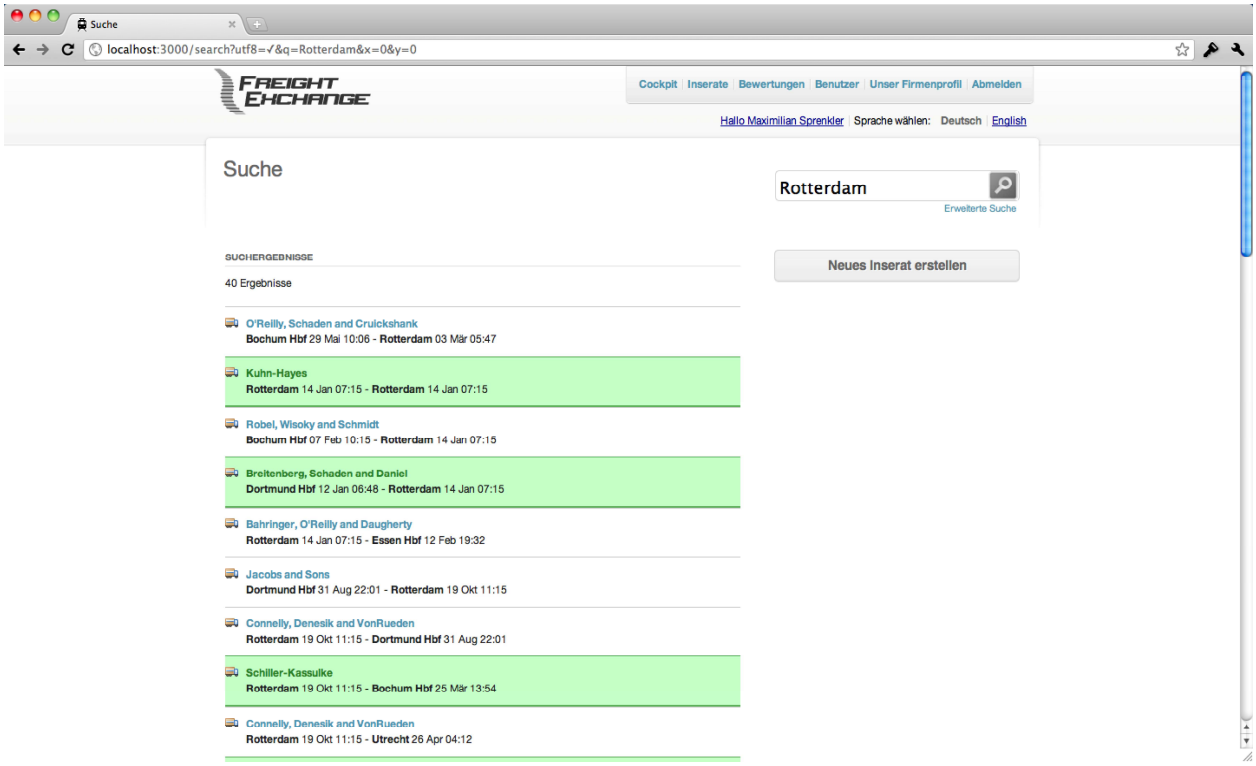

Abbildung 18: Screenshot der Suchergebnisseite

Die Suchergebnisse der Volltextsuche setzen sich aus Inseraten und Unternehmen zusammen, welche den gesuchten Begriffen entsprechen.[1]

Um auch komplexere Suchanfragen stellen zu können, wird ebenfalls eine „erweiterte Suche" implementiert, also eine komplexe Maske, mit deren Hilfe der Benutzer genaue Vorgaben zu einzelnen Datenbankfeldern der gewünschten Suchergebnisse machen kann.

In der Eingabemaske können für numerische Eigenschaften, wie bspw. das Gesamtgewicht, Intervalle angegeben werden, in denen sich die Werte der Suchergebnisse bewegen sollen. Für die natürlichsprachlichen Eigenschaften können Suchbegriffe eingegeben werden, die in den betreffenden Eigenschaften vorkommen müssen.[2] Insbesondere beim Feld „Postleitzahl" ist dies praktisch relevant, da bspw. durch Eingabe von „45" alle Inserate aus dem Raum Essen, Mülheim an der Ruhr, Recklinghausen und Gelsenkirchen gefunden werden.[3] Für alle auswählbaren Eigenschaften, wie bspw. die Angabe, ob es sich bei der Fracht um Gefahrengut handelt, kann der Benutzer eine Vorgabe machen oder das Feld freilassen.[4]

Im Rahmen einer erweiterten Suche müssen alle vom Benutzer angegeben Kriterien erfüllt sein.

Abbildung 19 zeigt die Eingabemaske der erweiterten Suche.

---

1) Die gefundenen Unternehmen werden in der Seitenleiste getrennt von den Inseraten aufgeführt.

2) Dies bedeutet, dass der eingegebene Text nicht exakt mit dem Attributswert übereinstimmen muss. Wird bspw. bei „Bahnhof" der Begriff „Rotterdam" eingegeben, werden alle Inserate gefunden, bei denen dieser Begriff im Bahnhofsnamen vorkommt (also bspw. „Rotterdam Centraal", aber auch „Bahnhof Rotterdam" oder „Rotterdam Station").

3) Der Grund hierfür ist, dass die Postleitzahlen dieser Städte mit "45" beginnen.

4) Wird ein solches Auswahlfeld freigelassen, so ist die Ausprägung des jeweiligen Attributs für die Suchanfrage irrelevant.

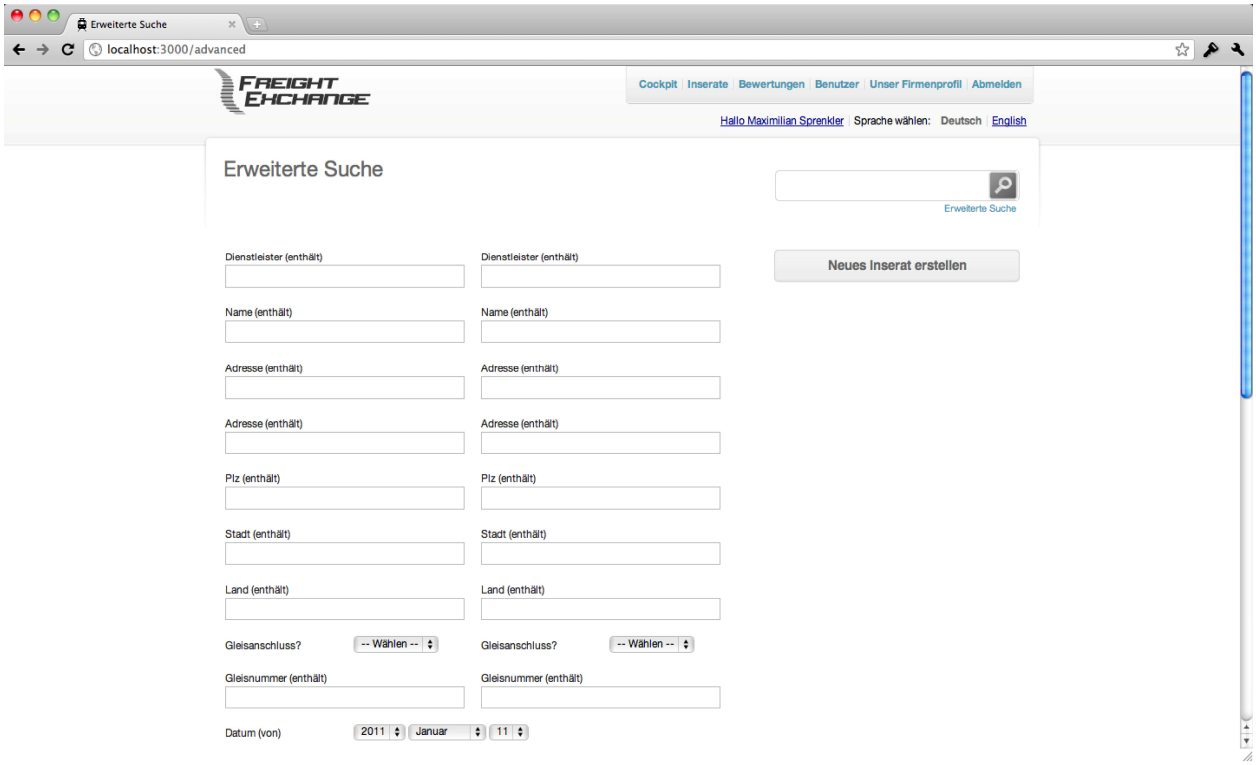

Abbildung 19: Screenshot der Eingabemaske zur erweiterten Suche

7.3.3 Benutzerverwaltung innerhalb eines Unternehmens

Das Frontend ermöglicht das Erstellen, Verwalten und Wiederverwenden von Benutzerdaten.[1]

Eine Übersicht zeigt dem Benutzer alle Benutzer seines Unternehmens und gibt ihm die Möglichkeit, neue Benutzer zu erstellen sowie bestehende Benutzer zu bearbeiten, zu löschen oder als Vorlage für neue Benutzer zu verwenden. Abbildung 20 zeigt diese Übersicht.

1) Vgl. Klippert/Kowalski/Bruns (2010), S. 81 sowie S. 42.

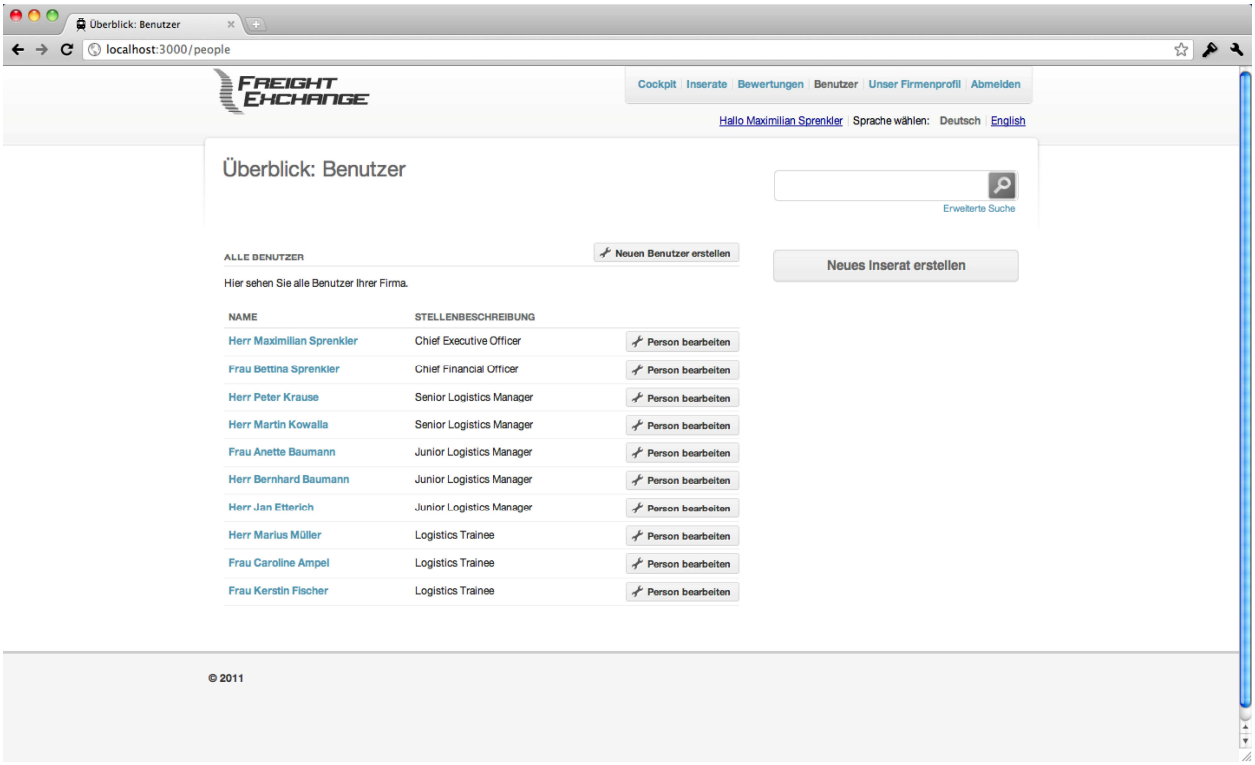

Abbildung 20: Screenshot der Benutzerübersicht eines Unternehmens

Die persönlichen Daten (bspw. Kontaktinformationen) der Benutzer können dabei ebenso bearbeitet werden wie die Stammdaten (bspw. Benutzerrechte).[1] Ein Benutzer wird nicht in der gleichen Eingabemaske bearbeitet, in der er angelegt wurde. So können die Personendaten eines bestehenden Benutzers getrennt von seinen Stammdaten bearbeitet werden, wodurch es möglich wird, dass der Inhaber einer Benutzerrolle zwar zur Änderung der Telefonnummer eines Mitarbeiters, nicht jedoch zur Änderung seines Passworts berechtigt ist.

Im Anschluss an die Erstellung oder Bearbeitung eines Benutzers wird der Benutzer an die Profilseite des soeben erstellten oder bearbeiteten Benutzers weitergeleitet.

Abbildung 21 zeigt die Ansicht eines solchen Benutzerprofils.

1) Vgl. Klippert/Kowalski/Bruns (2010), S. 81.

