Hochschule für Angewandte Wissenschaften Hamburg
Hamburg University of Applied Sciences

Bachelorarbeit

Martin Sadowski

Konzeption und prototypische Entwicklung eines Open-Source Vertriebssystems

Fakultät Technik und Informatik
Department Informatik

Faculty of Engineering and Computer Science
Department of Computer Science

Martin Sadowski

Konzeption und prototypische Entwicklung eines Open-Source Vertriebssystems

Bachelorarbeit eingereicht im Rahmen der Bachelorprüfung
im Studiengang Angewandte Informatik
am Department Informatik
der Fakultät Technik und Informatik
der Hochschule für Angewandte Wissenschaften Hamburg

Betreuender Prüfer: Prof. Dr. rer. nat. Jörg Raasch
Zweitgutachter: Prof. Dr. Olaf Zukunft

Kooperation mit Firma: novomind AG

Abgegeben am 07. Juli 2008

Martin Sadowski

Thema der Bachelorarbeit

Konzeption und prototypische Entwicklung eines Open-Source Vertriebssystems

Stichworte

CRM, Vertriebssystem, Konzeption, Prototyp, Open-Source, Java EE, EJB, JSF

Kurzzusammenfassung

Die Steigerung der Kundenzufriedenheit nimmt für ein Unternehmen einen immer höheren Stellenwert ein. Ein Vertriebssystem kann hierbei helfend unter die Arme greifen und somit auch zur Steigerung des Unternehmenserfolgs beitragen. Das Ziel dieser Arbeit ist die Neuentwicklung eines webbasierten Open-Source Vertriebssystems, das bei der novomind AG das bisher genutzte proprietäre Vertriebssystem ablösen soll. Dazu werden in einer Analysephase Gründe für eine Neuentwicklung zusammengetragen und eine Marktübersicht bereits existierender Vertriebssysteme erstellt. Mit einem Prototypen soll abschließend der Grundstein für ein marktreifes Vertriebssystem gelegt werden.

Martin Sadowski

Title of the paper

Design and prototypical development of an open-source distribution system

Keywords

CRM, distribution system, design, prototype, Open-Source, Java EE, EJB, JSF

Abstract

To enhance the customer satisfaction takes on an increasingly significant role for a company. A distribution system can help to contribute to the company's success. The goal of this study is the development of a new web-based open source distribution system for the novomind AG that will replace the so far used proprietary one. For that purpose, reasons for a new development are collected and a market overview of already existing open source distribution systems is created. A prototype which should lay the foundation for an market-ready product is developed in conclusion.

Danksagung

Ich möchte mich an dieser Stelle bei allen bedanken, die mich bei der Erstellung dieser Arbeit unterstützt haben.

Prof. Dr. rer. nat. Jörg Raasch danke ich für die hervorragende Betreuung dieser Arbeit. Ebenso danke ich Prof. Dr. Olaf Zukunft, der sich bereit erklärt hat, die Zweitkorrektur dieser Arbeit zu übernehmen.

Besonders bedanken möchte ich mich bei Herrn Peter Samuelsen für die Ermöglichung dieser Arbeit und für die fachliche Betreuung des Projektes, sowie bei allen Mitarbeitern der novomind AG für die Unterstützung während der Bearbeitung dieser Arbeit.

Ein weiterer Dank gilt André Schmer und Fahim Aleaf für die gute Zusammenarbeit in diesem Projekt.

Des Weiteren danke ich meinen Eltern, die mich während des gesamten Studiums unterstützt haben, und mir eine hervorragende Ausbildung ermöglicht haben.

Ein ganz besonderer Dank gilt meiner Freundin, die mich während der Erstellung dieser Arbeit ertragen musste, und mir immer zur Seite stand.

Martin Sadowski, Juli 2008

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

Quelltextverzeichnis

1 Einleitung

1.1 Motivation

Die Verwaltung von Kundendaten in einem Vertriebssystem ist für ein Unternehmen heutzutage unerlässlich. Nur durch eine ganzheitliche Sicht auf seine Kunden, kann sichergestellt werden, dass die Kundenkommunikation problemlos und effizient verläuft. Ein Vertriebssystem kann dabei jedoch immer nur Teil einer kundenorientierten Unternehmensstrategie sein, indem es die Speicherung, Verarbeitung und Nutzung von Kundendaten unterstützt.

Ein langjährig eingesetztes Vertriebssystem kann durch den immer schneller werdenden technologischen Wandel, bei fehlender Weiterentwicklung, schnell veralten und sein Einsatz kann sich immer problematischer gestalten. So kann es, wie im Fall der novomind AG, möglicherweise unausweichlich sein, ein seit Jahren etabliertes System durch ein Neues zu ersetzen. Vor allem bei der Nutzung eines proprietären Systems ist man als einsetzendes Unternehmen abhängig von der Weiterentwicklung durch den Hersteller. Es kann also, zur Beseitigung dieser Abhängigkeit, Sinn ergeben, eine Eigenentwicklung oder ein Open Source System einzusetzen. Immer mehr Unternehmen verbinden diese beiden Möglichkeiten und veröffentlichen ihre Eigententwicklungen als Open Source Software. Dies soll auch in diesem Projekt erfolgen, indem die Erfahrungen, die bei der Ablösung des bisherigen Vertriebssystems gesammelt werden, sei es durch den Einsatz und die Weiterentwicklung eines bereits existierenden Systems, oder eine Eigenentwicklung, in die Open Source Gemeinde fließen sollen.

1.2 Aufgabe und Ziel

Ziel dieser Arbeit ist es, den Weg für die Ablösung des bisherigen Vertriebssystems der novomind AG zu ebnen. Das Altsystem ist in die Jahre gekommen und soll durch einen würdigen Nachfolger abgelöst werden.

Diese Arbeit ist die erste von drei bereits geplanten Arbeiten für diese Aufgabenstellung und beschäftigt sich mit der Ausarbeitung eines Konzepts für ein neues Vertiebssystem. Dabei soll, durch geeignete Analysen, eine Vision des System erarbeitet werden und in einer

Marktübersicht mögliche einsetzbare Systeme gefunden werden. (An dieser Stelle soll vorgegriffen werden, dass keines der untersuchten Systeme die Anforderungen zu 100% erfüllt und daher eine Neuentwicklung angestrebt wird.) Zum Abschluss dieser Arbeit soll ein explorativer Prototyp erstellt werden, der die technische Realisierbarkeit des Konzepts beweist und als Grundlage für die weitere Entwicklung dient.

Aufbauend auf dieser Arbeit beschäftigt sich André Schmer (Schmer, 2008) mit der Entwicklung eines evolutionären Prototypens bis zum marktreifen Produkt. Dabei greift er das in dieser Arbeit zu entwickelnde Konzept eines Vertriebssystems auf und führt diese Arbeit fort. Fahim Aleaf erarbeitet in seiner Arbeit (Aleaf, 2008) den Teilbereich des Reportings und der Selektion.

1.3 Umfeld

Diese Arbeit entstand im Rahmen einer Tätigkeit bei der novomind AG und wurde in den Räumlichkeiten des Unternehmens bearbeitet.

Die novomind AG ist ein, 1999 gegründeter, Softwareanbieter und hat seinen Schwerpunkt bei Softwarelösungen für die Kundenkommunikation im Internet. Das Unternehmen beschäftigt heute mehr als 60 Mitarbeiter.
Die novomind AG bietet seinen Kunden zum einen die Self Service Suite, bestehend aus den Produkten novomind iMail™, ein professionelles E-Mail Management System, novomind IQ™, eine Lösung für virtuelle Kundenberater und novomind TrueTALK™, ein System zur interaktiven Echtzeit-Kommunikation im Internet.
Darüber hinaus realisiert die novomind AG im Kundenauftrag komplexe E-Business-Applikationen, wie z. B. skalierbare Online-Shop Lösungen.

Zu der Firmengruppe der novomind AG gehört zudem das Tochterunternehmen BTIM (Beraterteam für Informationsmanagement). Die BTIM Unternehmensberatung GmbH berät Kunden z. B. in den Bereichen Outsourcing oder der Einführung neuer Architekturen und Systeme.
Mit der Einführung eines neuen Vertriebssystems möchte die novomind AG zudem die Nutzung zweier getrennter Vertriebssysteme beenden, ein gemeinsames System schaffen und so den Austausch von Informationen vereinfachen.

Das in dieser Arbeit entwickelte, und darüber hinaus weiterentwickelte, Vertriebssystem möchte die novomind AG, sobald es einen marktreifen Status erlangt hat, als Open Source Software zur Verfügung stellen. Durch diese Veröffentlichung erhofft sich die novomind AG, Erfahrungen mit Open Source Software, aus der Sicht eines Anbieters, zu sammeln. Open Source Software wird heutzutage von nahezu jedem eingesetzt, aber nur wenige geben etwas in die Open Source Gemeinde zurück. Im idealen Fall wird das System von einer

Entwicklergemeinde gut aufgenommen und evt. weiterentwickelt. Solch eine Weiterentwicklung würde wiederum der novomind AG zu Gute kommen.

1.4 Gliederung der Arbeit

Kapitel 2 *„Grundlagen“* befasst sich mit den theoretischen Grundlagen, auf die diese Arbeit aufbaut. So werden in dem ersten Teil Konzepte des Customer Relationship Management erläutert. Anschließend wird auf die Grundlagen der Umsetzung des Prototypen, der mit Hilfe der Java EE Plattform entwickelt wurde, eingegangen.

In Kapitel 3 *„Vision“* wird eine Vision eines Vertriebssystems für die novomind AG entwickelt. Dazu werden durch Analysen, die Anforderungen an das neue Vertriebssystem ermittelt und in einer Marktübersicht mit existierenden Lösungen verglichen.

Kapitel 4 *„Architektur“* befasst sich mit dem Entwurf einer Architektur für das zu entwickelnde Vertriebssystem. Es werden in fachlicher, aber auch technischer, Hinsicht die Komponenten des Systems beschrieben. Außerdem wird das Konzept für die Benutzungsoberfläche beschrieben.

In Kapitel 5 *„Prototyp“* wird der entwickelte Prototyp vorgestellt. Dazu wird die Ablaufumgebung beschrieben, die bei der Umsetzung verwendet wurde. Zudem wird auf aufgetretene Probleme und mögliche Maßnahmen zu deren Lösung eingegangen.

Das Kapitel 6 beschreibt durch eine *„methodische Abstraktion“* die Abgrenzung und das Einsatzpotential des entwickelten Konzeptes eines Vertriebssystems.

Abschließend wird im Kapitel 7 *„Zusammenfassung“* ein kurzes Fazit der Arbeit gezogen und ein Ausblick auf eine Weiterentwicklung gewagt.

2 Grundlagen

In diesem Kapitel werden die grundlegenden Themen, auf die diese Arbeit aufbaut, erläutert. Zunächst wird auf das Customer Relationship Management eingegangen um den theoretischen Aspekt eines Vertriebssystems abzudecken. In einem technischeren Teil werden Java EE Konzepte und EJB, mit Hilfe derer die prototypische Umsetzung realisiert wurde, erklärt.

2.1 Customer Relationship Management

„CRM ist eine kundenorientierte Unternehmensstrategie, die mit Hilfe moderner Informations- und Kommunikationstechnologien versucht, auf lange Sicht profitable Kundenbeziehungen durch ganzheitliche und individuelle Merketing-, Vertriebs- und Servicekonzepte aufzubauen und zu festigen“ (Hippner und Wilde, 2003)

Diese Definition des CRM Begriffes beschreibt in treffender Weise die Reichweite von CRM als komplette Unternehmensstrategie und zeigt dabei die Diskrepanz des Begriffes in der heutigen, von Informationstechnologie beherrschten Zeit, auf. Dem Begriff CRM wird dabei ein hoher technologischer Anteil beigemessen (vgl. Brendel, 2003). Die Technik spielt im Gesamtkontext Customer Relationship Management jedoch keineswegs die dominierende Rolle. Vielmehr ist die komplette Ausrichtung des Unternehmens auf ein prozessoptimiertes Beziehungsmanagement wichtig.

Die Ziele des CRM-Ansatzes sind dabei die folgenden (vgl. Dangelmaier u. a., 2004):

- Höhere Qualität der Kundenbearbeitung
- Verbesserung der internen Bearbeitungsprozesse
- Verbessertes Kundendatenmanagement
- Verbesserung der Schnittstellen zum Kunden

Im Folgenden soll nun die technologische Komponente des CRM, das CRM-System bzw. das Vertriebssystem, näher betrachtet werden. Da das Ziel dieser Arbeit die Ablösung eines existierenden Systems ist, die Konzepte zum Beziehungsmanagement bei der novomind AG bereits etabliert sind und keiner grundlegenden Änderung bedürfen, soll für eine Vertiefung in die allgemeine Thematik des CRM auf die zahlreiche Literatur (z. B. Hippner und Wilde, 2006; Uebel u. a., 2004; Gawlik u. a., 2002) verwiesen werden.

2.1.1 Komponenten eines CRM-Systems

Es lassen sich folgende zentrale Aufgaben eines CRM-Systems identifizieren (vgl. Hippner u. a., 2006b):

- die Zusammenführung und Analyse aller Kundeninformationen
- die Unterstützung und Synchronisation der zentralen Customer Touch Points (Marketing, Vertrieb, Service)
- Steuerung und Integration aller Kommunikationskanäle zwischen Kunde und Unternehmen

Im Folgenden sollen die drei Komponenten, in die sich ein CRM-System, diesen Aufgaben entsprechend, unterteilen lässt, erläutert werden.

Analytisches CRM

In dem analytischen CRM werden alle Kundenkontakte und -reaktionen aufgezeichnet und durch verschiedene Analysen ausgewertet. Dabei kommt es darauf an, möglichst viel Wissen aus den gespeicherten Kundendaten zu gewinnen. Das Ziel des analytischen CRM ist es, Kundenreaktionen zu verwerten und die Kommunikation durch Vorhersagen an die Kundenbedürfnisse auszurichten.

Operatives CRM

Das operative CRM umfasst alle dem Kunden zugewandten Anwendungen, wie Kontaktmanagement, Vertriebsautomatisierung, Auftragsmanagement, Kampagnenmanagement und Kundenservice und soll helfen, die Geschäftsprozesse zu optimieren. Dabei ist es wichtig, dass alle Kontaktformen (Customer Touch Points) eingebunden werden. Die Basis des operativen CRM bildet eine Kundendatenbank. Ergänzend können vorhandene „Back Office“-Lösungen, wie z. B. ein Enterprise Resource Planning (ERP) System, angebunden werden.

Darüber hinaus kann das operative CRM zudem einen kollaborativen Bereich beinhalten, bei dem eine Interaktion des Kunden bzw. anderer beteiligter Akteure, wie z. B. Lieferanten, Hersteller, usw., mit der CRM-Software stattfindet (vgl. Kracklauer u. a., 2004).

Kommunikatives CRM

Unter die Aufgaben eines kommunikativen CRM fällt die Steuerung aller Kommunikationskanäle zum Kunden. Die bedeutungsvollsten Kanäle sind: Telefon, Internet, E-Mail und Brief. Man spricht auch vom Multichannel-Management (vgl. Gawlik u. a., 2002). Es soll dabei die effiziente Nutzung der verschiedenen Kommunikationskanäle sichergestellt werden.

2.1.2 Klassifizierung von CRM-Systemen

CRM-Systeme lassen sich generell in drei Gruppen klassifizieren (vgl. Hippner u. a., 2006a).

Integrierte Globallösungen sind für nahezu alle CRM-Funktionalitäten ausgelegt. Diese Systeme sind zum Teil um CRM-Funktionen erweiterte ERP-Systeme. Sie benötigen in der Regel einen hohen Aufwand, um sie an unternehmensspezifische Anforderungen anzupassen.

Daneben existieren CRM-Systeme, welche sich auf *funktionale Teillösungen*, d. h. auf eine Komponente (siehe 2.1.1) konzentrieren. So gibt es spezialisierte Lösungen für die Bereiche des operativen oder des analytischen CRM. Für diese Systeme ist es wichtig, Schnittstellen anzubieten, die notwendig sind, um mit anderen Teillösungen kommunizieren zu können.

Außerdem werden mittlerweile viele *branchenspezifische Standardlösungen* angeboten. Hierbei werden, je nach Branche, einzelne Funktionalitäten eingeschränkt oder aber verstärkt benötigt.

Darüber hinaus kann man CRM-Systeme, die für den Bussines-to-Bussines (B2B) oder den Business-to-Consumer (B2C) Bereich ausgelegt sind, unterscheiden. Bei der direkten Kommunikation mit Endkunden (B2C) fallen meist sehr große Datenmengen an und es werden leistungsfähige Analysesysteme (Analytisches CRM) eingesetzt. Im Gegensatz dazu fallen im B2B Markt nur sehr geringe Datenmengen an und die technologischen Anforderungen sind meist weniger komplex. Bei der B2B-Kommunikation spielt vor allem das operative CRM eine große Rolle.

2.2 Java EE

Als Java EE, oder Java Platform, Enterprise Edition, bezeichnet man die Spezifikation einer Softwarearchitektur von in Java programmierten, mehrschichtigen und verteilten, Anwendungen. Die Spezifikation bildet hierbei einen Rahmen für standardisierte und modulare Komponenten (vgl. Sun Microsystems Inc., 2008a). Die aktuelle Version der Spezifikation und der Referenzimplementierung ist die Version 5 vom Mai 2006. Im Folgenden sind alle Informationen auf diese Version bezogen. Java EE soll bei der Entwicklung von hochverfügbaren, transaktionssicheren, robusten und portierbaren Enterprise Anwendungen helfen. Diese Anwendungen benötigen eine spezielle Laufzeitumgebung, den sogenannten Java EE Application Server. Der Application Server stellt hierbei die technische Funktionalität für Sicherheit, Transaktionen, Namens- und Verzeichnisdienste, Kommunikation zwischen Java EE Komponenten u.v.m. zur Verfügung. (Wikipedia, 2008b)

Die Abbildung 2.1 zeigt die Referenzarchitektur einer Java EE Applikation. Als Basis dient das Enterprise Information System, was in einfachen Anwendungen nur aus einer Datenbank besteht, in komplexeren Anwendungen aber auch z. B. Altsysteme beinhalten kann. Darauf aufbauend befindet sich die Server-Side Business Logic, die die Geschäftslogik einer Anwendung wiederspiegelt. Die Präsentation ist unterteilt in Server-Side Presentation und Client-Side Presentation.

Die wichtigsten Komponenten der Java EE Spezifikation sind:

- Enterprise Java Beans (EJB) definieren die Geschäftslogik einer Anwendung und ermöglichen Zugriff auf persistente Daten
- Java Servlet API ermöglicht dynamische Webanwendungen
- Java Message Service (JMS), eine API für asynchronen Nachrichtenaustausch
- Java Authentication and Authorization Service (JAAS) ermöglicht eine Authentifikation und das Bereitstellen von Zugriffsrechten in Java Programmen
- JavaMail macht den Zugriff auf Mail-Dienste, wie z. B. SMTP, POP3 oder IMAP möglich
- Java Server Faces (JSF) ermöglicht dem Benutzer Komponenten für Benutzerschnittstellen in Webseiten auf einfache Art und Weise zu erstellen

Im Folgenden soll nun, die für die Entwicklung des Prototypen verwendete Technologie EJB näher erläutert werden.

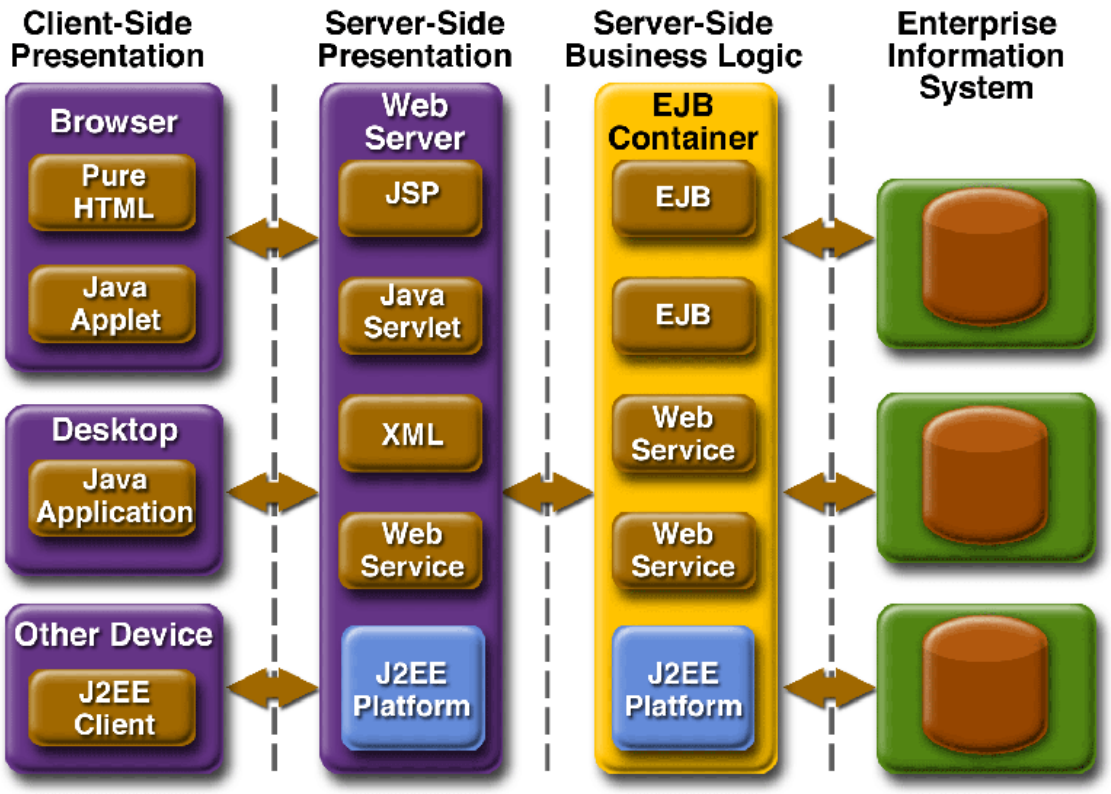

Abbildung 2.1: Java EE Application Model

2.2.1 EJB

Die Abkürzung EJB steht für Enterprise Java Beans und bezeichnet eine Komponente der Java EE Spezifikation. EJB hilft bei der Entwicklung der Business Logik einer Java EE Applikation. So soll der Code bei der Verwendung von Enterprise Java Beans einen höheren Wiederverwendungsgrad haben.

Bei Enterprise Java Beans unterscheidet man hauptsächlich 3 Arten von Beans (vgl. Ihns u. a., 2007):

- Entitiy Beans
- Session Beans
- Message Driven Beans

Entity Beans

Entity Beans, oder auch EJB 3.0 Entities genannt, bilden die persistenten Daten einer Java EE Anwendung. Sie sind verteilte Objekte, die als POJOs (Plain old Java Objects) im System realisiert sind. Über Java Annotations, d. h. im Quelltext enthaltene Metadaten, wird die Persistenz der Objekte gesteuert. Der Entity Manager übernimmt das objekrelationale Mapping zwischen der Datenbank und den Entity Beans.