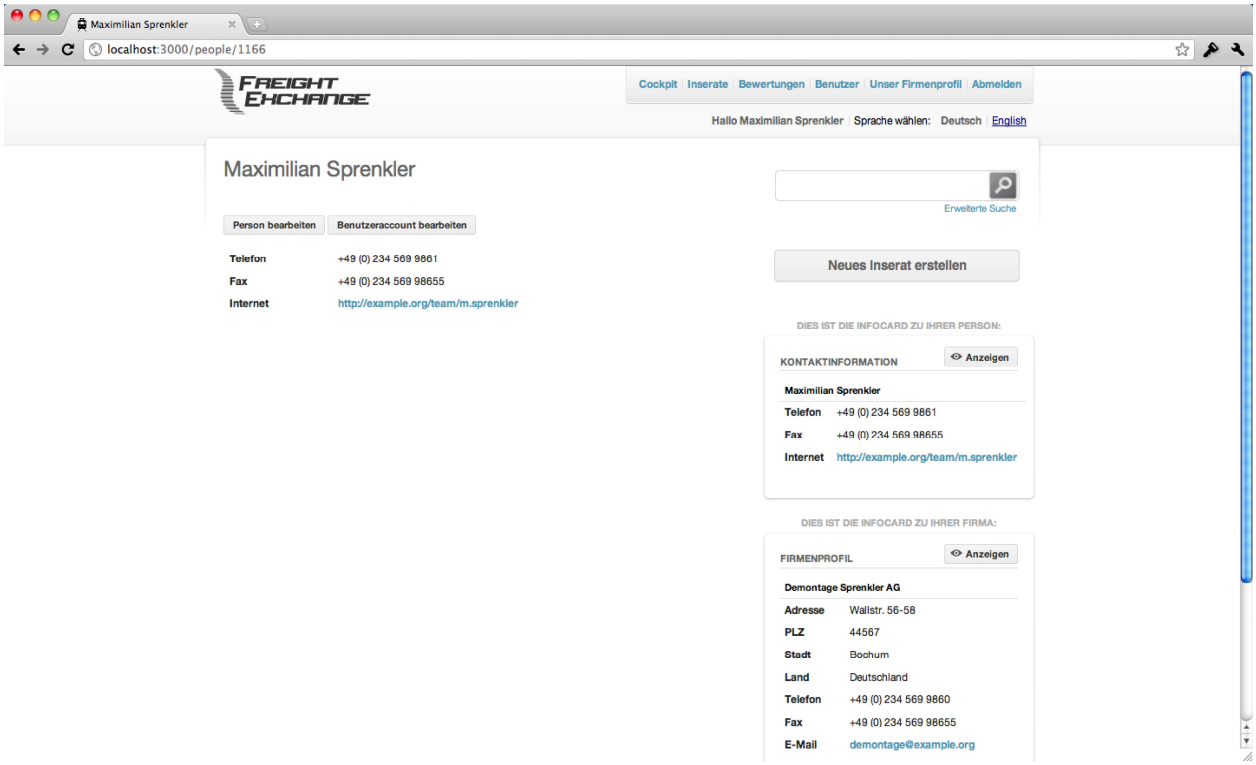

Abbildung 21: Screenshot eines Benutzerprofils

Zum Löschen eines Benutzers muss der Benutzer den Löschvorgang nochmals bestätigen. Eine Löschung aus dem System ist permanent und irreversibel.

Bei der Benutzung eines bestehenden Benutzers als Vorlage werden alle Attribute des bestehenden Benutzers als Initialwerte für die Attribute des neuen Benutzers verwendet. Anschließend kann der neue Benutzer noch angepasst werden, bevor er tatsächlich in die Datenbank gespeichert wird.

Ein zur Nutzung der Benutzerverwaltung berechtigter Benutzer kann wiederum andere berechtigen, Benutzer zu erstellen, zu verwalten und wiederzuverwenden.

Auf der Startseite ihres Unternehmens wird allen Benutzern ein Informationsstrom mit den Aktivitäten aller Benutzer ihres Unternehmens präsentiert.[1]

7.3.4 Hinterlegung von Kontaktinformationen

Benutzer können sowohl zu ihrer Person als auch zu ihrem Unternehmen Kontaktinformationen im System der Online-Frachtenbörse hinterlegen.[2]

Alle Kontaktinformationen können dabei als Freitext ohne Formatierungsvorgaben eingegeben werden. Der Softwareprototyp versucht anschließend diese Informationen zu normalisieren und einheitlich darzustellen.[3] So werden E-Mail- und Internetadressen bei der Anzeige automatisch ver-

1) Dieser "Newsfeed" wird mithilfe der Protokolleinträge zu Aktionen auf Objekten erzeugt. Er entspricht dem Informationsstrom in der Hauptansicht der Statistiken zu Aktionen auf Objekten, vgl. Abbildung 24, S. 5.

2) Gemeint sind hier Adress- und Telefondaten sowie E-Mail- und Internetadressen.

3) Schlägt der Normalisierungsversuch fehl, so wird die betreffende Information wie hinterlegt angezeigt. Da die Informationen nicht normalisiert in die Datenbank gespeichert werden, können die Routinen zur einheitlichen Darstellung jederzeit überarbeitet werden.

linkt und Telefon- sowie Faxnummern in ein einheitliches europäisches Nummernformat gebracht, um eine konsistente Datenpräsentation sicherzustellen.[1]

Sind die durch die Benutzer hinterlegten Informationen unvollständig, erscheint eine Aufforderung zur Vervollständigung der hinterlegten Informationen.[2]

Sowohl Benutzer- als auch Anbieterprofile können zudem lokalisierbare natürlichsprachliche Informationen enthalten, um bspw. eine biografische Beschreibung in mehreren Sprachen zu hinterlegen.[3]

7.3.5 Bewertung von Geschäftspartnern

Das Frontend kann Anbieter anhand der Anzahl ihrer positiven Bewertungen bei den Suchergebnissen optisch hervorheben.[4] So wird dem Benutzer zusätzlich zu den normalen Suchergebnissen eine Vorauswahl an Inseraten potentiell guter Anbieter präsentiert, wodurch das Ziel unterstützt wird, den Benutzer zu führen und möglichst nie zu überfordern.

Es erscheint plausibel, dass durch ausreichend differenzierte Kenntlichmachung der so belegten qualitativen Unterschiede zwischen den Anbietern diese dazu übergehen werden, ihre Geschäftspartner aufzufordern ihnen eine Bewertung in der Online-Frachtenbörse zu schreiben.[5]

In Abbildung 18 wurde die Hervorhebung einzelner Suchergebnisse auf Basis der positiven Bewertungen des jeweiligen Anbieters gezeigt.[6]

7.4 Funktionen des Backends

7.4.1 Generelle Administration der Online-Frachtenbörse

Das Backend ermöglicht die Administration der Online-Frachtenbörse.

Den ersten Berührungspunkt mit dem Backend bildet der Systemassistent, über den die Administratoren die Online-Frachtenbörse konfigurieren und generelle Funktionen des Backends aufrufen können.

So lassen sich über das Backend Einstellungen erstellen, verwalten, anpassen und löschen. Diese Einstellungen bestimmen bspw., ob die Applikation im sogenannten „Demo-Modus"[7] betrieben wird und wie hoch die Schwellenwerte für die Matching-API oder die Hervorhebung von Suchergebnissen sind.

---

1) So ist es unerheblich, ob der Benutzer "0234/56789-100", "+49 234 567 89100" oder "004923456789100" eingibt: Die Telefon- oder Faxnummer wird stets im Format "+49 (0) 234 567 89100" angezeigt.

2) Das Kriterium der Unvollständigkeit lässt sich über einen anpassbaren Schwellenwert konfigurieren. Bspw. kann ein zur Hälfte ausgefülltes Profil bereits als "hinreichend ausgefüllt" deklariert werden.

3) Zu diesen lokalisierbaren natürlichsprachlichen Attributen von Objekten vgl. S. 37 f.

4) Die optische Hervorhebung könnte bspw. mit dem in grün fettgeschriebenen Anbieternamen für 10 positive Bewertungen beginnen und bei der Hervorhebung des gesamten Angebots innerhalb der Suchergebnisse für mehr als 100 positive Bewertungen enden. Diese initialen Schwellenwerte sind über das Backend konfigurierbar und mit der Zeit durch empirisch belegte Werte zu ersetzen.

5) Vgl. Klippert/Kowalski/Bruns (2010), S. 49.

6) Vgl. S. 56.

7) Der "Demo-Modus" erlaubt den Administratoren ein "Demo-Unternehmen" zu erstellen.

Zudem unterstützt das Backend die Lokalisierung der Benutzeroberfläche in weitere Sprachen.[1] So bietet es eine separate Benutzeroberfläche zur Übersetzung aller verwendeten Ausdrücke und Formate. Die so übersetzten Lokalisierungsdaten können anschließend über den Systemassistenten ins System eingepflegt werden.[2]

Des Weiteren lassen sich über das Backend Benutzerrollen erstellen, verwalten, anpassen und löschen. Allerdings können über das Backend nicht die mit einer Benutzerrolle verbundenen Berechtigungen geändert werden.[3]

7.4.2 Suche nach Inhalten

Das Backend bietet die Möglichkeit, Objekte im System der Online-Frachtenbörse zu suchen. Im Gegensatz zur Suche im Frontend findet die Suche des Backends nicht nur Inserate, sondern Objekte jeden Typs.

Dies bedeutet, dass bspw. eine Suche nach dem Schlagwort „Rotterdam" nicht nur Fracht- und Laderauminserate liefert, die das Schlagwort in ihrer Beschreibung des Start- oder Zielortes oder einem sonstigen Freitextfeld beinhalten, sondern auch bspw. Benutzer, Bewertungen, Unternehmen und Benutzerrollen. Zusätzlich wird jedes Suchergebnis nach einem individuellen Schema dargestellt. So wird bei gefundenen Benutzern neben anderen Informationen automatisch die E-Mail-Adresse angezeigt und verlinkt, um eine Kontaktaufnahme mit dem Benutzer durch die Administratoren zu erleichtern.

Abbildung 22 zeigt diese Ansicht.

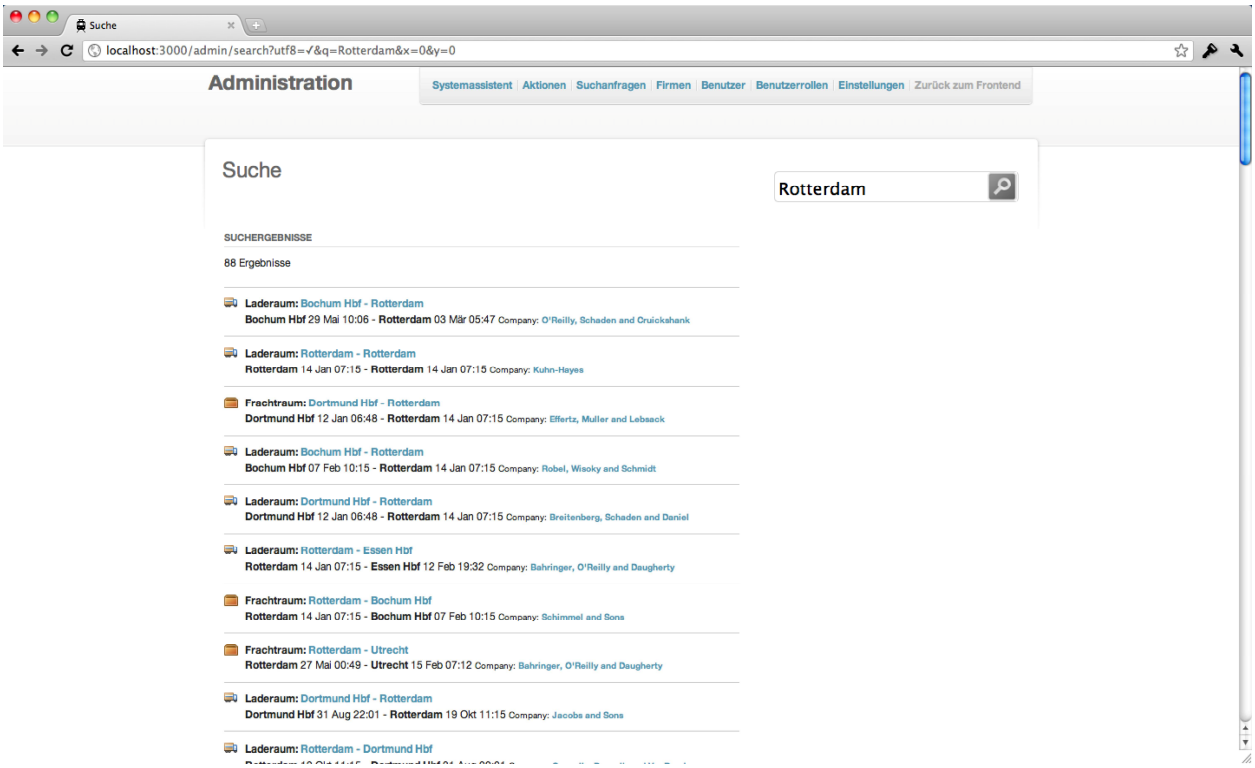

Abbildung 22: Screenshot der Suchergebnisseite des Backends

1) Zur Lokalisierung des Softwareprototyps vgl. S. 72.

2) Wird der Softwareprototyp als Produktionssystem betrieben, muss der Server neu gestartet werden, um die Lokalisierungsdaten neu zu laden.

3) Dies geschieht auf Quelltextebene innerhalb der Applikation.

Die Suchergebnisse können nach Objektart gefiltert werden, um so das eigentlich Gesuchte weiter einzugrenzen. So kann ein Administrator bspw. nach einem Teil einer E-Mail-Adresse eines Benutzers suchen und die Suche anschließend auf die Objektart „Benutzer" einschränken.

Zudem bieten einige Unterseiten, wie bspw. die Detailansicht einer Benutzerrolle, die Möglichkeit, eine Übersicht aller verwandten Objekte zu generieren, wie bspw. alle Benutzer, die die angezeigte Benutzerrolle innehaben, oder alle Inserate, Benutzer oder Bewertungen des angezeigten Unternehmens.

7.4.3 Benutzerverwaltung innerhalb des Backends

Im Rahmen der erweiterten Benutzerverwaltung des Backends haben die Administratoren der Online-Frachtenbörse die Möglichkeit, Benutzer und Unternehmen zu verwalten, ihre Teilnehmerdaten inklusive der zugewiesenen Benutzerrollen zu ändern sowie Benutzer und Unternehmen zu löschen.[1]

Abbildung 23 zeigt diese Ansicht.