Das Listing 2.1 zeigt ein Beispiel für eine Entity Bean. Über die Annotation in Zeile 1 wird festgelegt, dass die folgende Java Klasse ein Entity Bean ist. In Zeile 2 wird festgelegt, dass die Klasse in der Tabelle „cars“ persistent sein soll.
Für die Variable „id“ wird über die Annotation in Zeile 6 geregelt, dass die Datenbank für die Generierung der Werte zuständig ist. Der Entity Manager kümmert sich automatisch um den richtigen Mechanismus für die verwendete Datenbank. So wird z. B. bei einer MySQL Datenbank die Feldeigenschaft „AUTO INCREMENT“ verwendet, bei einer Oracle Datenbank hingegen eine Sequenz.
Über die Annotations in den Zeilen 10 und 11 wird festgelegt, dass die Variable „manufacturer“ über eine n-zu-eins Beziehung gespeichert werden soll. Die Feldbezeichner in Zeile 11 geben an, über welche Felder der Fremdschlüssel aufgebaut werden soll. Die Tabelle, welche referenziert werden soll, wird in der Definition der Klasse „Manufacturer“ angegeben. Des Weiteren sind für jede Variable die Getter- und Setter-Methoden definiert.

Eine solche Entity Bean könnte mit Hilfe des Entity Managers in einer Datenbank persistent gehalten werden und innerhalb der Java EE Anwendung überall verwendet werden.

```java
@Entity
@Table(name = "cars")
public class Car implements Serializable {

  @Id
  @GeneratedValue
  @Column(name="id")
  private int _id;

  @ManyToOne
  @JoinColumn(name="manufacturer", referencedColumnName="id")
  private Manufacturer _manufacturer;

  @Column(name = "model")
  private String _model;

  public int getId() {
    return _id;
  }

  public void setId(int id) {
    this._id = id;
  }

  public Manufacturer getManufacturer() {
    return _manufacturer;
  }

  public void setManufacturer(Manufacturer manufacturer) {
    this._manufacturer = manufacturer;
  }

  public String getModel() {
    return _model;
  }

  public void setModel(String model) {
    this._model = model;
  }

}
```

Listing 2.1: Beispiel einer Entity Bean

Session Beans

Mittels Session Beans werden i.d.R. Zugriffe auf Entity Beans gesteuert. (vgl. Märsch, 2006) Eine Session Bean muss ein Interface implementieren, das festlegt, welche Methoden publiziert werden. Der Aufrufende referenziert nur dieses Interface, kennt die Klasse, welche das Interface implementiert daher nicht.
Es wird bei Session Beans zwischen zustandslosen (stateless) und zustandsbehafteten (stateful) Session Beans unterschieden.

Stateful Session Beans speichern ihren Zustand über einen Methodenaufruf hinaus. Dadurch kann auf die Inhalte ihrer Variablen innerhalb einer Session zugegriffen werden.

Ein Stateless Session Bean speichert im Gegensatz zu einem Stateful Session Bean seinen Zustand nicht, d. h. die Inhalte seiner Variablen sind nicht persistent. Der Vorteil dieser Art einer Session Bean ist der geringere Overhead und die damit verbundene geringere Ressourcennutzung. Sie können zudem gefahrlos parallel aufgerufen werden. Wann immer die Nutzung einer Stateless Session Bean möglich ist, sollte sie einer Stateful Session Bean vorgezogen werden.

```
1  public interface CarServiceLocal {
2
3    public Car getCar(int id);
4    public List<Car> getAllCars();
5    public void addCar(Car c);
6    public void deleteCar(Car c);
7
8  }
```

Listing 2.2: Beispiel eines Interface für eine Session Bean

Das Listing 2.2 zeigt ein Beispiel für ein Interface, das eine Session Bean implementieren könnte. Es beinhaltet die Definition für Methoden, die entfernt aufrufbar sein sollen.

```
1  @Stateless
2  @Local
3  public class CarServiceBean implements CarServiceLocal {
4
5    @PersistenceContext(unitName="carDS")
6    protected EntityManager em;
7
8    public Car getCar(int id) {
9      Car c = em.find(Car.class, Integer.valueOf(id));
10     return c;
11   }
12
13   public List<Car> getAllCars() {
14     Query q = em.createQuery("SELECT c FROM CAR c");
15     return q.getResultList();
16   }
17
18   public void addCar(Car c) {
19     em.persist(c);
20   }
21
22   public void deleteCar(Car c) {
23     Car _car = em.merge(c);
24     em.remove(_car);
25   }
26
27 }
```

Listing 2.3: Beispiel einer Stateless Session Bean

Das Listing 2.3 enthält die Implementierung des Interfaces und die Definition einer Stateless Session Bean. Die Annotation in Zeile 1 deklariert die Session Bean als stateless. Mit der Annotation „Local“ in Zeile 2 wird festgelegt, dass die Bean nur innerhalb des EJB-Containers aufgerufen werden kann und nicht von Applikationen außerhalb des Containers verwendet werden kann.
Die Annotation in Zeile 5 steuert die Dependency Injection des Entity Managers. Der Variablen wird zur Laufzeit durch den EJB-Container eine gültige Instanz des Entity Managers injiziert.
Die Zeilen 9, 14, 19, 23 und 24 zeigen die Verwendung des Entity Managers zur Speicherung und Beschaffung von Objekten.

Message Driven Beans

Message Driven Beans (MDB) werden in EJB-Systemen für eine asynchrone Kommunikation genutzt. Dabei wird der Java Message Service (JMS) (siehe Sun Microsystems Inc., 2008b) genutzt. Eine MDB verknüpft sich mit einer JMS Nachrichten Warteschlange und verarbeitet eingehende Nachrichten. Diese Verarbeitung findet asynchron statt, d. h. der Aufrufende, bzw. der Sender der Nachricht, wartet nicht auf eine Antwort. Im Gegensatz zu den beiden anderen Arten von Beans kann eine MDB nicht direkt aufgerufen werden, sie ist für einen Aufrufenden nicht sichtbar.

```
1  @MessageDriven(activationConfig= {
2    @ActivationConfigProperty(propertyName="destinationType",
3       propertyValue="javax.jms.Queue"),
4    @ActivationConfigProperty(propertyName="destination",
5       propertyValue="queue/Cars"),
6    @ActivationConfigProperty(propertyName="maxSession",
7       propertyValue="1") })
8  public class LogMessageBean implements MessageListener {
9
10   public void onMessage(final Message message) {
11     \\do asynchron logging here.
12   }
13 }
```

Listing 2.4: Beispiel einer Message Driven Bean

Das Listing 2.4 zeigt den Aufbau einer Message Driven Bean. Über die Annotations in den Zeilen 1-7 wird die Art der MDB, der Name der Queue und die Anzahl der parallel zu verarbeitenden Nachrichten festgelegt. Wenn in der Queue eine Nachricht eintrifft, wird die Methode „onMessage“ der MDB aufgerufen. Als Parameter wird die Nachricht übergeben.

```
1  Car c = new Car();
2
3  InitialContext ic = new InitialContext();
4  final Queue queue = (Queue) ic.lookup("queue/Cars");
5  final QueueConnectionFactory factory = (QueueConnectionFactory) ic.lookup("java:/
       ConnectionFactory");
6  final QueueConnection connection = factory.createQueueConnection();
7  final QueueSession session = connection.createQueueSession(false, QueueSession.
       AUTO_ACKNOWLEDGE);
8
9  ObjectMessage message = session.createObjectMessage();
10 message.setObject(c);
11
12 final QueueSender sender = session.createSender(queue);
13 sender.send(message);
14
15 session.close();
16 connection.close();
```

Listing 2.5: Beispiel für die Nutzung einer MDB

Im Listing 2.5 wird beispielhaft die Nutzung einer Message Driven Bean gezeigt. So wird zunächst die Referenz zur QueueSession benötigt. Über diese Session kann eine Nachricht aufgebaut, und an die Warteschlange gesendet werden.

3 Vision

In diesem Kapitel geht es um die Entwicklung einer Vision für das neue Vertriebssystem. Ziel ist es, durch entsprechende Analysen die Anforderungen an das neue Vertriebssystem zu bestimmen. Dazu werden zunächst die Geschäftsprozesse des Vertriebs identifiziert und das Altsystem bewertet. Dadurch soll eine abstrakte Vision des neuen Vertriebssystem entstehen, die in eine Zusammenfassung der Anforderungen eingeht. Anschließend werden diese Anforderungen in einer Marktübersicht mit bereits verfügbaren CRM-Systemen verglichen, um eine Entscheidung treffen zu können, ob ein existierendes System eingesetzt werden kann oder ob eine Neuentwicklung geplant werden sollte.

3.1 Untersuchung des Altsystems

Zur Ausarbeitung der Vision eines neuen Vertriebssystems sollen zunächst die etablierten Prozesse des Vertriebs der novomind AG verstanden werden. Dabei muss das bisherige Vertriebssystem in seinen genutzten Funktionen verstanden werden.

Das bisherige Vertriebssystem wird hauptsächlich zur Speicherung sämtlicher Kundenkontakte und -aktivitäten genutzt. Es findet dabei keinerlei Verwendung in Form eines kollaborativem oder kommunikativem CRM Systems.

Die wichtigsten genutzten Funktionen des Altsystems sind die Folgenden:

- Verwaltung von Unternehmen
- Verwaltung von Kontaktpersonen aus Unternehmen
- Verwaltung von Aktivitäten
- Verwaltung von Opportunities
- Selektion von Datensätzen und Export in das Comma Separated Value (CSV)-Format

Der Benutzer hat die Möglichkeit, Unternehmen in dem System zu verwalten, d. h. anzulegen, einzusehen, zu aktualisieren und zu löschen. Zu einem Unternehmen können Kontaktpersonen zugeordnet werden. Aktivitäten werden einem Unternehmen zugeordnet und können zusätzlich auch Verweise auf Kontaktpersonen beinhalten. Außerdem haben Aktivitäten einen Beschreibungstext und ein Start- und Enddatum. Auch Opportunities werden einem Unternehmen zugeordnet und u. a. mit einer Abschlusswahrscheinlichkeit und einem erwarteten Enddatum versehen. Zu jeder Entität (Unternehmen, Kontaktpersonen, Aktivitäten und Opportunities) besteht zudem die Möglichkeit, benutzerdefinierte Felder anzulegen und das System so an seine Anforderungen anzupassen.
Des Weiteren wird zu jedem Datensatz gespeichert, welcher Benutzer ihn angelegt bzw. zuletzt bearbeitet hat und wem er zugeordnet ist. So ist z. B. eine Selektion aller offenen Aktivitäten eines bestimmten Vertriebmitarbeiters möglich. Solche Selektionen können als Listen im System angezeigt aber auch im CSV-Format exportiert und dann z. B. mit Microsoft Excel geöffnet werden.

3.2 Bewertung des Altsystems

In Interviews mit Vertriebsmitarbeitern der novomind AG wurde vor allem Unmut über die Bedienbarkeit des Altsystems geäußert. Das System verhalte sich nicht erwartungskonform und weiche z. B. bei Tastenkombinationen von Standards ab.
Darüber hinaus wurde das Datenmodell, das dem Altsystem zu Grunde liegt, kritisiert. So ist es nicht möglich, einem Unternehmen oder einer Kontaktperson mehrere Adressen zuzuordnen. Unternehmen lassen sich nicht mit ihren Niederlassungen, Call Centern oder Zweigstellen verknüpfen. Kontaktpersonen können bei einem Wechsel des Unternehmens nicht mit den kompletten Kontaktdaten verschoben werden.
Des Weiteren wurde auf die mangelhafte Selektionsmöglichkeit von Datensätzen hingewiesen, was einen umständlichen Export nach Microsoft Excel und dort z.T. eine manuelle Selektion erfordert.
Als Wunsch für das neue System wurde eine globale Suche genannt, die innerhalb aller Datensätze nach Stichworten sucht und das Auffinden, speziell von Kontaktpersonen, deutlich erleichtern würde.