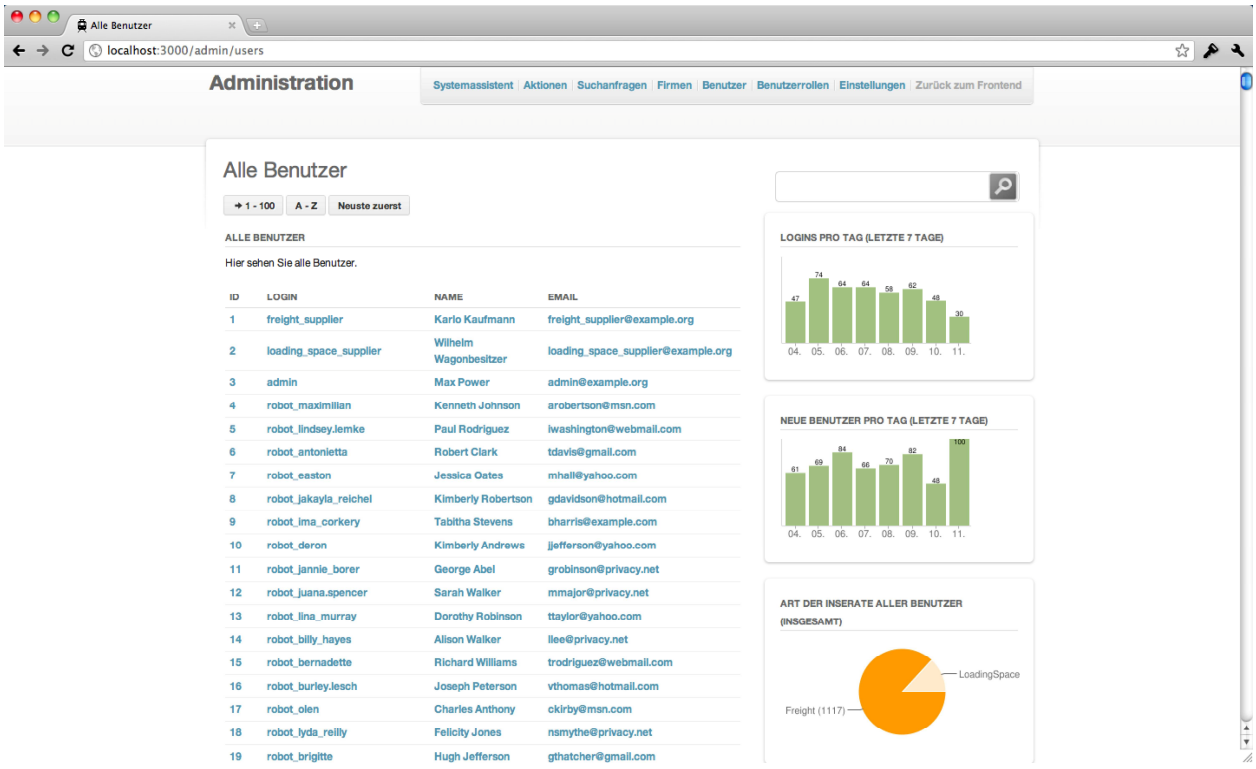

Abbildung 23: Screenshot der Benutzerübersicht des Backends

Die Möglichkeit einer zeitweisen Sperrung von Benutzern oder Unternehmen existiert nicht.[2] Zum Löschen eines Benutzers oder eines Unternehmens muss der Administrator den Löschvorgang nochmals bestätigen. Eine Löschung aus dem System ist permanent und irreversibel.

Über die Benutzerverwaltung des Backends können Administratoren andere Benutzer dazu berechtigen, das Backend zu nutzen.

Über das Backend können Benutzer zudem nach verschiedenen Kriterien, wie bspw. Benutzerrollen oder Unternehmenszugehörigkeit, gefiltert werden.

---

1) Vgl. Klippert/Kowalski/Bruns (2010), S. 81 ff.

2) Diese Funktionalität bildet jedoch ein identifizierbares Weiterentwicklungspotential des Softwareprototyps.

7.4.4 Angepasste Benutzeroberflächen des Backends

Mithilfe des Rechtesystems können im Backend für Mitarbeiter verschiedener Bereiche, bspw. Marketing oder Management, verschiedene Benutzeroberflächen zur Verfügung gestellt werden, die den Ansprüchen der jeweiligen Benutzergruppe Rechnung tragen.

Dies geschieht, indem die Applikation für jede Benutzerrolle des Administrators überprüft, ob für diese Benutzerrolle eine angepasste Benutzeroberfläche im System hinterlegt ist. So hat die Benutzerrolle „marketing" bspw. keinerlei zugriffsrestriktive Bedeutung, sorgt jedoch dafür, dass einem Inhaber dieser Benutzerrolle auf der Startseite des Backends nicht eine Übersicht der Benutzeraktionen, sondern die Entwicklung von Kennzahlen präsentiert wird.

Auf diese Weise können verschiedene Werkzeuge in die Benutzeroberfläche des Backends eingearbeitet werden, welche den Betreibern der Online-Frachtenbörse die Administration und Wartung erleichtern werden. So ist bspw. denkbar, Supportmitarbeitern die Kontaktdaten von Benutzern auch in Auflistungen und Übersichten direkt zur Verfügung zu stellen, so dass sie sich nicht auf das jeweilige Profil durchklicken müssen, um einen Benutzer zu kontaktieren.

Sollte ein Administrator mehrere Benutzerrollen besitzen, für die jeweils verschiedene, angepasste Benutzeroberflächen hinterlegt sind, so präsentiert die Applikation stets die zur ersten dieser Benutzerrollen gehörende Benutzeroberfläche.

7.4.5 Nutzungsstatistiken

Durch die Protokollierung des Benutzerverhaltens[1] ist das Backend in der Lage, detaillierte Statistiken über das Nutzungsverhalten der Teilnehmer der Online-Frachtenbörse zu erstellen.

Allen statistischen Ansichten des Backends liegt stets ein fester Betrachtungszeitraum zugrunde.[2] Sie werden durch grafische Abbildungen in Form von Diagrammen unterstützt.[3]

Die Statistiken dienen zum einen der Beantwortung der in Kapitel 7.2.5 aufgeworfenen Fragen über das Benutzerverhalten im Rahmen der Anforderungsanalyse und zum anderen der Visualisierung von Kennzahlen, wie bspw. der Kundenentwicklung,[4] sowie der Überwachung von Mitarbeitern der Online-Frachtenbörse.[5]

Abbildung 24 zeigt die Hauptansicht der Statistiken zu Aktionen auf Objekten.

---

1) Vgl. S. 43.

2) In der vorliegenden Implementierung ist dieser Zeitraum auf die vergangenen sieben Tage festgelegt. Die den Statistiken zugrundeliegenden Methoden wurden allerdings so implementiert, dass der Zeitraum für weitere Ansichten beliebig neu gewählt werden kann, um so bspw. Monats- oder Jahresübersichten zu generieren.

3) Vgl. Klippert/Kowalski/Bruns (2010), S. 83.

4) Vgl. Klippert/Kowalski/Bruns (2010), S. 83.

5) Vgl. Klippert/Kowalski/Bruns (2010), S. 84.

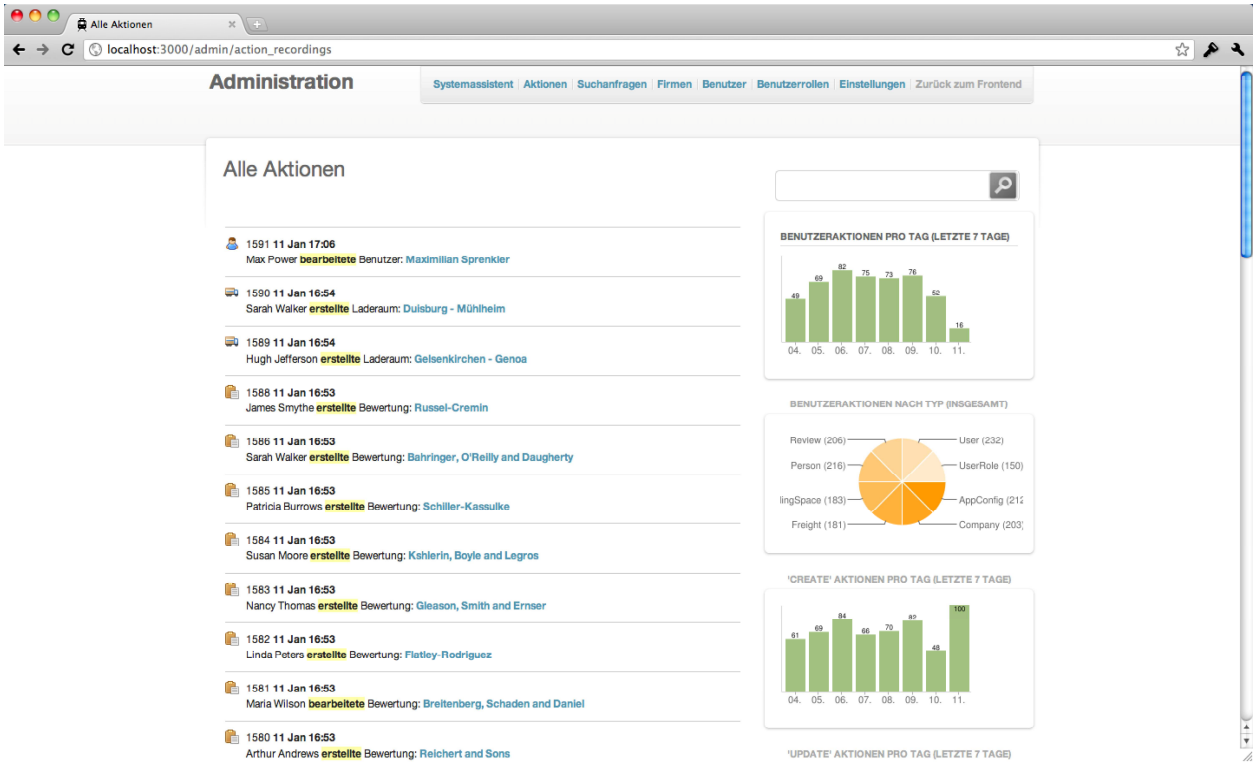

Abbildung 24: Screenshot der Statistik zu Aktionen auf Objekten

Die Statistiken zu Aktionen auf Objekten stellen dar, welche Benutzer im Betrachtungszeitraum wann aktiv sind und welche Aktionen sie dabei auf welchen Objekten ausführen. So kann bspw. ermittelt werden, wie oft Inserate nachträglich editiert werden oder wie schnell eine Bewertung im Durchschnitt von dem bewerteten Unternehmen freigeschaltet wird.

Abbildung 25 zeigt die Hauptansicht der Statistiken zu Suchanfragen.

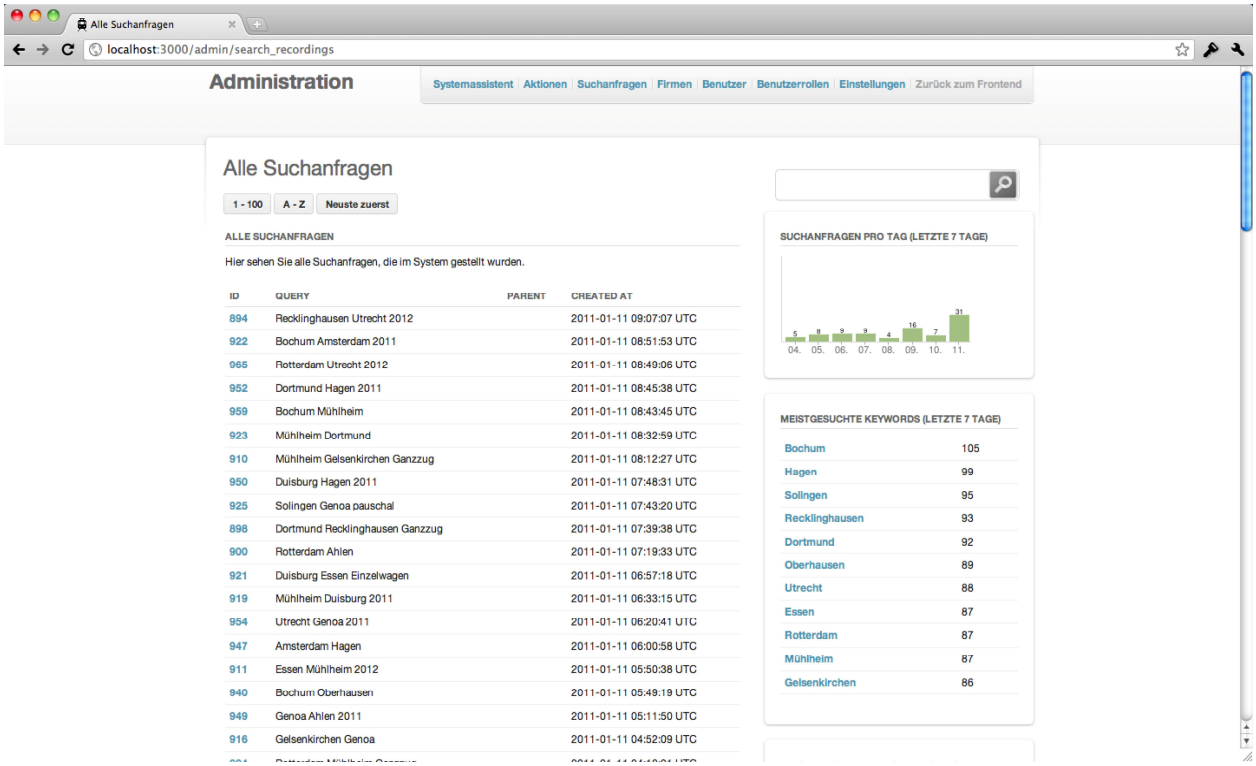

Abbildung 25: Screenshot der Statistik zu Suchanfragen

Die Statistiken zu Suchanfragen stellen dar, welche Benutzer im Betrachtungszeitraum wann nach welchen Schlüsselwörtern gesucht haben, welches die meistgesuchten Schlüsselwörter waren, welche Schlüsselwörter die meisten Resultate lieferten und bei welchen Schlüsselwörtern die meisten Suchergebnisse angeklickt wurden. So kann bspw. ermittelt werden, in welchen Kombinationen nach den meistgesuchten Schlüsselwörtern gesucht wurde, also bspw., ob „Rotterdam" häufig in Verbindung mit „Gefahrengut" und „Einzelwagen" gesucht wird.

Neben diesen dedizierten statistischen Ansichten werden Statistiken im Backend auch in Standardübersichten verwendet, um bspw. das Wachstum der betreffenden Datenbanktabelle zu visualisieren (bspw. neue Benutzer) oder die Verteilung bestimmter Attributsausprägungen innerhalb einer Tabelle (bspw. „Einzelwagen" als Wagenart bei Inseraten).