3.3 Vision für Neuplanung

Der Analyse des Altsystems nach zu urteilen, ist ein neues Vertriebssystem wünschenswert, das in erster Linie deutliche Verbesserungen bei der Bedienbarkeit liefert und in einem ersten Schritt „nur“ die auch bisher genutzte Funktionalität mit sich bringt. Das neue System

sollte sich also auf die Speicherung der Kundendaten konzentrieren, diese aber so komfortabel wie möglich gestalten. Dabei sollte die neue Lösung aber so beschaffen sein, dass sich jederzeit zusätzliche Funktionalität einbauen lässt, um bisher noch nicht absehbare Anforderungen zu erfüllen. Außerdem sollte es in dem neuen Vertriebssystem möglich sein, die eigene Unternehmensstruktur abzubilden und das Vertriebssystem innerhalb der gesamten Firmengruppe zu nutzen. Dadurch würden bisherige Insellösungen abgeschafft. Dabei muss es allerdings auch möglich sein, Sicherheitseinstellungen, wie z. B. die Sichtbarkeit von Elementen, fein granular einzustellen.

Das neue Vertriebssystem sollte zunächst die Anforderungen der novomind AG erfüllen und dann erst, in Hinblick auf eine Veröffentlichung als Open Source Software, erweitert werden. Diese Veröffentlichung sollte erst stattfinden, wenn das Vertriebssystem bei der novomind AG eingeführt und erste Erfahrungen damit gesammelt wurden. Daher kann die Veröffentlichung des zu entwickelnden Systems in dieser Arbeit nur vorausschauend erläutert werden.

3.4 Anforderungsanalyse

Zusammenfassend sollen in diesem Abschnitt die Anforderungen an das zu entwickelnde Vertriebssystem aufgelistet werden. Die Anforderungen sind aus Gesprächen mit den Vertriebsmitarbeitern und aus der Untersuchung des Altsystems hervorgegangen. Diese Anforderungen werden in funktionale und nichtfunktionale Anforderungen unterteilt. Funktionale Anforderungen legen fest, was das Vertriebssystem tun soll. Nichfunktionale Anforderungen legen fest, welche Eigenschaften das Vertriebssystem haben soll. Die Anforderungen werden in „must“, „should“ und „nice to have“ klassifiziert. Außerdem werden technische Randbedingungen beschrieben, die bei der Umsetzung berücksichtigt werden müssen.

Abbildung 3.1 gibt einen groben Überblick über die Anwendungsfälle und Akteure, die identifiziert wurden.

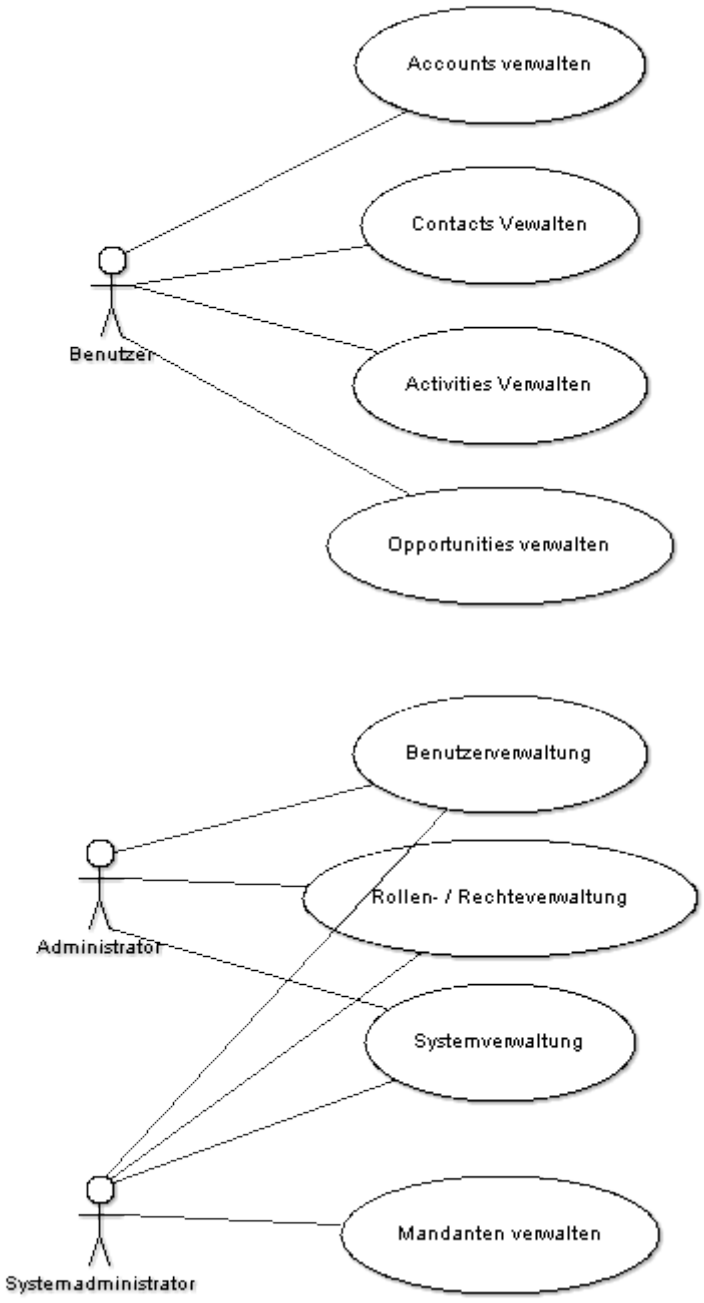

Abbildung 3.1: Anwendungsfalldiagramm

3.4.1 Funktionale Anforderungen

Datenorientierte Anforderungen

1. Verwalten von Accounts (must)

Der Benutzer muss die Möglichkeit haben, Accounts im Vertriebssystem abzuspeichern, wiederzufinden, zu editieren und ggf. zu löschen. Für das Auffinden von Accounts muss dem Benutzer eine Liste mit sämtlichen im System vorhandenen Accounts zur Verfügung stehen. Innerhalb dieser Liste muss er mit Hilfe eines Filters bestimmte Accounts selektieren können.

2. Verwalten von Kontaktpersonen (must)

Der Benutzer muss die Möglichkeit haben, Kontaktpersonen im Vertriebssystem abzuspeichern, wiederzufinden, zu editieren und ggf. zu löschen. Auch hier muss dem Benutzer eine Liste und ein Filter zur Verfügung gestellt werden, um Kontaktpersonen im System wiederzufinden.

3. Verschieben von Kontaktpersonen (should)

Der Benutzer sollte die Möglichkeit haben, Kontaktpersonen von einem Account in einen anderen Account zu verschieben. Dabei müssen die Stammdaten der Kontaktperson erhalten bleiben und später über den neuen Account auffindbar sein.

4. Verwalten von Aktivitäten (must)

Der Benutzer muss die Möglichkeit haben, Aktivitäten im Vertriebssystem abzuspeichern, wiederzufinden, zu editieren und ggf. zu löschen. Außerdem muss der Benutzer die Möglichkeit haben, einer Aktivität einen längeren Beschreibungstext anzufügen. Innerhalb diesen Textes sollte es auch möglich sein, Formatierungen vorzunehmen.

5. Verwalten von Opportunities (must)

Der Benutzer muss die Möglichkeit haben, Opportunities im Vertriebssystem abzuspeichern, wiederzufinden, zu editieren und ggf. zu löschen.

6. Benutzerdefinierte Felder (must)

Der Benutzer muss die Möglichkeit haben, in den vier Entitäten (Accounts, Contacts, Activities, Opportunities) benutzerdefiernierte Felder anzulegen und ggf. zu löschen. Dadurch ist es dem Benutzer erst möglich, das Vertriebssystem an das CRM-Konzept seines Unternehmens anzupassen.

7. Businessprofil (must)

Der Benutzer muss die Möglichkeit haben, das Profil seines Unternehmens im Vertriebssystem abzubilden. Dazu erstellt er eine Unternehmensgruppe, zu der ein oder mehrere Unternehmen gehören, die aus einer oder mehreren Business Units bestehen.

8. Selektion / **Export** (should)

Ein Benutzer des Vertriebssystems sollte die Möglichkeit haben, aus den vier Entitäten (Accounts, Contacts, Activities und Opportunities) nach von ihm zu definierenden Kriterien zu selektieren und diese Selektion in ein geeignetes Format zu exportieren.

Weitere Anforderungen

9. Globale Suche (should)

Der Benutzer sollte die Möglichkeit haben über ein zentrales und auf jeder Seite sichtbares Suchfeld innerhalb aller Datensätze suchen zu können. Auf der Ergebnisseite sollte eine Unterscheidung zwischen den Entitäten (Accounts, Contacts, Activities und Opportunities) stattfinden.

10. Benutzerverwaltung (must)

Einem Administrator muss es möglich sein, Benutzer in das System einzutragen, die Daten eines Benutzers zu editieren oder ihn ggf. zu löschen. Außerdem muss der Administrator die Möglichkeit haben, jedem Benutzer Rollen zuzuweisen.

11. Rechte- / **Rollenverwaltung** (should)

Da das Vertriebssystem Rollenbasiert aufgebaut wird, sollte ein Administrator die Möglichkeit haben, die Rechte, die ein Benutzer des Systems haben kann, sinnvoll auf Rollen zu verteilen.

12. Mandantenfähigkeit (should)

Das Vertriebssystem sollte Mandantenfähig sein, d. h. es sollte möglich sein, auf einer Installation des Systems mehrere Unternehmen arbeiten zu lassen, die sich gegenseitig jedoch nicht beeinflussen und nur Zugriff auf ihre eigenen Daten haben dürfen.

13. Mehrsprachenfähigkeit (nice to have)

Da das neue Vertriebssystem der Open Source Gemeinde zur Verfügung gestellt werden soll, wäre es sinnvoll das System in verschiedenen Sprachen betreiben zu können. Dazu sollte der Benutzer beim Login die Möglichkeit haben, eine Sprache zu wählen. Diese Einstellung der Sprache sollte auch Auswirkungen auf Einstellungen wie Zahlen- oder Datumsformate und die Währung haben.

3.4.2 Nichfunktionale Anforderungen

Aussehen und Handhabung

Da das Vertriebssystem zu einem späteren Zeitpunkt als Open Source Software veröffentlicht werden soll, muss das Aussehen veränderbar sein und von einem einsetzenden Unternehmen an das eigene „Corporate Design“ anpassbar sein. Bei der Auslieferung sollte das Vertriebssystem eine neutral gestaltete Bedienoberfläche besitzen. Es sollte jedoch für den Einsatz bei der novomind AG in seinem „Look and Feel“ dem Corporate Design der novomind AG entsprechen. In Abschnitt 4.3 wird der Entwurf der Benutzeroberfläche beschrieben.

Bedienbarkeit

Dies ist die wichtigste nichtfunktionale Anforderung an das System, da sehr hohe Erwartungen der zukünftigen Nutzer an die Bedienbarkeit gestellt werden. In diesem Zusammenhang soll die Erwartungskonformität und die Fehlertoleranz besonders erwähnt werden, da dies die größten Kritikpunkte am Altsystem waren. Das Vertriebssystem sollte daher bei Fehlern, wie z. B. falsche Eingaben, trotzdem seine Funktionsweise aufrechterhalten und in allen Situationen konsistent und den Erwartungen des Benutzers entsprechend reagieren.

Performance

Das System muss für maximal 60 Benutzer zur selben Zeit ausgelegt sein, im Durchschnitt etwa 10 Benutzer zur selben Zeit bewältigen können und dabei immer noch akzeptable Antwortzeiten liefern. Eine Verletzung dieser Anforderung würde zu einer Missstimmung der Nutzer führen und die Akzeptanz des Systems gefährden.

Portierbarkeit

Da das zu entwickelnde System webbasiert sein soll, ist es von Vorteil wenn es unabhängig von Hardware und Betriebssystem ablaufen kann. Dazu ist die Auswahl des Webservers und der Datenbank so zu treffen, dass eine Integration in eine bestehende Unternehmensumgebung leicht möglich ist. Des Weiteren sollte es möglich sein, das Vertriebssystem ohne Schwierigkeiten, auf schnellere Hardware oder evt. ein Computercluster zu portieren um die Performance zu verbessern.

Vertraulichkeit

Die Mandantenfähigkeit des Systems erfordert eine strikte Einhaltung der Vertraulichkeit. Ein Benutzer darf zu keinem Zeitpunkt Zugriff auf die Daten eines anderen Mandanten haben. Aber auch innerhalb eines Mandanten kann es sinnvoll sein, die Sichtbarkeit von Datensätzen (z. B. Aktivitäten) einzuschränken und diese z. B. nur innerhalb der Unternehmensgrenzen zur Verfügung zu stellen.

Weitere Anforderungen

Auf weitere nichtfunktionale Anforderungen wie Erweiterbarkeit, Wartbarkeit, Robustheit, Zuverlässigkeit und Korrektheit kann bei keinem Softwaresystem verzichtet werden. Auf diese Anforderungen soll hier im Einzelnen aber nicht näher eingegangen werden, sie werden jedoch bei der Entwicklung des Prototypen soweit wie möglich berücksichtigt.

3.4.3 Technische Randbedingungen

Das zu entwickelnde System sollte in der Programmiersprache Java implementiert werden. Dies ergibt sich aus der Firmenphilosophie der novomind AG, die fast ausschließlich mit der Programmiersprache Java arbeitet. Des Weiteren soll das System webbasiert sein, d. h. die zukünftigen Benutzer sollen es mit ihrem Browser bedienen können. Da das zu entwickelnde System der Open Source Gemeinde zur Verfügung gestellt werden soll, müssen die verwendeten Frameworks oder Bibliotheken ebenfalls als Open Source zur Verfügung stehen.

3.5 Untersuchung existierender Systeme

Die gesammelten Anforderungen an das neue Vertriebssystem werden nun mit bereits vorhandenen Open Source Lösungen verglichen. In einem Fazit wird die Entscheidung getroffen, ob ein vorhandenes System eingesetzt werden kann, oder ob eine Neuentwicklung gestartet werden soll.

Es gibt bereits eine Vielzahl an Open Source CRM Systemen auf dem Markt. Auf sourceforge.net sind momentan 457 CRM-Systeme registriert (Stand: 03.05.2008) (vgl. Sourceforge.net, 2008). Aus dieser Fülle von Systemen muss zur Marktübersicht daher eine Auswahl getroffen werden. Es werden demnach nur webbasierte Systeme untersucht, die sich, gemessen an der Downloadzahl, einer hohen Beliebtheit erfreuen und. Zudem sollten die Projekte nicht mehr als ein Jahr ohne Aktivität sein. Folgende Open-Source CRM Systeme werden näher betrachtet:

- SugarCRM (siehe: SugarCRM, 2007)
- vTiger CRM (siehe: vTiger, 2007)
- XRMS CRM (siehe: XRMS, 2007)
- Hipergate (siehe: hipergate, 2007)
- Openbravo ERP (siehe: OpenbravoERP, 2007)
- SalesPortal (siehe: SalesPortal, 2007)
- OpenTaps (siehe: OpenTaps, 2007)
- OpenCRX (siehe: OpenCRX, 2007)

Die Tabelle 3.1 zeigt eine Übersicht über diese Systeme. Sie enthält die Versionsnummern der heruntergeladenen Systeme, die genutze Technologie bzw. die genutze Datenbank und eine Beurteilung welchen Schwierigkeitsgrad die Installation des Systems hatte. Diese Beurteilung beschreibt eine subjektive Meinung und berücksichtigt z. B. wie hilfreich die Installationsanleitung war oder wie viel Konfigurationsaufwand das System nach der Installation erforderte.
Das System „OpenCRX“ konnte auch nach mehreren Versuchen nicht lokal installiert werden. Daher wurde für eine Analyse dieses System die vom Hersteller bereitgestellte Online Demo verwendet.

Im Folgenden sollen nun die drei erfolgversprechendsten Systeme: SugarCRM, vTiger CRM und OpenTaps näher vorgestellt und evaluiert werden. Die beiden Systeme SugarCRM und vTiger CRM sollen, trotz der Implementierung in der Programmiersprache PHP, ebenfalls

System	Version	Schwierigkeitsgrad der Installation	Technologie / Server / DB
SugarCRM	4.5.1	Leicht	PHP / Apache / MySQL
Hipergate	3.0.21	Schwer	Java / Tomcat / MySQL
Openbravo ERP	2.34	Leicht	Java / Tomcat / PostgreSQL
SalesPortal	1.2.1	Leicht	Java / Tomcat / PostgreSQL
XRMS CRM	1.99.2	Mittel	PHP / Apache / MySQL
vTiger CRM	5.0.3	Leicht	PHP / Apache / MySQL
OpenTaps	1.0.0p5	Leicht	Java / Tomcat / Derby
OpenCRX	1.11.0	Nicht erfolgreich -> Online Demo	Java / JBoss / MySQL

Tabelle 3.1: Untersuchte Open Source CRM Systeme

untersucht werden, da sie die ausgereiftesten Systeme sind. Obwohl ein Einsatz dieser Systeme nicht in Frage kommt, soll die Bedienbarkeit der Systeme untersucht werden, um Erfahrungen für eine mögliche Neuentwicklung zu sammeln. Von den in der Programmiersprache Java realisierten Systemen wurde OpenTaps ausgewählt, weil es die Anforderungen am ehesten erfüllt.

3.5.1 SugarCRM

Name:	SugarCRM
Version:	4.5.1
Website:	http://www.sugarcrm.com/
Online-Demo:	http://demo.sugarcrm.com/sugarcrm_os/
Programmiersprache:	PHP
Datenbank:	MySQL

Bei SugarCRM handelt es sich um das am weitesten verbreitete und das professionellste Open Source CRM System. Das Projekt ist im April 2004 auf Sourceforge gestartet und erreichte schnell großen Erfolg. Die Firma hinter diesem CRM System ist SugarCRM Inc. und beschäftigt rund 150 Mitarbeiter weltweit. (SugarCRM, 2008)

Die Firma bietet ihr CRM System in drei verschiedenen Versionen an: Enterprise, Professional und Community Edition. Hierbei ist nur die letzte Version unter einer Open Source Lizenz veröffentlicht. Die beiden anderen Versionen werden unter verschiedenen Geschäftsmodellen, wie z. B. eine ASP-Lösung, On-Site oder On-Demand, vertrieben.

Der Funktionsumfang der SugarCRM Community Edition umfasst, neben der Datenhaltung von Unternehmen, Kontaktpersonen und Aktivitäten, das Verwalten von Marketingkampagnen, ein Dokumentenmanagement, das Verwalten von Kundensupportanfragen und ein umfangreiches Reporting. Auch kollaborative Funktionen wie eine Email Integration sind enthalten.

Abbildung 3.2 zeigt die Startseite einer SugarCRM Installation. Die Hauptnavigation im oberen Bereich besteht aus Reitern zur Auswahl der Module. Unter der Hauptnavigation befindet sich eine Liste der zuletzt aufgerufenen Objekte. Links befindet sich eine kontextabhängige Navigation über die man schnell Zugriff zu hilfreichen Funktionen hat. Die Startseite ist, in ihrem Aufbau, für jeden Benutzer frei definierbar. Dabei kann aus einer Reihe von vorgefertigten Komponenten gewählt werden. In der Standardeinstellung findet man dort die nächsten geplanten Aktivitäten oder die eigene Salespipeline.

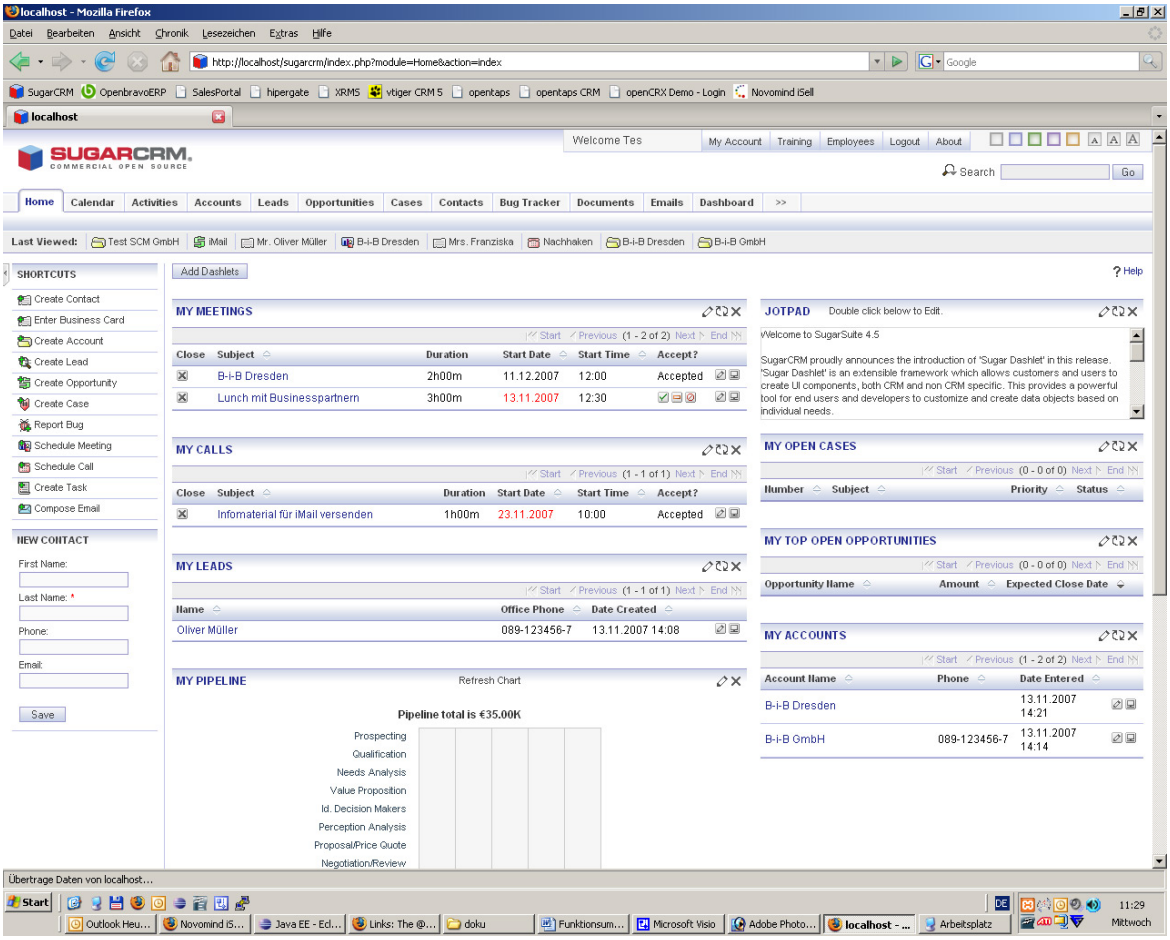

Abbildung 3.2: SugarCRM v4.5.1

3.5.2 vTiger

Name:	vTiger
Version:	5.0.3
Website:	http://www.vtiger.de/
Online-Demo:	http://5.vtiger-crm.de/
Programmiersprache:	PHP
Datenbank:	MySQL

vTiger CRM ist ein freies Open Source CRM System, das aus einer frühen Version von SugarCRM abgeleitet, und anschließend von einer Entwicklergemeinde weiterentwickelt wurde. Der Funktionsumfang umfasst etwas mehr als SugarCRM, nämlich zusätzlich noch eine Produktverwaltung, inklusive der Möglichkeit Preislisten zu erstellen, und einer Bestandsverwaltung mit der Möglichkeit Lieferanten und Einkaufsbestellungen zu verwalten. Auch in vTiger gibt es die Möglichkeit Emails direkt im System zu empfangen und zu versenden.