8 Installation des Softwareprototyps

8.1 Voraussetzungen

Grundlegende Voraussetzungen zur Installation des Softwareprototyps sind eine funktionsfähige Installation des Ruby-Kernpakets sowie der Paketmanagement-Software RubyGems.[1] Durch die Plattformunabhängigkeit aller verwendeten Technologien ist das eingesetzte Betriebssystem frei wählbar. Über RubyGems muss zuerst das Gem „bundler" installiert werden. Bundler installiert alle benötigten Komponenten automatisch.[2]

8.2 Konfiguration

Ist der Softwareprototyp wie beschrieben installiert worden, muss die Datenbank mit bestimmten Startwerten initialisiert werden.[3] Dies geschieht über die Rake-Tasks „db:migrate" sowie „db:seed".[4]

Je nachdem, ob die aktuelle Installation des Softwareprototyps als Entwicklungs- oder Produktionssystem betrieben wird, werden durch den letzten Rake-Task weitere Startwerte hinzugefügt.[5]

Nach dem Initialisieren der Datenbank ist der Softwareprototyp bereit für den ersten Start. Die weitere Konfiguration der Online-Frachtenbörse geschieht nun über das Backend.

---

1) RubyGems ist im Internet unter der URL http://rubygems.org/ kostenlos verfügbar. Bei der Implementierung des Softwareprototyps wurden die Versionen Ruby 1.8.7 und RubyGems 1.3.7 verwendet. Prinzipiell sollte der Softwareprototyp auch mit jüngeren Versionen betrieben werden können, getestet wurde dies, bspw. mit Ruby 1.9, jedoch nicht.

2) Während auf vielen unixoiden Betriebssystemen das Datenbanksystem SQLite bereits vorinstalliert ist (bspw. Mac OS X), muss es bspw. unter Windows XP nachträglich manuell heruntergeladen und installiert werden, vgl. Kreibich (2010), S. 18 sowie S. 20. Die Komponente "nokogiri" muss ebenfalls manuell installiert werden. Der Grund hierfür ist, dass nokogiri auf die Komponenten "libxslt" sowie "libxml2" angewiesen ist, vgl. Anhang, S. 91.

3) Startwerte sind bspw. vordefinierte Benutzerrollen.

4) Rake-Tasks werden über die Kommandozeile aufgerufen, vgl. Anhang, S. 91.

5) So werden im Falle eines Entwicklungssystems zusätzlich einige Anbieter, Benutzer und Inserate angelegt.

9 Betrieb des Softwareprototyps

Nach der Installation des Softwareprototyps wird dieser erstmalig gestartet. Dabei erhält der Benutzer die Möglichkeit, die Online-Frachtenbörse über den Systemassistenten zu konfigurieren.[1] Unabhängig davon, ob ein Entwicklungs- oder Produktionssystem betrieben wird, kann die weitere Konfiguration nun über die Benutzeroberfläche erfolgen.

Abbildung 26 zeigt die Startseite dieses Konfigurationsprozesses.

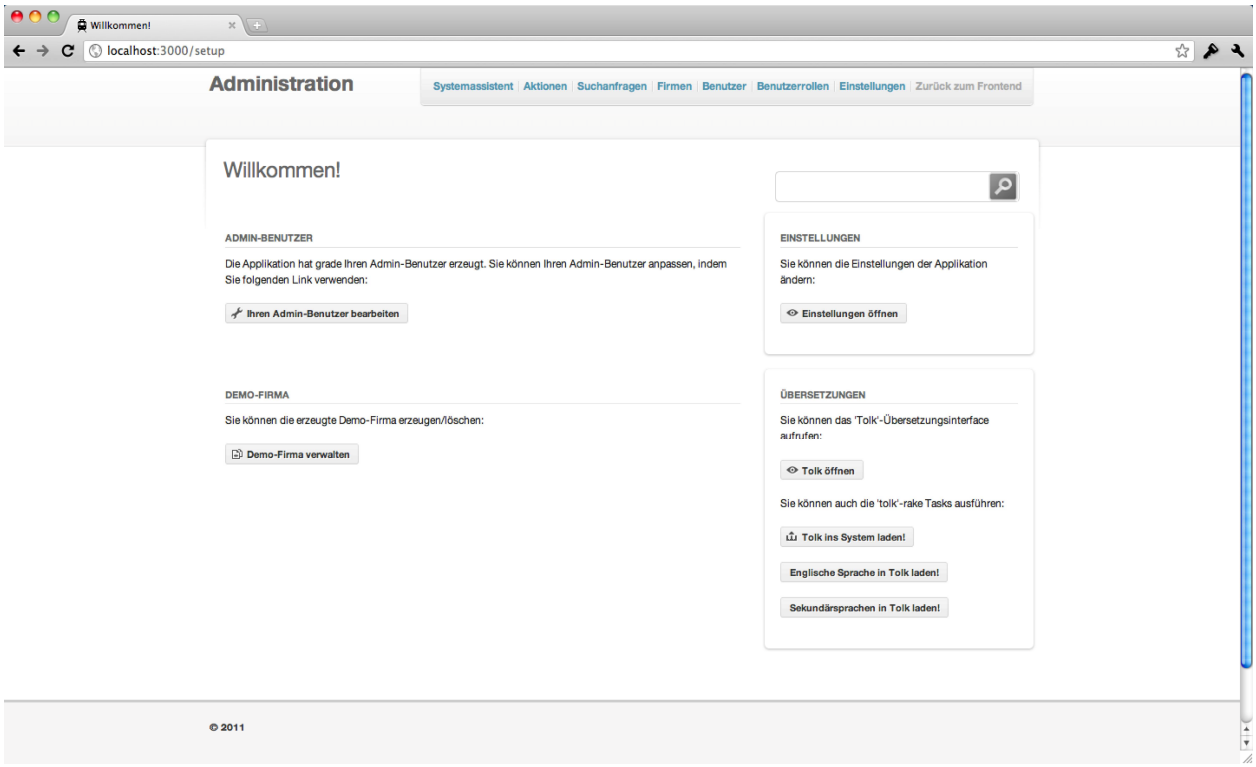

Abbildung 26: Screenshot der Startseite des Systemassistenten

Zunächst wird der erste Administrator angelegt.

Nachdem die Konfiguration abgeschlossen ist, kehrt der erstmalige Benutzer zu der normalen Benutzeroberfläche zurück und ist als der eben angelegte Administrator im System der Online-Frachtenbörse angemeldet.

Der Betrieb des Softwareprototyps und sein Einsatz im Rahmen der weiteren Anforderungsanalyse erfordern eine nach anderen Kriterien gewählte Umgebung, als die Implementierung des Softwareprototyps erforderte.

So sollte der Softwareprototyp als Produktionssystem auf einem performanten, modernen Server betrieben werden. Obwohl die Ausführungsgeschwindigkeit im Rahmen der Implementierung des Softwareprototyps als unwichtig eingestuft wurde, so ist die Ausführungsgeschwindigkeit des fertigen Softwareprototyps im Betrieb für den Benutzer von subjektiv hoher Bedeutung, da sie sein Benutzungserlebnis mit der Software schnell negativ prägen kann, sollte sie hinter seinen Erwartungen zurückbleiben.

---

1) Der Systemassistent unterstützt die Administratoren bei der Administration der Online-Frachtenbörse.

Daraus folgt, dass die Ausführungsgeschwindigkeit des Softwareprototyps der einer realen Online-Frachtenbörse entsprechen sollte, um im Rahmen der Anforderungsanalyse verlässliche Ergebnisse zu liefern.[1] Kann die Online-Frachtenbörse durch technische Einschränkungen nicht wie von den Betreibern vorgesehen genutzt werden, leidet hierunter auch die Verifizierbarkeit, Reproduzierbarkeit und Objektivität der mithilfe des Softwareprototyps gewonnenen Nutzungsstatistiken.

Zur Steigerung der Authentizität des Benutzungserlebnisses können die in Kapitel 0 beschriebenen Roboter eingesetzt werden, um die Online-Frachtenbörse belebter wirken zu lassen.

Der Softwareprototyp kann optional in einem „Demo-Modus" betrieben werden, welcher es Benutzern ermöglicht, ein „Demo-Unternehmen"[2] anzulegen und sich ohne Eingabe eines Benutzernamens oder Passworts als ein Benutzer dieses Unternehmens anzumelden.[3]

1) Anforderungen an die Performanz werden im Rahmen der Anforderungsanalyse oft vernachlässigt, vgl. Bray (2002), S. 16 f.

2 Dieses "Demo-Unternehmen" ist ein durch das Backend dynamisch erstellbares Unternehmen, welche mit mehreren Mitarbeitern und Inseraten initialisiert ist.

3 Diese Funktionalität ist aus Sicherheitsgründen nicht grundsätzlich verfügbar, sondern muss über den Systemassistenten von einem Administrator freigeschaltet werden.

10 Weiterentwicklung des Softwareprototyps

10.1 Konzeption der Weiterentwicklung

Es konnten im Rahmen dieses Projektberichts aus finanziellen, zeitlichen und personellen Gründen nicht alle von der Grundlagenforschung bisher identifizierten und für das Primärziel dieses Projektberichts als relevant erachteten Funktionalitäten implementiert werden, wodurch die vorliegende Implementierung als unvollständig anzusehen ist.

Im Folgenden werden die vorhandenen Defizite des Softwareprototyps identifiziert und es wird jeweils dargestellt, wie diese als Weiterentwicklungspotentiale begriffen und genutzt werden können.

Die Weiterentwicklungspotentiale des Softwareprototyps werden in drei Bereichen dargestellt:

- systemisches Weiterentwicklungspotential,
- Weiterentwicklungspotential des Frontends sowie
- Weiterentwicklungspotential des Backends

Mit „systemischem Weiterentwicklungspotential" sind mögliche Verbesserungen gemeint, die nicht direkt Teil der Benutzeroberfläche sind, aber dennoch durch das System der Online-Frachtenbörse bereitgestellt werden könnten.

Mit „Weiterentwicklungspotential des Frontends" sind mögliche Verbesserungen der Benutzeroberfläche und -führung gemeint sowie die allgemeine Datenpräsentation der Online-Frachtenbörse.

Mit „Weiterentwicklungspotential des Backends" sind analog mögliche Verbesserungen gemeint, die nur im den Administratoren zugänglichen Teil der Systemkomponenten möglich sind.

Teilweise entstehen durch diese Aufteilung Probleme bei der Zuordnung einzelner Defizite und daraus resultierende Weiterentwicklungspotentiale des Softwareprototyps. So ist die Implementierung effektiverer Suchalgorithmen ein systemisches Weiterentwicklungspotential, die bessere Präsentation der besseren Ergebnisse ein Weiterentwicklungspotential des Frontends und die detailliertere grafische Analyse der Suchanfragen ein Weiterentwicklungspotential des Backends. In derartigen Fällen werden die Teilaspekte der jeweiligen Verbesserungen möglichst trennscharf voneinander in den korrespondierenden Kapiteln erläutert.

Im Folgenden wird dargestellt, wie die Weiterentwicklungspotentiale des Systems, Frontends und Backends genutzt werden können, um die Ziele des Softwareprototyps zu erfüllen. Nicht betrachtet werden Weiterentwicklungspotentiale, die nicht dem Primärziel dieses Projektberichts (der Unterstützung der Anforderungsanalyse) dienen.[1] Auch werden Potentiale nur aufgrund ihrer technischen Machbarkeit und plausiblen Nützlichkeit identifiziert, jedoch nicht explizit empfohlen. Die weitere Anforderungsanalyse muss zeigen, inwiefern sich Weiterentwicklungen in den identifizierten Bereichen lohnen.

1) Gemeint sind hier bspw. die Implementierung eines hohen technischen Sicherheitsstandards (vgl. Klippert/ Kowalski/Bruns (2010), S. 40 sowie S. 60), Prognosen zu Umsatz- und Kundenentwicklung über das Backend (vgl. Klippert/Kowalski/Bruns (2010), S. 83) und die Implementierung von Instrumenten zur Neukundenakquise (vgl. Klippert/Kowalski/Bruns (2010), S. 82 f.), was dazu führt, dass das Weiterentwicklungspotential des Backends im Folgenden knapper behandelt wird als bspw. das Weiterentwicklungspotential des Frontends.

10.2 Systemisches Weiterentwicklungspotential

10.2.1 Fracht- und Laderauminserate

Im Bereich der Fracht- und Laderauminserate ergeben sich Weiterentwicklungspotentiale, da diese Inserate detaillierter beschrieben werden können. Dies bedeutet, dass die Datenstrukturen und Eingabemasken zu Fracht- und Laderauminseraten in der vorliegenden Implementierung als strukturelle Konzepte anzusehen sind, welche auf Basis der zur Verfügung stehenden Literatur modelliert wurden.[1]

Auch fehlt in der vorliegenden Implementierung die informationstechnische Abbildung realer Begebenheiten auf Strecken und in Bahnhöfen sowie anderer ortsspezifischer Besonderheiten. Eine quantitative Erfassung derartiger kriterieller Besonderheiten einer nachgefragten Dienstleistung könnte dazu beitragen, jene Merkmale, die momentan rein natürlichsprachlich beschrieben werden, ebenfalls strukturiert über die Matching-API vergleichen zu können und so zu qualitativ besseren Vergleichen zwischen Angebot und Nachfrage zu gelangen.

Somit kann die Verbesserung der Beschreibbarkeit von Inseraten als wichtiges Weiterentwicklungspotential identifiziert werden.

10.2.2 Bewertung von Geschäftspartnern

Das implementierte Bewertungssystem könnte weiterentwickelt werden, indem zusätzlich zu den Geschäftspartnern, die sich untereinander bewerten, auch andere Benutzer zu einer veröffentlichten Bewertung angeben können, ob sie diese bei der Entscheidung für oder gegen den Anbieter hilfreich fanden oder nicht.

Des Weiteren wäre zu eruieren, ob man zu den rein textlichen Bewertungen zusätzlich quantitative Bewertungsmerkmale implementiert[2] und evtl. Kommentare anderer Nutzer zu veröffentlichten Bewertungen zulässt.[3]

Die Implementierung eines weiterführenden, sozial vernetzteren Bewertungssystems kann somit als Weiterentwicklungspotential identifiziert werden.

10.2.3 Schnittstellen

10.2.3.1 Matching-API

Die implementierten Algorithmen der Matching-API demonstrieren deren Funktionsweise und bilden ein Grundgerüst zur Durchführung eines strukturierten Vergleichs zweier Objekte. Allerdings sind sie nicht dazu geeignet, ernstzunehmende Ergebnisse zur Kompatibilität von Fracht- und Laderauminseraten zu liefern.

Die Implementierung effektiver, empirisch verifizierter Vergleichsalgorithmen in die Matching-API stellt somit ein wichtiges Weiterentwicklungspotential dar.

---

1) Vgl. Klippert/Kowalski/Bruns (2010), S. 78.

2) Vgl. Klippert/Kowalski/Bruns (2010), S. 49.