Die Abbildung 3.3 zeigt die Startseite einer vTiger Installation. Auch hier befindet sich die Hauptnavigation in Form von Reitern im oberen Bereich. Auch in vTiger hat man die Möglichkeit seine Startseite individuell zu gestalten und aus Komponenten seine favorisierten auszuwählen.

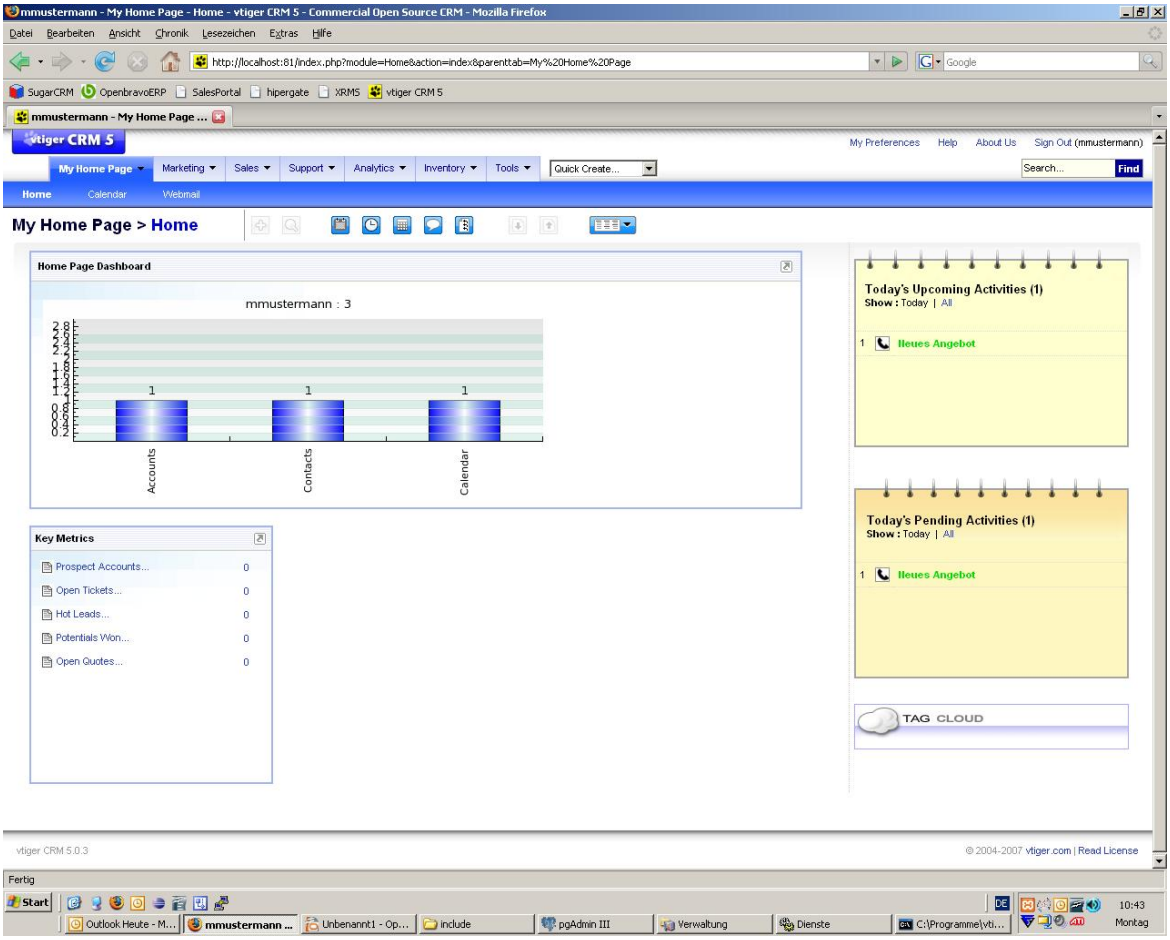

Abbildung 3.3: vTiger v5.0.3

3.5.3 OpenTaps

Name:	OpenTaps
Version:	1.0.0p5
Website:	http://www.opentaps.org/
Online-Demo:	http://demo1.opentaps.org/
Programmiersprache:	Java
Datenbank:	Derby

OpenTaps ist ein webbasiertes ERP und CRM System für kleine und mittlere Unternehmen (KMU). Es ist als freie Open Source Software veröffentlicht. OpenTaps ist das umfangreichste der getesteten Systeme, da es neben den CRM Funktionen auch die Aufgaben eines Enterprise Ressource Planning Systems erfüllt. Diese Funktionalität wurde allerdings für die Untersuchung nicht näher betrachtet.

Abbildung 3.4 zeigt die Startseite des CRM Moduls einer OpenTaps Installation. Der Aufbau ähnelt den zwei zuvor untersuchten Systemen. Auch bei OpenTaps ist die Hauptnavigation in Form von Reitern im oberen Bereich zu finden. Unter der Hauptnavigation ist, wie bei SugarCRM, eine Liste der letzten aufgerufenen Objekte. Im linken Bereich findet man ein kontextabhängiges Menü, das, wie bei SugarCRM, schnellen Zugriff zu wichtigen Funktionen bietet. In der Standardeinstellung befindet sich im Hauptbereich der Startseite ein Kalender, der in Tages-, Wochen- oder Monatsansicht die geplanten Aktivitäten anzeigt.

3.5.4 Evaluierung der getesteten Systeme

Die drei vorgestellten Systeme wurden zusammen mit den Mitarbeitern der novomind AG einem Usability Test unterzogen. Ziel war es, anhand vorher erstellter Aufgaben, die typischen genutzten Funktionen durchzuführen und dabei die Bedienbarkeit der drei Systeme zu bewerten. Da allerdings nur 3 Probanden gefunden wurden, konnte der Usability Test nicht in vollem gewünschten Umfang duchgeführt werden. Bei dieser geringen Anzahl an Probanden werden sicherlich nicht alle vorhandenen Probleme aufgedeckt. Das Ergebnis des Tests sollte daher auch unter Berücksichtigung dessen analysiert werden. Einen ausführlichen Usability Test, der auch einen Vergleichstest beinhalten wird, führt André Schmer in seiner Arbeit (Schmer, 2008) durch.

Die Aufgaben (siehe Anhang B), die den Probanden während des Tests gestellt wurden, sind so gewählt, dass sie innerhalb von 20 bis 30 Minuten zu lösen waren. Die Probanden sollen anhand der Aufgaben die typische Funktionen der CRM Systeme nutzen.

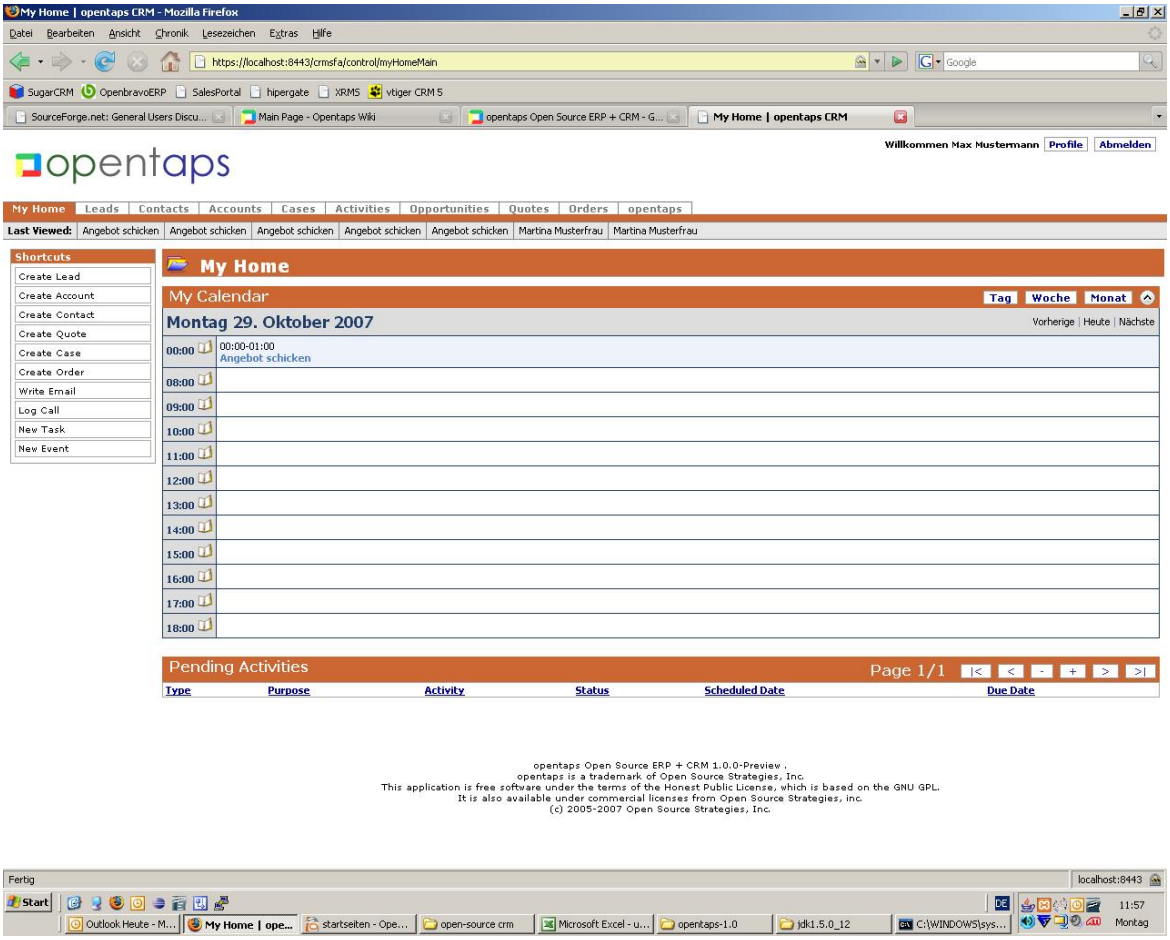

Abbildung 3.4: OpenTaps v1.0.0p5

Der Test soll Schwachstellen der Systeme bei deren Bedienbarkeit aufdecken und so die Einsetzbarkeit dieser Systeme testen und ggf. Hilfestellungen für eine Neuentwicklung liefern. Es wird vor allem die Selbstbeschreibungsfähigkeit der Systeme getestet, da die Probanden noch keine Erfahrung mit den getesteten Systemen haben und daher auf eine intuitive Bedienbarkeit bzw. nützliche Beschreibungen angewiesen sind.

Ergebnis der Untersuchung

Die Untersuchung ergab, dass die getesteten Systeme ohne detaillierte Einführung nur sehr schwer zu bedienen waren. Alle Probanden hatten Probleme, die benötigten Funktionen auf Anhieb zu finden. Dadurch brauchten sie für eine Aufgabe längere Zeit, oder konnten die Aufgabe ohne Hilfe nicht lösen.

Am Besten bewertet wurde SugarCRM, weil es den Probanden dort am einfachsten fiel, die gestellten Aufgaben zu meistern. Es bot eine sehr gut strukturierte Oberfläche und war im Großen und Ganzen intuitiv bedienbar. Dabei wurden aber auch Kritikpunkte gefunden. So ist die Präsentation der Stammdaten z. B. eines Accounts nicht optimal gelöst und wirkt eher ungeordnet. Hier wäre eine Strukturierung in Gruppen wünschenswert.

vTiger CRM wurde von den Probanden etwas schlechter bewertet als SugarCRM. Obwohl es auf den ersten Blick einen aufgeräumteren Eindruck macht und etwas intuitiver zu bedienen ist, muss es aber dadurch Funktionen verstecken, die dann nur schwer zu finden sind. Als Beispiel soll hier der Plus-Button zum Anlegen von Datensätzen genannt werden, der erst bei genauem Hinsehen auffällt. Diese Problematik ist in SugarCRM durch eine Seitennavigation mit wichtigen Funktionen besser gelöst.