3) Das Zulassen von Kommentaren könnte die soziale Interaktion im System der Online-Frachtenbörse zusätzlich fördern. Die Literatur sieht in der Schaffung derartiger "Community-Dienste" einen möglichen Mehrwert einer Online-Frachtenbörse, vgl. Klippert/Kowalski/Bruns (2010), S. 54.

10.2.3.2 Such-API

Die implementierte Suchtechnologie basiert auf dem statischen Vergleich der Suchanfrage mit allen relevanten Einträgen im Suchindex. Dieses Verfahren ist einfach und performant, allerdings sind effektivere Suchalgorithmen denkbar, die bspw. Tippfehler ignorieren oder semantische Zusammenhänge in der Suchanfrage erkennen. Bessere Suchergebnisse bilden die Grundlage für ein besseres Zusammenbringen von Angebot und Nachfrage.

Die Implementierung einer effektiveren Suchtechnologie stellt somit ein wichtiges Weiterentwicklungspotential dar.

10.2.3.3 XML/JSON-API

Die implementierte XML/JSON-API ermöglicht die Ansteuerung des Softwareprototyps durch Drittanbietersoftware. Allerdings existiert noch keine Anbindungsmöglichkeit an bestehende Systeme, um bspw. Daten aus bestehenden ERP- und WW-Systemen auslesen zu können.[1]

Die Implementierung einer Anbindungsmöglichkeit externer Systeme stellt somit ein weiteres Weiterentwicklungspotential dar.

10.2.4 Rechtesystem

Das implementierte Rechtesystem ermöglicht die Zuweisung von Benutzerrollen an Benutzer. Diese Zuweisung könnte ebenfalls benutzt werden, um Benutzern nicht nur Rechte, sondern auch Typen im Sinne einer Typisierung zuzuweisen. Hierdurch wäre eine Benutzer- und damit letztlich Kundensegmentierung, bspw. zur Bildung von A-, B- und C-Gruppen, möglich.[2] Das implementierte Rechtesystem müsste hierzu nicht erweitert werden. Es müsste lediglich eine sinnvolle Benutzertypisierung erarbeitet und eine Auswertungsmöglichkeit implementiert werden.

Auf die gleiche Weise könnten Benutzern Bearbeitungszustände zugewiesen werden, um diesen einen schrittweisen Zugang zum System der Online-Frachtenbörse zu gewähren.[3] Außerdem ließen sich so auch geschlossene Gruppen im System der Online-Frachtenbörse realisieren.[4]

Die effizientere Nutzung des bestehenden Rechtesystems kann somit als Weiterentwicklungspotential identifiziert werden.

10.2.5 Roboter

Die implementierten Roboter sind sehr schlicht konzipiert und verhalten sich zufalls- und heuristikorientiert. Sie genügen damit den Anforderungen im Rahmen dieses Projektberichts.

Während die implementierten Roboter als ausreichend erachtet werden, um dynamische Inhalte im System der Online-Frachtenbörse zu generieren, könnten intelligentere Roboter im Rahmen der Anforderungsanalyse natürliches, menschliches Verhalten simulieren und eine authentischere, hochfrequentierte Testumgebung schaffen.

---

1) Vgl. Klippert/Kowalski/Bruns (2010), S. 74.

2) Vgl. Klippert/Kowalski/Bruns (2010), S. 83.

3) Vgl. Klippert/Kowalski/Bruns (2010), S. 48 f.

4) Vgl. Klippert/Kowalski/Bruns (2010), S. 49.

Die Implementierung intelligenterer Roboter kann somit als Weiterentwicklungspotential identifiziert werden.

10.2.6 Protokollierung des Benutzerverhaltens

Die implementierten Protokollierungsmaßnahmen erfassen alle durch einen im System der Online-Frachtenbörse angemeldeten Benutzer getätigten Veränderungen an Objekten und seine Suchanfragen sowie die im Rahmen der jeweiligen Suchanfragen betrachteten Suchergebnisse. Nicht erfasst werden dagegen Lesevorgänge und die Nutzung von Funktionalitäten, die keine Datenbankoperation oder Suchanfrage zur Folge haben.[1] Durch eine umfangreichere Überwachung des Benutzerverhaltens könnten auch diese Vorgänge erfasst und in Protokolleinträgen gespeichert werden.

Hierzu müssten keine neuen Datenbankstrukturen bereitgestellt werden, da die beschriebenen Vorgänge in den existierenden Objekten der Protokolleinträge abgebildet werden können.[2] Dies würde zu vollständigeren Nutzungsstatistiken des Softwareprototyps beitragen und damit die Anforderungsanalyse unterstützen.

Somit kann die vollständigere Protokollierung der Aktionen der Benutzer innerhalb des Systems der Online-Frachtenbörse als Weiterentwicklungspotential identifiziert werden.

10.2.7 Lokalisierung

Um potentiellen Benutzern des Softwareprototyps den Zugang zur Benutzeroberfläche zu vereinfachen, ist die gesamte Benutzeroberfläche lokalisierbar.[3]

Während der implementierte Softwareprototyp die Möglichkeit bietet, lokalisierte Texte in bspw. Inserate einzubinden, ist die Benutzeroberfläche nur in den Sprachen Englisch und Deutsch abrufbar.

Es ist jedoch mit den vorhandenen Strukturen möglich, die Benutzeroberfläche mit geringem Aufwand in weitere Sprachen zu lokalisieren.

Da die Akzeptanz einer real existierenden Online-Frachtenbörse zum Teil von der Muttersprachlichkeit der Benutzeroberfläche abhängt, lässt sich in diesem Bereich ein wichtiges Weiterentwicklungspotential identifizieren.

10.2.8 Automatisierte Tests

Automatisierte Tests sollen sicherstellen, dass alle Module, Klassen und Funktionen einer Applikation wie vorgesehen funktionieren.[4]

---

1) So wird bspw. die Nutzung des Feedbackformulars oder der erfolglose Versuch ein Objekt zu erstellen nicht protokolliert.

2) So könnte bspw. für einen Lesevorgang "read" als Aktion gespeichert werden, für einen gescheiterten Erstell- oder Löschvorgang "attempt-create" bzw. "attempt-destroy".

3) Lokalisierung bedeutet hier, dass sowohl alle Texte als auch alle Formate, wie bspw. Zahlenformate, Datumsangaben und Preise, für Benutzer in verschiedenen Regionen adäquat übersetzt und gegebenenfalls formatiert werden. Teilweise wird der Begriff "Internationalization" für die Übersetzung aller Texte und der Begriff "Localization" für das Anpassen aller Formate verwendet, wobei beide Begriffe unscharf voneinander abgegrenzt und nicht immer disjunkt sind, vgl. Ediger (2008), S. 236.

4) Vgl. Carneiro/Barazi (2010), S. 233.

Das Webframework Ruby on Rails unterscheidet hierzu drei verschiedene Arten automatisierter Tests:[1]

- Unit Tests,
- Functional Tests sowie
- Integration Tests.

Der implementierte Softwareprototyp enthält keine dieser Tests, da aus zeitlichen Gründen auf ihre Implementierung verzichtet wurde. Generell sollte jede moderne Software automatisch getestet werden, unabhängig davon, ob sie zur Anforderungsanalyse oder als reales Produkt entwickelt wird.

Hierdurch kann in diesem Bereich ein wichtiges Weiterentwicklungspotential identifiziert werden.

10.3 Weiterentwicklungspotential des Frontends

10.3.1 Grafische Präsentation

Es wurde im Rahmen der Implementierung der Benutzeroberfläche des Softwareprototyps primär Wert auf ein funktionales und ergonomisches Design gelegt, das betriebswirtschaftliche Abläufe und Prozesse unterstützt sowie Informationen klar und strukturiert präsentiert. Der öffentlich (ohne vorherige Registrierung) zugängliche Teil der Applikation wurde auf das Registrierungsformular beschränkt, wodurch keine marktplatztypische „Homepage" zum Ziel der Neukundenakquise existiert.

Im Rahmen der Anforderungsanalyse wird sich herausstellen, inwiefern das Erscheinungsbild der Benutzeroberfläche des Softwareprototyps von den getesteten Zielgruppen als angenehm empfunden und angenommen wird.[2] Eine Anpassung der grafischen Präsentation von Daten, Formularen und Suchergebnissen im Rahmen der Weiterentwicklung des Softwareprototyps an die ermittelten Anforderungen erscheint sinnvoll, damit diese die marktspezifischen Anforderungen nicht missachtet.[3]

Des Weiteren muss die Benutzeroberfläche noch in allen relevanten Internetbrowsern getestet und gegebenenfalls angepasst werden.[4] Hierzu müsste zunächst ermittelt werden, welche Computersysteme (und damit Internetbrowser) in der Transportbranche, insbesondere im Schienengüterverkehr, als relevant anzusehen sind.

Somit kann die Überarbeitung der grafischen Präsentation des Softwareprototyps als Weiterentwicklungspotential identifiziert werden.

10.3.2 Informationsmanagement

Der auf der Startseite eines Unternehmens implementierte Informationsstrom stellt die Aktivitäten der eigenen Benutzer im System der Online-Frachtenbörse dar. Dieser „Newsfeed" könnte dahinge-

---

1) Vgl. Carneiro/Barazi (2010), S. 234.

2) Zur Wichtigkeit der Benutzeroberfläche vgl. Klippert/Kowalski/Bruns (2010), S. 59.

3) Vgl. Klippert/Kowalski/Bruns (2010), S. 59.

4) Im Rahmen der Implementierung des Softwareprototyps wurden die modernen Internetbrowser „Apple Safari", „Google Chrome" und „Mozilla Firefox" eingesetzt. Insbesondere in älteren Internetbrowsern (bspw. "Microsoft Internet Explorer" in der Version 7.0 oder früher) dürfte die Benutzeroberfläche aufgrund noch fehlender Standardimplementierungen weniger ansprechend präsentiert werden, so dass hier gegebenenfalls Anpassungen nötig wären.

hend erweitert werden, dass er nicht nur die Aktivitäten der eigenen Benutzer zeigt, sondern bspw. auch das Veröffentlichen einer verfassten Bewertung auf dem Profil eines Geschäftspartners. Darüber hinaus könnte der Newsfeed die Möglichkeit bieten, bestimmte Aktionen anderer Anbieter, wie bspw. das Einstellen von Inseraten, zu abonnieren, um so leichter Notiz von Inseraten dieser Anbieter nehmen zu können.[1] Anfragen an die Anbieter sollten ebenfalls direkt über den Newsfeed möglich sein. Es sollte die Möglichkeit geben, diese Dienste als Anbieter zu untersagen.

Die Implementierung eines sozial vernetzteren Informationsmanagements und die damit verbundene bessere Aufbereitung und Verteilung von vorhandenen Informationen innerhalb des Systems der Online-Frachtenbörse kann somit als Weiterentwicklungspotential identifiziert werden.

10.3.3 Angepasste Benutzeroberflächen des Frontends

Im Rahmen der Adminstration der Online-Frachtenbörse über das Backend bietet der Softwareprototyp verschiedene Benutzeroberflächen je nach Benutzerrolle des angemeldeten Administrators. Es wäre denkbar, dass die Übertragung dieser Funktionalität auf das Frontend einen Zusatznutzen für Anbieter darstellen kann, wenn die verschiedenen Bedürfnisse der verschiedenen Benutzer identifiziert wurden. Hier muss die weitere Anforderungsanalyse zeigen, welche Aufgaben der Dienstleister sich im System der Online-Frachtenbörse voneinander abgrenzen lassen. So erscheint es plausibel, dass ein Mitarbeiter eines Unternehmens rein zu Kommunikationszwecken im System der Online-Frachtenbörse registriert ist und dort primär Neukundenakquise und Recherche betreibt sowie Bewertungen verfasst und veröffentlicht. Sollte diese Rolle eines Mitarbeiters sich als typisch erweisen, könnte für diese Benutzerrolle bspw. eine spezielle Startseite entworfen werden, welche die kommunikativen Elemente des Frontends zentralisiert.

Somit kann die Bereitstellung angepasster Benutzerflächen im Frontend als Weiterentwicklungspotential identifiziert werden.

10.4 Weiterentwicklungspotential des Backends

10.4.1 Benutzerverwaltung innerhalb des Backends

Die implementierte Benutzerverwaltung des Backends ermöglicht die Bearbeitung und Löschung von Benutzern. Nicht ermöglicht wird die zeitweise Sperrung von einzelnen Benutzern oder Unternehmen.

Eine weitere Verbesserungsmöglichkeit wäre die gleichzeitige Bearbeitung des Benutzer- und Personenobjekts in derselben Ansicht sowie die gleichzeitige Bearbeitung mehrerer Datensätze, bspw. das Zuweisen einer Benutzerrolle zu mehreren Benutzern.

Somit kann die Überarbeitung der Benutzerverwaltung innerhalb des Backends als Weiterentwicklungspotential identifiziert werden.

10.4.2 Nutzungsstatistiken

Die implementierten Nutzungsstatistiken des Backends ermöglichen umfangreiche Auswertungen über das Verhalten der Benutzer. Die bestehenden Möglichkeiten könnten jedoch erweitert werden,

1) Das Abonnieren der Inserate einzelner Anbieter entspräche darüber hinaus einem "white-listing" (dem Führen einer Positiv-Liste), vgl. Klippert/Kowalski/Bruns (2010), S. 49.

indem die grafischen Darstellungen, bspw. mit interaktiven Diagrammen, verbessert und die zugrundeliegenden Statistiken zum Herunterladen angeboten werden.

Falls im Rahmen einer umfangreicheren Protokollierung mehr Daten über das Benutzerverhalten erhoben werden, könnten diese auch zu detaillierteren Nutzungsstatistiken führen.

Somit kann die Überarbeitung der Nutzungsstatistiken als Weiterentwicklungspotential identifiziert werden.

11 Fazit und Ausblick

Das wissenschaftliche Problem des vorliegenden Projektberichts lag in der teilweisen Implementierung der vorliegenden Erkenntnisse über die Anforderungen an einen Softwareprototyp einer Online-Frachtenbörse für den Schienengüterverkehr, um mithilfe des resultierenden Softwareprototyps die weitere Anforderungsanalyse zu unterstützen.

Es wurde dargelegt, warum ein evolutionärer Prototyp geeignet ist, die weitere Grundlagenforschung im Bereich „Online-Frachtenbörse für den Schienengüterverkehr" zu unterstützen, warum das Primärziel dieses Projektberichts in der Unterstützung der weiteren Anforderungsanalyse besteht und warum die gewählten, teils modernen und flexiblen Arbeitstechniken die Erfüllung dieses Ziels unterstützen.