Da der Aufbau von OpenTaps dem von SugarCRM sehr ähnelt, ist es auch ähnlich zu bedienen. Allerdings ist die Aufbereitung der Informationen und die Anordnung der Eingabemasken schlechter und unübersichtlicher als in SugarCRM. Vor allem die Visualisierung der Beziehungen zwischen den Entitäten ist in OpenTaps am schlechtesten gelungen, da die Informationen nicht übersichtlich aufbereitet sind.

Obwohl es nicht Teil der Untersuchung war, soll die Bedienung des Administrationsbereiches der Systeme aber trotzdem kurz erwähnt werden. Hierbei ist vor allem OpenTaps negativ aufgefallen, denn durch die Komplexität und Fülle von Funktionen ist die Administration dieses Systems kaum bedienbar. Das Anlegen von Benutzern gestaltete sich schon problematisch. Die beiden Systeme SugarCRM und vTiger CRM waren dagegen intuitiv bedienbar und eine Administration war ohne Probleme möglich.

3.6 Fazit

Zusammenfassend bleibt zu sagen, dass die gestellten Anforderungen an ein Vertriebssystem von keinem der untersuchten Systeme zufriedenstellend erfüllt werden. Einige der Systeme boten zu viel Funktionalität, es waren komplette ERP Systeme, andere boten zu wenig Funktionalität. Viele der getesteten Systeme waren kaum bedienbar, oder die Daten wurden nicht vorteilhaft aufbereitet.

Dabei bleibt festzuhalten, dass keines der untersuchten Systeme die Möglichkeit bot, eine Unternehmensstruktur abzubilden und die Sicherheitseinstellungen dementsprechend granular vorzunehmen. Da das Hinzufügen dieser Funktion in ein bereits existierendes System einen weitreichenden Eingriff in dieses zur Folge hätte, wird eine Neuentwicklung angestrebt. Außerdem sind die ausgereiftesten Systeme, die für eine solche Veränderung in Frage kämen, in der Programmiersprache PHP implementiert und können daher für einen Einsatz bei der novomind AG nicht genutzt werden.

Aus dieser begründeten Entscheidung für eine Neuentwicklung eines Vertriebssystems geht zugleich das Alleinstellungsmerkmal hervor. So wird es in dem zu entwickelndem System möglich sein, die Unternehmensstruktur abzubilden. Mit Hilfe dieser Funktion ist es dem einsetzenden Unternehmen möglich, Insellösungen in Tochtergesellschaften abzuschaffen und ein einheitliches System für die Verwaltung der Kundenkommunikation einzuführen. Dabei wird es die Möglichkeit geben, die Sicherheitseinstellungen dementsprechend fein granular vorzunehmen, so dass sensible Daten nicht über die Unternehmens- oder Abteilungsgrenzen hinaus verfügbar sein werden.

Durch diese Alleinstellungsmerkmale wird zudem ein Erfolg in der Veröffentlichung des zu entwickelnden Vertriebssystem als Open Source Software erwartet. Da kein System gefunden wurde, das diese Funktionalität bietet, ist dies eine Lücke, in die sich das System platzieren wird.

Aus den Usability Untersuchungen werden zudem Erfahrungen für die Entwicklung der Benutzeroberfläche gezogen. So sollen die Daten übersichtlich präsentiert und wenn möglich zu Gruppen zusammengefasst werden. Die Präsentation der Beziehungen zwischen den Objekten soll alle benötigten Informationen bieten, dabei aber nicht zu dominant wirken, oder ggf. ausblendbar sein. In dem folgenden Kapitel wird die Architektur für die Neuentwicklung des Vertriebssystems vorgestellt.

4 Architektur

Der Architekturentwurf, der in diesem Kapitel beschrieben wird, stellt in fachlicher, aber auch technischer Hinsicht eine Beschreibung des zu entwickelnden Systems dar. Dazu wird, aufbauend auf der im vorherigen Kapitel erarbeiteten Vision, das System mit seinen Systemkomponenten und deren Beziehungen untereinander beschrieben.

4.1 Fachliche Architektur

Das zu entwickelnde System lässt sich in folgende Komponenten aufteilen (siehe Abbildung 4.1): Die Datenhaltung, die Authentifikation & Autorisation, die Komponenten für Accounts, Contacts, Activities, Opportunities und die Komponente für Reports und Selektionen.

Die *Datenhaltung* ist zuständig für die Persistenz der Daten, ermöglicht den Zugriff auf bereits angelegte und die Speicherung neuer Daten. Die anderen Komponenten greifen lesend bzw. schreibend auf diese zu. Die Datenhaltung bildet somit die Basis des zu entwickelnden Systems.

Die *Authentifikation & Autorisation* ist zum einen dafür zuständig, den Benutzer gegenüber dem System zu authentifizieren. Dies erfolgt mittels einer Login Seite, auf die der Benutzer beim ersten Zugriff auf das System umgeleitet wird. Zum Anderen werden bei der Autorisation die Rechte des Benutzer beim Zugriff auf Objekte überprüft und ihm ggf. der Zugriff verweigert.

Die Komponenten für *Accounts*, *Contacts*, *Activities*, *Opportunities* und *Reporting / Selektion* sind zuständig für den Zugriff auf die jeweiligen Objekte. Sie bieten dem Benutzer Mechanismen zum Auffinden, Erstellen und Löschen von Daten. Das Auffinden von Daten erfolgt idealerweise mittels einer Liste der im System abgespeicherten Daten und der Möglichkeit diese Liste mit einem Filter einzuschränken.

Über die *Administration* kann ein Benutzer, der die erforderlichen Rechte besitzt, das System verwalten und sämtliche Einstellungen vornehmen.

Die gesamte Funktionalität wird dem Benutzer über eine *Benutzungsschnittstelle* zur Verfügung gestellt.

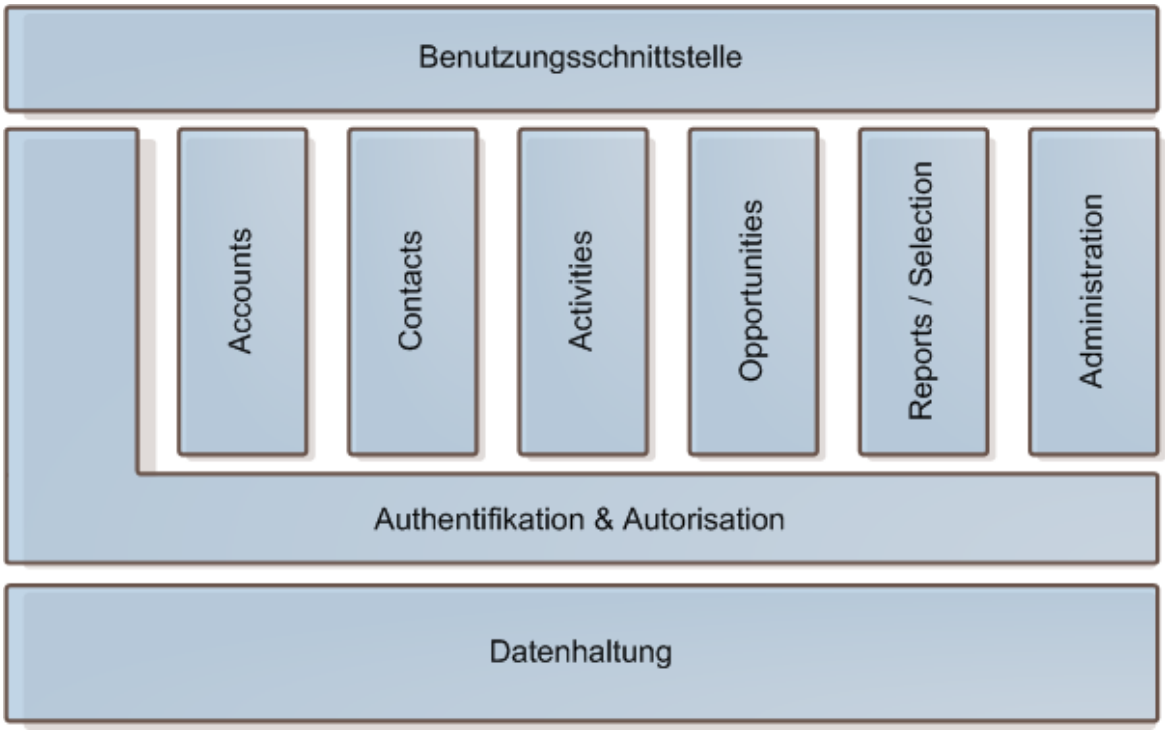

Abbildung 4.1: Komponenten

4.1.1 Businessprofil

Das Businessprofil, das in dem Vertriebssystem abbildbar ist, ist folgendermaßen aufgebaut. Abbildung 4.2 gibt einen Überblick über die Objekte und deren Beziehungen.

Auf oberster Ebene gibt es die Möglichkeit *Mandanten (Tenant)* abzubilden. Dadurch hat der Benutzer die Möglichkeit, mehrere voneinander unabhängige Instanzen des Systems zu nutzen. Innerhalb eines Mandanten kann der Benutzer eine oder mehrere *Firmengruppen (Group)* anlegen. Eine Firmengruppe kann aus mehreren *Firmen (Company)* bestehen, die wiederum mehrere *Business Units* beherbergen können. Eine Firma kann mehrere *Standorte (Location)* haben an denen sie tätig ist. Ein *Mitarbeiter (Employee)* einer Firma wird aber nur genau einem dieser Standorte zugeordnet. Außerdem ist ein Mitarbeiter in genau einer Business Unit tätig. Ein Mitarbeiter kann als *Benutzer (User)* des Systems eingerichtet werden, es ist aber nicht zwingend erforderlich. Falls z. B. nur Zuständigkeiten eines Mitarbeiters für einen Kunden abgespeichert werden sollen, muss dem Mitarbeiter kein Benutzerzugang eingerichtet werden.

Der Vorteil einer solchen Konfiguration ist die Möglichkeit, das Vertriebssystem unternehmensweit zu nutzen, ohne dabei sämtliche Vertriebsdaten allen Mitarbeitern öffentlich zu

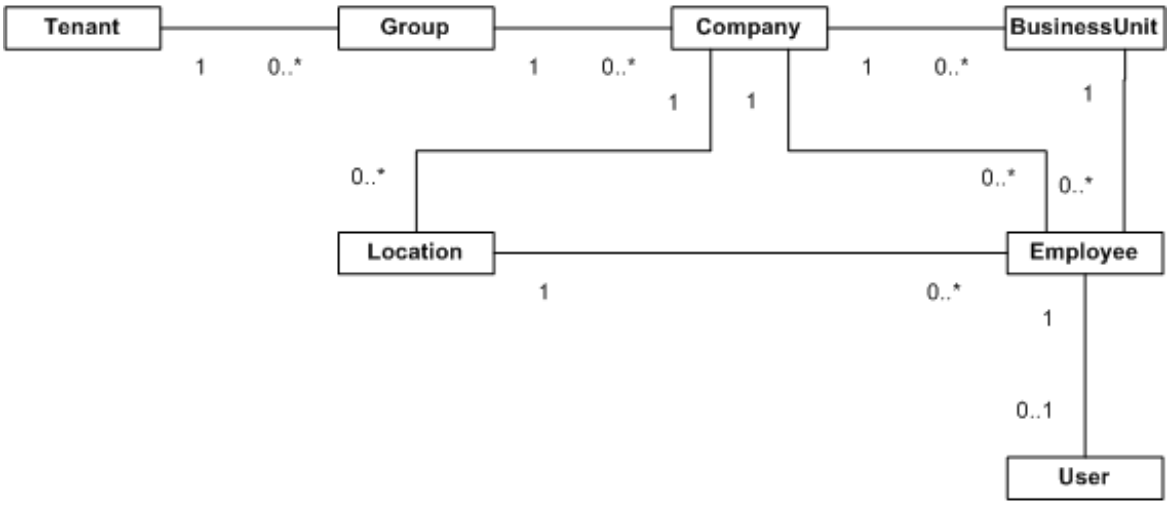

Abbildung 4.2: Businessprofil

machen. Das System könnte z. B. so eingerichtet werden, dass Aktivitäten und Opportunities nur innerhalb einer Firma sichtbar sind. Mitarbeiter einer Tochterfirma könnten dann auf die im Vertriebssystem abgespeicherten Unternehmen und Kontaktpersonen zugreifen, ohne Zugang zu möglicherweise sensiblen Aktivitätendaten zu haben.