Neben dem Leistungsumfang des vorliegenden Softwareprototyps wurden auch dessen Defizite und wichtige Weiterentwicklungspotentiale beschrieben. Insbesondere die implementierten Schnittstellen wurden hierbei detailliert beschrieben und einer kritischen Prüfung unterzogen.

So können die Genauigkeit und Effizienz der implementierten Algorithmen, bspw. in der Such- oder Matching-API, als verbesserungswürdig angesehen werden.

Jedoch wurden im Rahmen dieses Projektberichts Strukturen in Form wohldefinierter Schnittstellen geschaffen, welche die Funktionalitäten der Algorithmen zu abstrakten Werkzeugen bündeln und durch diese Bündelung eine Implementierung genauerer und effizienterer Algorithmen unterstützen.

Der hierdurch geschaffene Mehrwert für die Grundlagenforschung im Bereich „Online-Frachtenbörse für den Schienengüterverkehr" überwiegt zum Zeitpunkt der Erstellung dieses Projektberichts nach Ansicht der Autoren den Mehrwert, den ein weniger wohlstrukturierter Softwareprototyp mit genaueren und effizienteren Algorithmen hätte.[1]

Die Weiterentwicklung der vorliegenden Implementierung des Softwareprototyps zu einem marktreifen Produkt wäre somit realisierbar.

Jedoch erscheint es ebenso plausibel, den Softwareprototyp nach einer oder mehrerern Testphasen mithilfe der hierbei gewonnenen Erkenntnisse über die Anforderungen von Benutzern an das System der Online-Frachtenbörse und seine Systemkomponenten[2] unter Verwendung anderer, noch zu bestimmender Arbeitstechniken neu zu implementieren.

Somit ist im Rahmen der weiteren Anforderungsanalyse erst noch abzuwägen, ob unter anforderungsanalytischen, informationstechnischen und betriebswirtschaftlichen Aspekten die sukzessive Weiterentwicklung des bestehenden Quelltextes einer Neuimplementierung des Softwareprototyps vorzuziehen ist.

---

1) Begründet wird dies durch folgende Plausibilitätsüberlegung: Die implementierten Algorithmen funktionieren und liefern nachvollziehbar plausible Ergebnisse. Allerdings können bei entsprechender Recherche genauere und effizientere Algorithmen identifiziert werden, deren zeitaufwendige Implementierung bspw. die Matching-Ergebnisse signifikant verbessern würde. Im Rahmen der Implementierung eines ersten Softwareprototyps zur ersten Anforderungsanalyse erscheinen jedoch ein gesamtheitlich durchdachtes System flexibler Strukturen und wohldefinierter Schnittstellen der weiteren angewandten Forschung dienlicher als ein starrer "Wegwerfprototyp", in dem einzelne Teilfunktionalitäten bereits marktreif implementiert wurden.

2) Gemeint sind hier bspw. Zielgruppen, Leistungsumfang oder Nutzungsverhalten.

12 Literaturverzeichnis

Ammelburger/Scherer (2008)
Ammelburger, D.; Scherer, R.: Webentwicklung mit CakePHP. 1. Auflage, Sebastopol 2008.

Anderson/Lehnhardt/Slater (2010)
Anderson, J.; Lehnhardt, J.; Slater, N.: CouchDB: The Definitive Guide. 1. Auflage, Sebastopol 2010.

Andrew/Shafer (2006)
Andrew, R.; Shafer, D.: CSS – Anspruchsvolle Websites mit Cascading Stylesheets – Grundlagen, Designtechniken und Referenz. 2. Auflage, Collingwood 2006.

Aurum/Wohlin (2005)
Aurum, A.; Wohlin, C.: Engineering and Managing Software Requirements. 1. Auflage, Berlin, Heidelberg 2005.

Bell (2005)
Bell, D.: Software Engineering for Students – A Programming Approach. 4. Auflage, Essex 2005.

Berenbach et al. (2009)
Berenbach, B.; Paulish, D.; Kazmeier, J.; Rudorfer, A.: Software & Systems Requirements Engineering – In Practice. 1. Auflage, New York 2009.

Berlin/Rooney (2006)
Berlin, D.; Rooney, G.: Practical Subversion. 2. Auflage, New York 2006.

Berube (2008)
Berube, D.: Practical Reporting with Ruby and Rails. 1. Auflage, New York 2008.

Bray (2002)
Bray, I.: An Introduction to Requirements Engineering. 1. Auflage, Essex 2002.

Carlson/Richardson (2006)
Carlson, L.; Richardson, L.: Ruby Cookbook – Recipes for Object-Oriented Scripting. 1. Auflage, Sebastopol 2006.

Carneiro/Barazi (2010)
Carneiro, C.; Barazi, R.: Beginning Rails 3. 1. Auflage, New York 2010.

Chacon (2009)
Chacon, S.: Pro Git. 1. Auflage, New York 2009.

Chak (2009)
Chak, D.: Enterprise Rails. 1. Auflage, Sebastopol 2009.

Cooper (2009)
Cooper, P.: Beginning Ruby: From Novice to Professional. 2. Auflage, New York 2009.

Crockford (2008)
Crockford, D.: JavaScript: The Good Parts – Unearthing the Excellence in JavaScript. 1. Auflage, Sebastopol 2008.

Dobbs-Sciortino (2006)
Dobbs-Sciortino, A.: Mongrel. 1. Auflage, Sebastopol 2006.

Ediger (2008)
Ediger, B.: Advanced Rails. 1. Auflage, Sebastopol 2008.

Ferrara/MacDonald (2002)
Ferrara, A.; MacDonald, M.: Programming .NET Web services. 1. Auflage, Sebastopol 2002.

Flanagan (2006)
Flanagan, D.: JavaScript – The Definitive Guide. 5. Auflage, Sebastopol 2006.

Flanagan/Matsumoto (2008)
Flanagan, D.; Matsumoto, Y.: The Ruby Programming Language. 1. Auflage, Sebastopol 2008.

Gilmore (2008)
Gilmore, W.: Beginning PHP and MySQL: From Novice to Professional. 3. Auflage, New York 2008.

Hansson (2008)
Hansson, D.: Riding Rails: Merb gets merged into Rails 3! Im Internet unter der URL http://weblog.rubyonrails.org/2008/12/23/merb-gets-merged-into-rails-3, Zugriff am 27.09.2010.

Heaton (2008)
Heaton, J.: Introduction to Neural Networks with Java. 2. Auflage, Chesterfield 2008.

Henderson (2006)
Henderson, C.: Building Scalable Web Sites. 1. Auflage, Sebastopol 2006.

Hudson (2006)
Hudson, P.: PHP in a Nutshell. 1. Auflage, Sebastopol 2006.

Hull/Jackson/Dick (2005)
Hull, E.; Jackson, K.; Dick, J.: Requirements Engineering. 2. Auflage, London 2005.

Jones (2007)
Jones, C.: Estimating Software Costs – Bringing Realism to Estimating. 2. Auflage, New York 2007.

Kannengiesser (2007)
Kannengiesser, M.: Objektorientierte Programmierung mit PHP 5. 1. Auflage, Poing 2007.

Katz (2008)
Katz, Y.: Rails and Merb Merge « Katz Got Your Tongue? Im Internet unter der URL http://yehudakatz.com/2008/12/23/rails-and-merb-merge/, Zugriff am 27.09.2010.

Kersken (2007)
Kersken, S.: Praxiswissen Ruby. 1. Auflage, Köln 2007.

Klippert/Kowalski/Bruns (2010)
Klippert, S.; Kowalski, M.; Bruns, A.: Anforderungsanalyse für eine Online-Frachtenbörse im Eisenbahngüterverkehr – Entwicklung einer Anforderungsspezifikation aus betriebswirtschaftlicher Perspektive. Projektbericht des INTERREG-IVB-NWE-Projekts CODE24 Nr. 2, Universität Duisburg-Essen, Institut für Produktion und Industrielles Informationsmanagement, Essen 2010.

Kofler/Kramer (2005)
Kofler, M.; Kramer, D.: The Definitive Guide to MySQL 5. 1. Auflage, New York 2005.

Kreibich (2010)
Kreibich, J.: Using SQLite. 1. Auflage, Sebastopol 2010.

Laurent/Dumbill (2009)
Laurent, S.; Dumbill, E.: Learning Rails. 1. Auflage, Sebastopol 2009.

Laurie/Laurie (2003)
Laurie, B.; Laurie, P.: Apache – Das umfassende Handbuch. 2. Auflage, Sebastopol 2003.

Lennon (2009)
Lennon, J.: Beginning CouchDB. 1. Auflage, New York 2009.

Lerdorf/Tatroe/MacIntyre (2006)
Lerdorf, R.; Tatroe, K.; MacIntyre, P.: Programming PHP. 2. Auflage, Sebastopol 2006.

Liberty/MacDonald (2009)
Liberty, J.; MacDonald, B.: Learning C# 3.0. 1. Auflage, Sebastopol 2009.

Lippert (2004)
Lippert, E.: Rumours of VBScript's Death Have Been Greatly Exaggerated. Im Internet unter der URL http://blogs.msdn.com/b/ericlippert/archive/2004/04/09/110508.aspx, Zugriff am 21.09.2010.

Loeliger (2009)
Loeliger, J.: Version Control with Git – Powerful Techniques for Centralized and Distributed Project Management. 1. Auflage, Sebastopol 2009.

Marshall/Pytel/Yurek (2007)
Marshall, K.; Pytel, C.; Yurek, J.: Pro Active Record for Ruby: Databases with Ruby on Rails. 1. Auflage, New York 2007.

Mauthe/Thomas (2004)
Mauthe, A.; Thomas, P.: Professional Content Management Systems – Handling Digital Media Assets. 1. Auflage, West Sussex 2004.

Merkel/Kromer (2002)
Merkel, H.; Kromer, S.: Virtuelle Frachtbörsen – Top oder Flop? In: Hossner, R. (Hrsg.): Jahrbuch der Logistik 2002, Handelsblatt Fachverlag: Düsseldorf 2002, S. 82-87.

Meyer (2007)
Meyer, E.: CSS – The Definitive Guide. 3. Auflage, Sebastopol 2007.

Mornini/Loy (2006)
Mornini, T.; Loy, M.: Capistrano and the Rails Application Lifecycle. 1. Auflage, Sebastopol 2006.

Morsy/Otto (2008)
Morsy, H.; Otto, T.: Ruby on Rails 2 – Das Entwickler-Handbuch. 1. Auflage, Bonn 2008.

Musciano/Kennedy (2007)
Musciano, C.; Kennedy, B.: HTML & XHTML – The Definitive Guide. 6. Auflage, Sebastopol 2007.

Nash (2010)
Nash, T.: Accelerated C# 2010. 1. Auflage, New York 2010.

Niemeyer/Knudsen (2005)
Niemeyer, P.; Knudsen, J.: Learning Java. 3. Auflage, Sebastopol 2005.

Owens (2006)
Owens, M.: The Definitive Guide to SQLite. 1. Auflage, New York 2006.

Pankratz (2003)
Pankratz, G.: Zweiseitige kombinatorische Auktionen in elektronischen Transportmärkten, Potenziale und Probleme. Diskussionsbeitrag 351, Diskussionsbeiträge des Fachbereichs Wirtschaftswissenschaft der FernUniversität. Hagen, 2003. Im Internet unter der URL ftp://ftp.fernuni-hagen.de/pub/fachb/wiwi/winf/forschng/publi/gp_p6.pdf, Zugriff am 29.08.2010.

Pilato/Collins-Sussman/Fitzpatrick (2008)
Pilato, C.; Collins-Sussman, B.; Fitzpatrick, B.: Version Control with Subversion. 2. Auflage, Sebastopol 2008.

Pilato/Collins-Sussman/Fitzpatrick (2009)
Pilato, C.; Collins-Sussman, B.; Fitzpatrick, B.: Subversion 1.6 Official Guide – Version Control with Subversion. 1. Auflage, Palo Alto 2009.

Pohl/Rupp (2009)
Pohl, K.; Rupp, C.: Basiswissen Requirements Engineering – Aus- und Weiterbildung nach IREB-Standard zum Certified Professional for Requirements Engineering Foundation Level. 1. Auflage, Heidelberg 2009.

Reese et al. (2002)
Reese, G.; Yarger, R.; King, T.; Williams, H.: Managing and Using MySQL. 2. Auflage, Sebastopol 2002.

Richardson/Ruby (2007)
Richardson, L., Ruby, S.: RESTful Web Services. 1. Auflage, Sebastopol 2007.

Rothfuss/Ried (2003)
Rothfuss, G.; Ried, C.: Content Management mit XML. 2. Auflage, Berlin, Heidelberg 2003.

Rupp (2004)
Rupp, C.: Requirements-Engineering und -Management – Professionelle, iterative Anforderungsanalyse für die Praxis. 3. Auflage, München, Wien 2004.

Schwartz et al. (2008)
Schwartz, B.; Zaitsev, P.; Tkachenko, V.; Zawodny, J. D.; Lentz, A; Balling, D. J.: High Performance MySQL. 2. Auflage, Sebastopol 2008.

Solis (2008)
Solis, D.: Illustrated C# 2008. 1. Auflage, New York 2008.

Sommerville (2007)
Sommerville, I.: Software Engineering. 8. Auflage, Essex 2007.

Stefanov (2010)
Stefanov, S.: JavaScript Patterns. 1. Auflage, Sebastopol 2010.

Suh (2005)
Suh, W.: Web Engineering: Principles and Techniques. 1. Auflage, London 2005.

Tahaghoghi/Williams (2007)
Tahaghoghi, S.; Williams, H.: Learning MySQL. 1. Auflage, Sebastopol 2007.

Tate/Carlson/Hibbs (2009)
Tate, B.; Carlson, L.; Hibbs, C.: Rails – Up and Running. 2. Auflage, Sebastopol 2009.

W3C (2010)
W3C: HTML & CSS - W3C. Im Internet unter der URL http://www.w3.org/standar-ds/webdesign/htmlcss, Zugriff am 19.08.2010.

Watson (2009)
Watson, M.: Scripting Intelligence: Web 3.0 Information Gathering and Processing. 1. Auflage, New York 2009.

Williams (2007)
Williams, J.: Rails solutions – Ruby on Rails made easy. 1. Auflage, New York 2007.

Anhang

Anhang

Dokumentation der Datenbanktabellen

Im Folgenden ist die Dokumentation der Datenbanktabellen, die nicht im Text besprochen wurden, angehängt.

Ihre Auflistung dient einer möglichst vollständigen Beschreibung des implementierten Softwareprototypen.

Die folgenden Tabellen zeigen für die jeweiligen Objekte des Softwareprototypen, aus welchen Feldern die zugehörige Datenbanktabelle besteht, welchen Datentyps sie ist und einen Beispielwert.

Name	**Typ**	**Beschreibung**	**Beispiel**
name	String	Name der Konfigurationsvariable	„language“
value	Text	Wert der Konfigurationsvariable	„de“

Tabelle 12: Struktur der „app_configs“-Datenbanktabelle

Name	**Typ**	**Beschreibung**	**Beispiel**
name	String	Name der Firma	„Mustermann GmbH“
phone	String	Festnetznummer	„0123-5436895“
fax	String	Faxnummer	„0123-5436896“
mobile	String	Mobilfunknummer	„0123-5436897“
email	String	E-Mail-Adresse	„mm@example.org“
website	String	URL der Webseite	„www.example.org“
address	String	Addresse	„Musterstr. 21“
address2	String	Addresse (Fortsetzung)	„3. Etage“
zip	String	Postleitzahl	„12345“
city	String	Ort	„Musterstadt“
country	String	Land	„Deutschland“
contact_person_id	Integer	Verweis auf die Person, die Ansprechpartner sein soll	1

Tabelle 13: Struktur der „companies“-Datenbanktabelle

Name	**Typ**	**Beschreibung**	**Beispiel**
user_id	Integer	Verweis auf den Benutzer, dem das Objekt gehört	1
company_id	Integer	Verweis auf die Firma, zu der das Objekt gehört	1
origin_site_info_id	Integer	Verweis auf den Startort	34
destination_site_info_id	Integer	Verweis auf den Zielort	35
weight	Integer	Gesamtgewicht (in t)	52000
loading_meter	Integer	Lademeter	30
hazmat	Boolean	Ist das Gut ein Gefahrengut?	true
transport_type	String	Art der Wagen	single_wagon
wagons_provided_by	String	Wer stellt die Wagen bereit?	railway
desired_proposal_type	String	Welche Art von Angebot wird gewünscht?	ton_price
contact_person_id	Integer	Verweis auf die Person, die Ansprechpartner sein soll	1
transport_weight	Integer	Gewicht pro Transport (in t)	1000
transports_per_year	Integer	Transporte pro Jahr	52
paying_freight	String	Frachtzahler	„Sender“

Tabelle 14: Struktur der „freights“-Datenbanktabelle

Name	**Typ**	**Beschreibung**	**Beispiel**
user_id	Integer	Verweis auf den Benutzer, dem das Objekt gehört	1
company_id	Integer	Verweis auf die Firma, zu der das Objekt gehört	1
origin_site_info_id	Integer	Verweis auf den Startort	34
destination_site_info_id	Integer	Verweis auf den Zielort	35
weight	Integer	Gesamtgewicht (in t)	52000
loading_meter	Integer	Lademeter	30
hazmat	Boolean	Ist das Gut ein Gefahrengut?	true
transport_type	String	Art der Wagen	single_wagon
contact_person_id	Integer	Verweis auf die Person, die Ansprechpartner sein soll	1
transport_weight	Integer	Gewicht pro Transport (in t)	1000
transports_per_year	Integer	Transporte pro Jahr	52
paying_freight	String	Frachtzahler	„Sender“

Tabelle 15: Struktur der „loading_spaces“-Datenbanktabelle

Name	Typ	Beschreibung	Beispiel
a_type	String	Verweist auf den Typ des A-Objekts	„Freight“
a_id	Integer	Verweist auf die ID des A-Objekts	182
b_type	String	Verweist auf den Typ des B-Objekts	„LoadingSpace“
b_id	Integer	Verweist auf die ID des B-Objekts	98
result	Float	Das Resultat des Vergleichs	0.765

Tabelle 16: Struktur der „matching_recordings“-Datenbanktabelle

Name	Typ	Beschreibung	Beispiel
first_name	String	Vorname	„Max“
last_name	String	Nachname	„Mustermann“
gender	String	Geschlecht	„male“
job_description	String	Dienstbezeichnung	„Vertriebsleiter Nord“
phone	String	Festnetznummer	„0123-5436895“
fax	String	Faxnummer	„0123-5436896“
mobile	String	Mobilfunknummer	„0123-5436897“
email	String	E-Mail-Adresse	„mm@example.org“
website	String	URL der Webseite	„www.example.org“
locale	String	Sprache, in der die Person die Benutzeroberfläche nutzt	„de“

Tabelle 17: Struktur der „people“-Datenbanktabelle

Name	Typ	Beschreibung	Beispiel
name	String	Name der Benutzerrolle	„company_admin“

Tabelle 18: Struktur der „user_roles“-Datenbanktabelle

Name	Typ	Beschreibung	Beispiel
user_id	Integer	Verweis auf den Benutzer, dem die Benutzerrolle gehört	23
user_role_id	Integer	Verweis auf die Benutzerrolle, die dem Benutzer gehört	11

Tabelle 19: Struktur der „user_roles_users“-Datenbanktabelle

Name	Typ	Beschreibung	Beispiel
login	String	Benutzername	„max.mustermann“
email	String	E-Mail	„mm@example.org“
crypted_password	String	Verschlüsseltes Passwort	„55024db979...“
password_salt	String	Geheimer Passwortschlüssel	„55024db979...“
login_count	Integer	Anzahl Logins	120
failed_login_count	Integer	Fehlgeschlagene Loginversuche	13
current_login_ip	String	Aktuelle IP	„192.188.142.11“
last_login_ip	String	Letzte IP	„192.188.142.11“
company_id	Integer	Verweis auf die Firma, zu der der Benutzer gehört	1
person_id	Integer	Verweis auf die Person, zu der der Benutzer gehört	1
api_key	String	Alphanumberischer Schlüssel zur Ansteuerung der XML/JSON-API	„55024db979...“
posting_type	String	Verweis auf die Inserate, die der Benutzer einstellt	„Freight“

Tabelle 20: Struktur der „users“-Datenbanktabelle

Dokumentation des Quelltextes

Im Folgenden ist die Dokumentation des Quelltextes angehängt.

Ihre Auflistung dient einer möglichst vollständigen Beschreibung des implementierten Softwareprototypen.

Diese ist in englischer Sprache verfasst, um einer möglichst breiten Entwicklerbasis die Arbeit mit dem Quelltext zu ermöglichen.

Rails Application Documentation

Build this documentation with either rake doc:* for standard rails doc output, or with the rake doc:ajax task to get nicer doc layout using breakpointer‘s ajax-rdoc (which is available at github.com/breakpointer/ajax-rdoc and has to be installed separately).

Installation

First, run:

```
bundle install
```

Unfortunately, the nokogiri gem needs to be installed separately.

Follow these instructions:

nokogiri.org/tutorials/installing_nokogiri.html

The next step is to run these rake tasks:

```
rake db:migrate
rake db:seed
```

Now the database is fully migrated and seeded. Start the webserver with

```
rails server
```

A setup screen will be ready at localhost:3000

Starting points for further reading

Search API

For any information on the search functionality of the app, take a look at the Search module.

Matching API

For any information on the matching of Freight and LoadingSpace objects, take a look at the Matching module.

XML/JSON-API

For any information on the XML/JSON-API access on controllers, take a look at the RemoteController class.

Class: ActionRecording

ActionRecording objects inherit from Recorder::Recording and are logs of user actions in the app.

They belong to a user and his company, so company or user specific reports can be created.

GeneralObserver creates ActionRecording objects whenever an user editable record is created, updated or deleted in the app.

Class: ActiveRecord::Base

Public Class methods

brackets_find_by*(attribute_name)*

Adds a convenient [] find_by to the model utilizing the given attribute and returning the first record matching the condition. Therefore this is best used on attributes that go through validates_uniqueness_of.

```
class Country < ActiveRecord::Base
  brackets_find_by :iso_code
end

Country[:de]
  # => #<Country id: 1, name: "Germany", iso_code: "de">
```

human_attribute_value*(attribute_name, value, i18n_opts = {})*

Returns the localized version of the given attribute and its value.

```
Freight.human_attribute_value(:transport_type, 'single_wagon')
  # => 'Single Wagon'
I18n.t('activerecord.human_attribute_values.freight.
           transport_type.single_wagon')
  # => 'Single Wagon'
```

searchable*(opts = {})*

Adds a model to the search index.

```
class User < ActiveRecord::Base
  searchable
end
```

Public Instance methods

attributes_filled*()*

Returns how many of the attributes are not blank.

```
obj.attributes_filled # => 0.65
```

belongs_to?*(user = current_user)*

Returns `true`, if the record belongs to a certain user.

```
obj.belongs_to?(@user) # => true
```

human_attribute_value*(attribute_name, i18n_opts = {})*

Returns the localized version of the attribute‘s value.

```
freight[:transport_type]
  # => 'single_wagon'
freight.human_attribute_value(:transport_type)
  # => 'Single Wagon'
freight.human_attribute_value(:transport_type, :locale => :de)
  # => 'Einzelwagen'
```

mine?*(user = current_user)*

Alias for belongs_to?

Module: ActiveRecord::HasLocalizedInfos

This module can be included in ActiverRecord::Base objects to provide the functionality of LocalizedInfo attributes to that object.

Example:

```
 class Person < ActiveRecord::Base
   include ActiveRecord::HasLocalizedInfos
 end

person = Person.new
infos = [
    {:name => 'misc_text', :lang => 'de', :text => 'Etwas Text...'},
    {:name => 'misc_text', :lang => 'en', :text => 'Some text...'}
  ]
person.localized_infos!(infos)
```

Module: ActiveRecord::HasLocalizedInfos::InstanceMethods

Public Instance methods

localized_info*(name, lang = I18n.default_locale)*

Returns the localized info object for the given name and language.

```
obj.localized_info(:sample)
obj.localized_info(:sample, :de)
```

localized_infos!*(array_of_hashes)*

Saves (or updates) the provided localized infos.

```
infos = [
    {:name => 'misc_text', :lang => 'de', :text => 'Etwas Text...'},
    {:name => 'misc_text', :lang => 'en', :text => 'Some text...'}
  ]
obj.localized_infos!(infos)
```

update_localized_infos*()*

Updates all localized_infos belonging to this ActiveRecord.

Class: AppConfig

AppConfig objects store system specific configuration values for the application.

Reading

Read any value from the database (or the default config yaml, if it is not yet in the db) by using the [] accessor.

```
AppConfig[:language]  # => "en"
```

Storing

Store any value in the database by using the []= accessor.

```
AppConfig[:language] = 'de'  # => "de"
AppConfig[:some_option] = 'some value' # => "some value"
```

Public Class methods

AppConfig[key] # => Object

Returns the value stored for key in the database or its default value.

```
AppConfig[:language]  # => "en"
```

AppConfig[key] = value

Stores the value for key in the database.

```
AppConfig[:language] = 'de'  # => "de"
```

Class: ApplicationController

Private Class methods

login_required*(opts = {})*

Use this in a controller to restrict access.

```
class UsersController < ApplicationController
  login_required :only => [:edit, :update, :show]
end
```

ownership_required*(opts = {})*

Use this in a controller to restrict access to owners.

role_or_ownership_required*(roles, opts = {})*

Use this in a controller to restrict access to either users of certain roles (e.g. admins) or the rightful owner of an object.

```
class PostingController < ApplicationController
  role_or_ownership_required [:posting_admin, :administrator]
end
```

role_required*(roles, opts = {})*

Use this in a controller to restrict access to users of certain roles (e.g. admins).

```
class Admin::BaseController < ApplicationController
  role_required :administrator
end
```

same_company_required*(opts = {})*

Private Instance methods

controller_catalog()

Returns the i18n catalog path for the current controller.

```
class UsersController < ApplicationController
  def index
    controller_catalog # => 'users'
  end
end

class Admin::UsersController < Admin::BaseController
  def index
    controller_catalog # => 'admin.users'
  end
end
```

current_company()

Returns the Company object of the currently logged in user or `nil` if no user is logged in.

current_person()

Returns the Person object of the currently logged in user or `nil` if no user is logged in.

current_user()

Returns the User object of the currently logged in user or `nil` if no user is logged in.

demo_mode?()

Returns `true` if the application is running in demo mode.

Module: ApplicationHelper

The ApplicationHelper provides basic helper methods for all views.

Public Instance methods

admin?*()*

Returns `true` if the current controller is an Admin::BaseController

badge_for*(count)*

Renders a badge labelled with count, unless the given count is zero.

box*(title = nil, &block)*

clear_both*()*

Returns a DIV tag that clears floating.

collection_choices*(model, attribute_name, const = nil)*

Returns the collection of localized choices for a given attribute.

Example:

```
collection_choices(Person, :gender)
```

This will look up `Person::GENDER_CHOICES` and return the keys and localized values.

contact_info(object, attr) # => String

Returns a formatted string version of the attribute.

Examples:

```
<%= contact_info(@company, :phone) %>
  # => '+49 (0) 234 366 98007'

<%= contact_info(@company, :website) %>
  # => '\url{www.example.org}'
```

controller?(name) # => **boolean**

Returns `true` if c is the current controller. Example:

```
<%= controller?(:root) %>
  # => true
```

format_multiline_input*(text)*

Returns a HTML formatted version of `text`.

Example:

```
<%= format_multiline_input("First line.\nSecond Line.") %>
  # => "First line.\\Second line."
```

highlight_in_search?(result) # => **boolean**

Returns `true` if the given result should be highlighted, i.e. if the company behind the posting has a certain number of positive reviews.

humanize_recording*(rec)*

Returns a humanized string for the given ActionRecording.

link_back*(text = t("common.link_back"))*

Returns a link back to the last visited page with a localized caption.

link_btn*(text, path, opts = {})*

Renders a linked button (optionally with an icon).

link_to_item*(item)*

Renders a link to the given item.

link_to_result*(t, result)*

Renders a link to the given result.

localized_info*(obj, name, lang = I18n.locale)*

Returns a formatted string for the associated LocalizedInfo object.

localized_info_fields*(f, name, locales = I18n.available_locales)*

only_some_attributes_filled?*(ar)*

REturns `true`, if only some attributes are not blank for the given ActiveRecord object.

phone_number*(nr)*

Tries to render a phone number in a certain format. Returns the reformatted number.

posting_freights?*()*

Returns `true`, if the current_user is posting freights.

render_company_info*(company = resource.company)*

Renders a partial with the contact information for the given company. Example:

```
<%= render_person_info current_company %>
```

render_partial*(partial, options = {})*

Renders a partial taking into account the current user‘s UserRole objects.

render_person_info*(person)*

Renders a partial with the contact information for the given person. Example:

```
<%= render_person_info current_person %>
```

text_with_badge*(snippet, count)*

Renders a text next to a badge.

Class: Company

Company objects represent the organisation of a User.

Each Company has different types of users, e.g. admins.

Public Instance methods

approved_reviews*()*

Returns all approved reviews for the company.

ensure_admin*()*

Ensures there is at least one `:company_admin` left in this company. If no admin can be found, the first user of the company is assigned the admin role.

unapproved_reviews*()*

Returns all unapproved reviews for the company.

Class: Contact

Public Instance methods

contact*(text, user)*

Returns a contact mail from the given user to the support email address.

Class: ErrorMessages

ErrorMessages objects are used to render an array of error messages as XML and JSON.

Public Class methods

ErrorMessages.new(messages)

Takes an array of strings and creates an ErrorMessages object.

```
arr = ['Unauthorized access']
ErrorMessages.new(arr)
```

Public Instance methods

to_json*(*args)*

Renders the given messages as JSON.

to_xml*(*args)*

Renders the given messages as XML.

Class: Freight

Freight objects are postings for freight.

They have two SiteInfo objects attached for their origin and destination and several LocalizedInfo objects to describe the posting.

Public Instance methods

calc_matchings!*()*

Calculates and saves all matching results for the freight.

matching_loading_spaces*(limit = 3)*

Returns the given number of matching loading space objects for the freight.

matching_objects*(limit = 3)*

Alias for matching_loading_spaces

Class: GeneralObserver

The GeneralObserver inherits Recorder::Observer to provide basic recording functionality.

It watches all user editable models in the app.

Class: LoadingSpace

LoadingSpace objects are postings for loading space.

They have two SiteInfo objects attached for their origin and destination and several LocalizedInfo objects to describe the posting.

Public Instance methods

matching_freights*(limit = 3)*

Returns the given number of matching freight objects for the loading space.

matching_objects*(limit = 3)*

Alias for matching_freights

Class: LocalizedInfo

LocalizedInfo objects contain localized text snippets that can be associated with other objects, e.g. freights, people etc.

```
opts = {
  :name => 'misc_text',
  :lang => 'de',
  :text => 'Etwas Text...',
  :item => Person.find(1)
}
LocalizedInfo.create(opts)
```

Public Instance methods

info.update_or_destroy!

Saves or destroys the object, whether its text is present or blank.

Module: Matching

The Matching module provides a set of classes and methods to match objects. On top of this, it provides an extendable generic API for matching Freight and LoadingSpace objects (see compare_freight_and_loading_space method).

Public Class methods

Match.compare_freight_and_loading_space(freight, space) # => **Float**
Match.fls(freight, space) # => **Float**

Returns the likeness of a Freight and a LoadingSpace object.

```
Match.fls(Freight.first, LoadingSpace.first) # => 0.977920227850516
```

Module: Matching::Compare

The Compare module provides a set of classes and methods to match objects like Strings, Numbers and Dates.

Class: Matching::Compare::Base

Compare objects compare two objects A and B based on their type/class.

Creation

Compare objects accept two constructor parameters for the A and the B object.

```
compare = Compare::String.new('one string', 'another string')
compare.result # => 0.6428...
```

Conditions

By default, a compare object compares copies of the entire objects it is passed. It is also possible to only compare certain attributes of an object.

```
class UserComparer < Matching::Compare::Base
  compare :gender, :weight
end
```

Thresholds can be used to ensure that only objects who meet certain criteria are considered alike.

```
class UserComparer < Matching::Compare::Base
  compare :weight, :threshold => 10
  # => User A can be 10 kilos heavier or lighter than user B

  compare :weight, :threshold => 0.05
  # => User A can be 5% heavier or lighter than user B

  compare :weight, :threshold => {:up => 0, :down => 0.1}
  # => User A can be 10% lighter than user B, but not any heavier.

  compare :weight, :threshold => :perfect
  # => User A and B have to have the same weight
end
```

All object-pairs not meeting the threshold criteria are automatically assigned a result of 0.0 (not matching at all).

Overwriting defaults

Blocks can be used to override the default comparisions.

Example:

```
class UserCompanyComparer < Matching::Compare::Base
  # Do not compare the email with the default String processor
  # but compare the email hosts and eliminate the pair if they
  # are not matching.
  compare :email do |a, b|
    email_domain = /[^@]+$/
    a[email_domain] == b[email_domain]
  end
end
```

Public Class methods

compare(*attributes, options = {}, &block)

Specifies one or more attribute(s) that will be compared using the defined options and the block, if given.

Options

- `:as` - A Symbol identifying the Comparer class to be used (e.g. `:String`)

```
class UserComparer < Matching::Compare::Base
  compare :created_at, :as => :Time
end
```

- `:threshold` - If the attribute of the B object differs more

than the given threshold the comparison fails, resulting in a 0.0 match. `:up` and `:down` options are available as well. Floats are interpreted as relative, Fixnums as absolute thresholds.

```
class UserComparer < Matching::Compare::Base
  compare :weight, :threshold => 10
  # =>  User A can be 10 kilos heavier or lighter than user B

  compare :weight, :threshold => 0.05
  # =>  User A can be 5% heavier or lighter than user B
```

```
    compare :weight, :threshold => {:up => 0, :down => 0.1}
    # =>  User A can be 10% lighter than user B, but not any heavier

    compare :weight, :threshold => :perfect
    # =>  User A and B have to have the same weight
  end
```

Block evaluation If a block is given, the compared attributes are passed and the result of the block is the final result for the comparison (with `true` being interpreted as 1.0).

```
class UserComparer < Matching::Compare::Base
  compare :email do |a, b|
    email_domain = /[^@]+$/
    a[email_domain] == b[email_domain]
  end
end
```

new(a, b)

Create a new Compare object to compare the given objects.

Public Instance methods

result*()*

Compares two objects and returns a result between 0.0 (not alike) and 1.0 (perfect match).

Examples:

```
Comparer::Base.new(true, false) # => 0.0
Comparer::Base.new(true, true) # => 1.0
```

Protected Instance methods

calc_result*(hsh)*

compare_attribute*(attr, opts = {})*

compare_attributes_and_calc_result*()*

compared_attributes*()*

comparer_for*(klass)*

in_threshold*(x, y, result, threshold = {})*

Floats are interpreted as relative, Fixnums as absolute thresholds.

Class: MatchingRecording

MatchingRecording objects are used to save the results of matching operations.

Public Class methods

MatchingRecording.update!

Recalculates and updates the matching results for all given Freight objects (defaults to all).

Class: Object

Public Instance methods

obj.full?
obj.full? { |f| ... }

Returns wheter or not the given obj is not blank?. If a block is given and the obj is full?, the obj is yielded to that block.

```
salary = nil
salary.full? # => nil
salary.full? { |s| "#{s} $" } # => nil
salary = 100
salary.full? { |s| "#{s} $" } # => "100 $"
```

With Rails' implementation of Symbol#to_proc it is possible to write:

```
current_user.full?(&:name) # => "Dave"
```

Class: Person

Person objects contain personal information about a User.

```
opts = {
  :gender => 'male',
  :first_name => 'Maximilian',
  :last_name => 'Sprenkler',
  :job_description => 'Chief Executive Officer',
  :phone => '+49 (0) 234 569986-1',
  :fax => '+49 (0) 234 569986-55',
  :website => 'example.org/team/m.sprenkler',
}
user.person.create(opts)
```

Public Instance methods

person.name # => **String**

Returns the full name of the person.

Class: RemoteController

Public Class methods

remote_enabled*(opts = {})*

Use this in a controller to allow access to certain actions (defaults to all actions) via xml and json.

```
class StationsController < InheritedResources::Base
  remote_enabled :only => :show
end
```

To restrict API access to actual users, simply use `login_required`. This way, requests have to provide an API key in the params. Of course you can also use `role_required` and the like for finer access management.

```
class PeopleController < InheritedResources::Base
  remote_enabled
  login_required
  role_or_ownership_required :company_admin, :only => [:edit, :update]
end
```

Public Instance methods

show*()*

The standard show action.

Sets a default page title.

Class: Review

Review objects contain a text review of a company.

They have to be approved by an user of that company to appear on their company‘s profile.

Public Instance methods

approved?*()*

Returns `true`, if the review has been approved by a user.

name*()*

Returns the name of the company of this review.

Module: Search

search.rb

The Search module acts as a wrapper to whatever search engine is running in the background.

Public Class methods

Search.clear_index_for(record)

Removes the search index for the given record.

```
user = User.create(:name # => 'Bob')
  # => #<User id: 1, name: "Bob">
Search.find 'bob'
  # => [#<User id: 1, name: "Bob">]

Search.clear_index_for(user)
Search.find 'bob'
  # => []
```

Search.count(query) # => **int**
Search.count(query, models) # => **int**

Returns the total number of results.

```
Search.count "some query"
Search.count "Berlin", [User, Company])
```

Search.find(query) # => **array**
Search.find(query, models) # => **array**
Search / query # => **array**

Returns the matching records from the database.

```
Search.find "some query"
Search.find "Berlin", [User, Company])
Search / "some other query"
```

Search.update_index_for(model_or_record)

Adds a record or a model to the search index.

```
Search << User.first  # update the index for a specific user
Search << User        # update the index of all users
```

Class: SearchRecording

SearchRecording objects are logs of the search queries performed by users.

Public Instance methods

recording.clicks # => int

Returns the number of clicked search results for this search.

Class: SiteInfo

SiteInfo objects contain information about loading and unloading sites, such as name of the site, address of the site, name of the contractor etc.

```
opts = {
  :name => 'Rotterdam Centraal',
  :address => 'Centraal Station',
  :zip => '3013',
  :city => 'Rotterdam',
  :country => 'Nederlands'
}
freight.origin_site_info.create(opts)
```

Class: User

User objects respresent a user of the app and are used to authenticate users upon login (using acts_as_authentic plugin) and handle permission handling via assigned UserRole objects.

Data concerning the actual, human user (like company, gender, language etc.) is stored in associated Person and Company objects.

Public Instance methods

user.has_role?(role_name) # => **boolean**

Returns true if a user has a UserRole with the given `name`.

```
user.has_role?(:administrator) # => true
```

is?*(name)*

Alias for has_role?

postings*()*

recent_site_infos*(origin_or_destination = nil)*

user.roles # => **array**

Returns an array of role names.

```
user.roles # => ["administrator", "company_admin"]
```

user.search_type # => **string**

Returns the type of postings the user is searching for, i.e. freights if the user is posting loading space and vice versa.

```
user.posting_type # => "Freight"
user.search_type # => "LoadingSpace"
```

Class: UserRole

UserRoles grant a logged in User access to certain parts of the application.

Creation

UserRoles are created and identified via their `:name` attribute.

```
UserRole.create(:name => 'employee_of_the_month')
```

Find by name

UserRoles can be found via their `:name` attribute using the `[]` accessor.

```
UserRole[:employee_of_the_month]
```

Assigning

Finally, UserRoles can be eassigned to a User with the « operator.

```
user.user_roles << UserRole[:employee_of_the_month]
```

To access the backend e.g. a user must have administrator priviligues:

```
user.user_roles << UserRole[:administrator]
```

This is also used in the frontend to restrict the priviligues of users in companies.

```
user.user_roles << UserRole[:company_admin]
```

Public Class methods

UserRole.frontend_roles # => Array

Returns an array with all UserRoles that can be assigned via the frontend.

```
UserRole.frontend_roles.map(&:name)
  # => ['company_admin', 'employee', 'employee_of_the_month']
```

Public Instance methods

user_role.user_count # => int

Returns the number of users who have this UserRole.

```
UserRole[:company_admin].user_count
  # => 521
```

user_role.user_percentage # => Float

Returns the percentage of users who have this UserRole.

```
UserRole[:company_admin].user_percentage
  # => 0.763
```

Class: UserSession

UserSession objects are used to handle session management using AuthLogic.

Public Class methods

UserSession.login(user) # => boolean

Authenticates a user and logs him in.

```
UserSession.login(User.first) # => true
```

Autoren:

Dipl.-Kfm. René Föhring

E-Mail: rene.foehring@googlemail.de

Dipl.-Kff. Adina Silvia Bruns

E-Mail: adina.bruns@pim.uni-due.de

Impressum:

Institut für Produktion und
Industrielles Informationsmanagement

Universität Duisburg-Essen, Campus Essen

Fakultät für Wirtschaftswissenschaften

Universitätsstraße 9, 45141 Essen

Website (Institut PIM): www.pim.wiwi.uni-due.de

Website (CODE24): www.code-24.eu

ISSN: 1866-9255

This project has received European Regional Development Funding through INTERREG IV B.

INTERREG IVB

Das Drittmittelprojekt CODE24 – Corridor 24 Development Rotterdam-Genoa wird mit Mitteln der Europäischen Union innerhalb des Rahmenkonzepts „Strategic Initiatives Framework“ des Programms INTERREG IVB NWE gefördert.

Die Projektpartner danken für die großzügige Unterstützung ihrer Forschungs- und Transferarbeiten.

Kooperationspartner:

Universität Duisburg-Essen – Campus Essen
Institut für Produktion und Industrielles Informationsmanagement

Projektberichte des INTERREG-IVB-NWE-Projekts CODE24

ISSN 1866-9255

Nr. 1 Bruns, A.S.; Zelewski, S.; Klumpp, M.: Trends in der Güterverkehrslogistik – eine Expertenbefragung unter besonderer Berücksichtigung von Schienengüterverkehren. Institut für Produktion und Industrielles Informationsmanagement, Universität Duisburg-Essen, Campus Essen. Essen 2010.

Nr. 2 Klippert, S.R.; Kowalski, M.; Bruns, A.S.: Anforderungsanalyse für eine Online-Frachtenbörse im Eisenbahngüterverkehr – Entwicklung einer Anforderungsspezifikation aus betriebswirtschaftlicher Perspektive. Institut für Produktion und Industrielles Informationsmanagement, Universität Duisburg-Essen, Campus Essen. Essen 2010.

Nr. 3 Föhring, R.; Bruns, A.S.: Implementierung des Softwareprototyps einer Online-Frachtenbörse. Institut für Produktion und Industrielles Informationsmanagement, Universität Duisburg-Essen, Campus Essen. Essen 2011.