Abbildung 4.3 zeigt beispielhaft ein mögliches Businessprofil, wie es ein Benutzer im System abbilden könnte. Es besteht aus einem Mandanten, der eine Unternehmensgruppe beinhaltet. Die Unternehmensgruppe besteht aus zwei Unternehmen, die jeweils zwei Business Units haben.

4.1.2 Datenmodell

Bei dem Datenmodell (siehe Abbildung 4.4) handelt es sich um eine übersichtliche Darstellung des zugrundeliegenden Datenbankmodells. So sind in diesem Modell vor allem die Beziehungen zwischen den fachlichen Objekten zu erkennen.

Die Datenhaltung im Vertriebssystem besteht, sehr stark abstrahiert, aus den vier Entitäten Account, Contact, Activity und Opportunity. Ein Account, also ein im Vertriebssystem abgespeichertes Unternehmen, hat, neben Stammdaten wie einem Namen, einer Telefonnummer oder einer Homepage, einen Typ und eine Branche in der es tätig ist. Der Typ beschreibt, wie der Account mit dem eigenen Unternehmen in Verbindung steht, z. B. ein Kunde, ein Partner, oder ein Zulieferer. Es können auch Beziehungen von Accounts untereinander gespeichert werden, um Verknüfungen zwischen einem Unternehmen und seinem Mutterkonzern oder

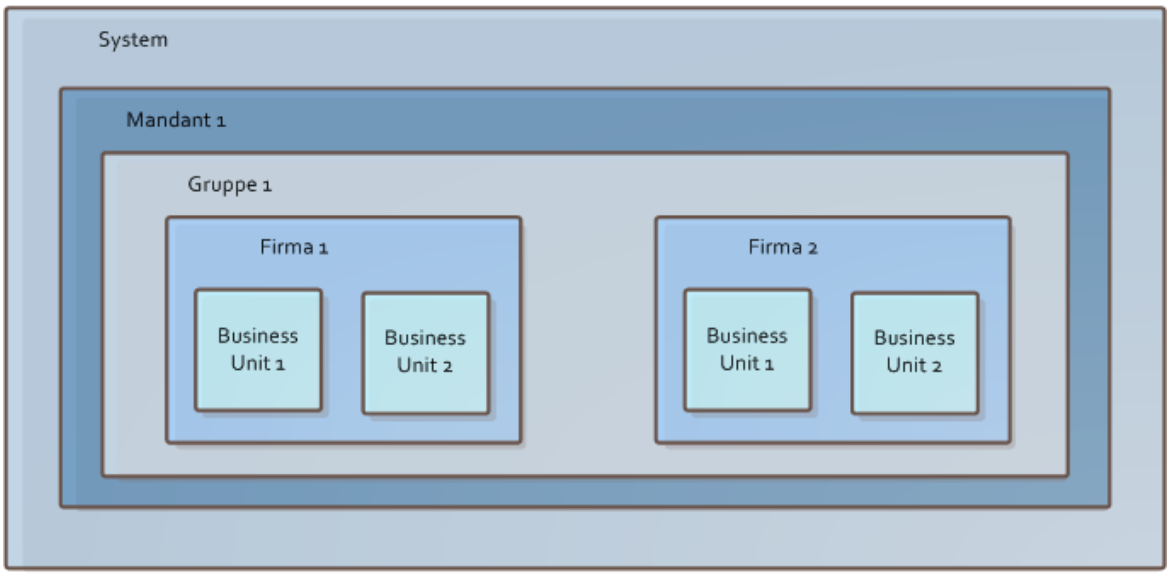

Abbildung 4.3: Beispiel Businessprofil

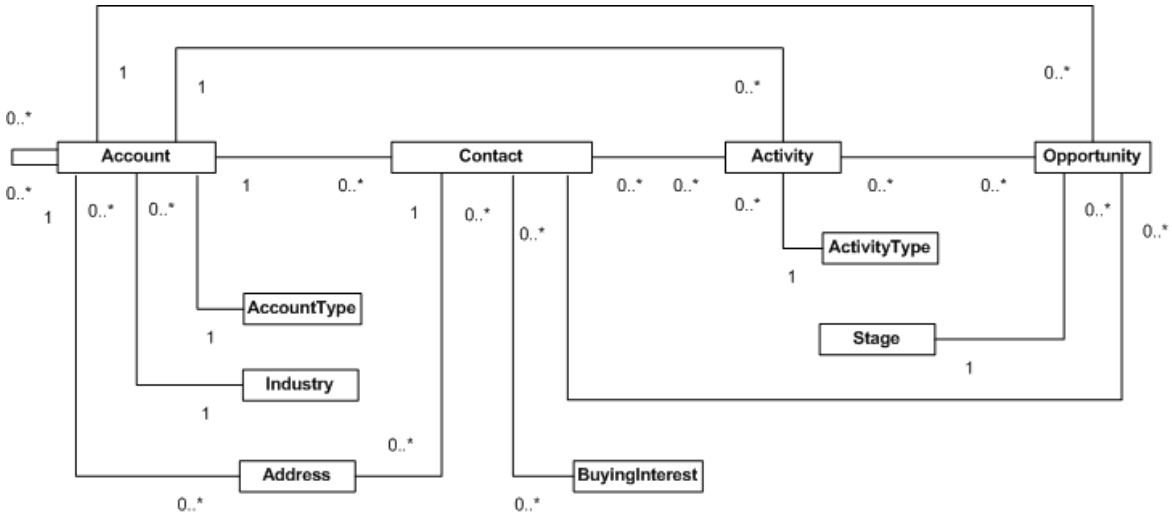

Abbildung 4.4: Übersicht des Datenmodells

zwischen zwei Partnerunternehmen hinterlegen zu können. Zu Accounts kann man beliebig viele Kontakpersonen definieren. Eine Kontaktperson kann, neben Stammdaten wie einem Namen, dem Geburtsdatum oder den Telefonnummern, eine Reihe von Kaufintressen besitzen. Die Kaufinteressen sind vor allem für Marketingaktionen enorm wichtig. Zu beiden Entitäten (Accounts und Contacts) können beliebig viele Adressen zugeordnet werden. Jede Adresse ist von einem Typ, z. B. Lieferadresse oder Rechnungsadresse. Einem Account können zudem Aktivitäten zugeordnet werden. Aktivitäten beschreiben i.d.R. Telefongespräche, Treffen oder Wiedervorlagen. Einer Aktivität kann man außerdem Kontaktpersonen zuteilen. Dabei wird z. B. bei Telefongesprächen der Gesprächspartner hinterlegt. Opportunities werden auch einem Account zugeordnet und können Verknüpfungen zu Aktivitäten oder Kontaktpersonen haben.

4.1.3 Benutzerrollen

Da das System über eine rollenbasierte Benutzerverwaltung verfügen soll, werden im Folgenden die benötigten Rollen identifiziert.

Die in diesem Abschnitt beschriebene Zuordnung der Rechte auf Benutzerrollen soll nur eine Empfehlung darstellen und ist bei einer späteren Implementierung des Vertriebssystem flexibel zu lösen. Die Konfiguration ist sicherlich von der Größe des Unternehmens und von den vorliegenden Strukturen abhängig. Daher sollte die Zuordnung durch einen Administrator änderbar sein, um das Vertriebssystem an die eigenen Bedürfnisse und an die eigene Firmengröße anzupassen.

Die wichtigste Benutzerrolle im System, ist die Rolle des Systemadministrators. Benutzer, die diese Rolle innehaben, müssen in der Lage sein, Mandanten im System anzulegen, zu verwalten und ggf. zu löschen. Das Anlegen eines Mandanten erfordert zugleich das Anlegen mindestens einer Firmengruppe, einer Firma und einer Business Unit. Des Weiteren dürfen diese Benutzer neue Benutzer im System registrieren und entfernen. Zur Zeit der Auslieferung des Vertriebssystems muss mindestens ein Benutzer eingerichtet sein, der diese Rolle innehat, da sonst eine Konfiguration des System und die Einrichtung neuer Benutzer nicht möglich ist.

Benutzer in der Rolle des Administrators dürfen, im Gegensatz zu solchen in der Rolle des Systemadministrators, Einstellungen nur innerhalb des ihnen zugeordneten Mandanten vornehmen. Dazu zählt auch die Möglichkeit neue Benutzer im System zu registrieren, oder neue Firmengruppen, Firmen und Business Units anzulegen.

Die Rolle des Vertriebsleiters ermöglicht einem Benutzer die Einsicht in sämtliche Daten, sowie das Anlegen und Löschen dieser.

Die Rolle des Vertriebsmitarbeiters ist die am niedrigsten eingestufte Benutzerrolle und sollte jedem neuen Benutzer standardmäßig zugeteilt werden. Der Benutzer hat dann die Möglichkeit, auf alle Accounts und Kontakte zuzugreifen und neue anzulegen. Das Löschen ist ihm nicht gestattet. Auf Aktivitäten und Opportunities kann er nur begrenzt zugreifen. Er hat nur zu solchen Zugang, bei denen die Zuständigkeit in seiner Business Unit liegt.

4.2 Technische Architektur

4.2.1 Schichten Architektur

Gemäß dem in Kapitel 2.2 beschriebenen Java EE Application Model baut das neue Vertriebssystem auf einer 4-Schichten Architektur auf. Dabei wird das System in Komponenten zerlegt, die in ihre jeweiligen Aufgabenbereiche eingeteilt werden. Aus dieser Aufteilung ergeben sich folgende Schichten (siehe Abbildung 4.5):

- Client
- Präsentation
- Anwendungslogik
- Datenhaltung

Für die Datenhaltung im Vertriebssystem wird ein relationales Datenbanksystem eingesetzt. Die Anwendungslogik, die auch die Persistenzabbildung beinhaltet, wird in einem Anwendungsserver ablaufen. Die Kommunikation der Anwendungslogik mit der Datenbank erfolgt über JDBC und ist somit datenbankunabhängig. Die Präsentation ist aufgeteilt in eine Server-seitige und in eine Client-seitige Komponente. Der Webserver ist für die Generierung der Webseiten zuständig. Auf Seite des Clients wird aber zum Teil asynchrone Kommunikation eingesetzt, die für eine bessere Benutzbarkeit sorgt. Der Client benötigt für die Benutzung des Vertriebssystems daher einen Web-Browser, der Java Script fähig ist.

Die Aufteilung des Systems in solch eine Architektur hat den Vorteil, dass Skalierbarkeit und Wartbarkeit erhöht werden. So ist es leicht möglich, die einzelnen Schichten auf verschiedene Computer zu verteilen. Die Datenhaltung kann so z. B. auf einem separaten Computer verlagert werden, um die Perfomance des Systems zu steigern. Auch können einzelne Schichten unter Einhaltung der Schnittstellen ausgetauscht werden. Die Präsentationsschicht, die webbasiert ist, könnte z. B. durch einen Java Client ersetzt werden, ohne Anpassungen in den übrigen Schichten nötig zu machen.

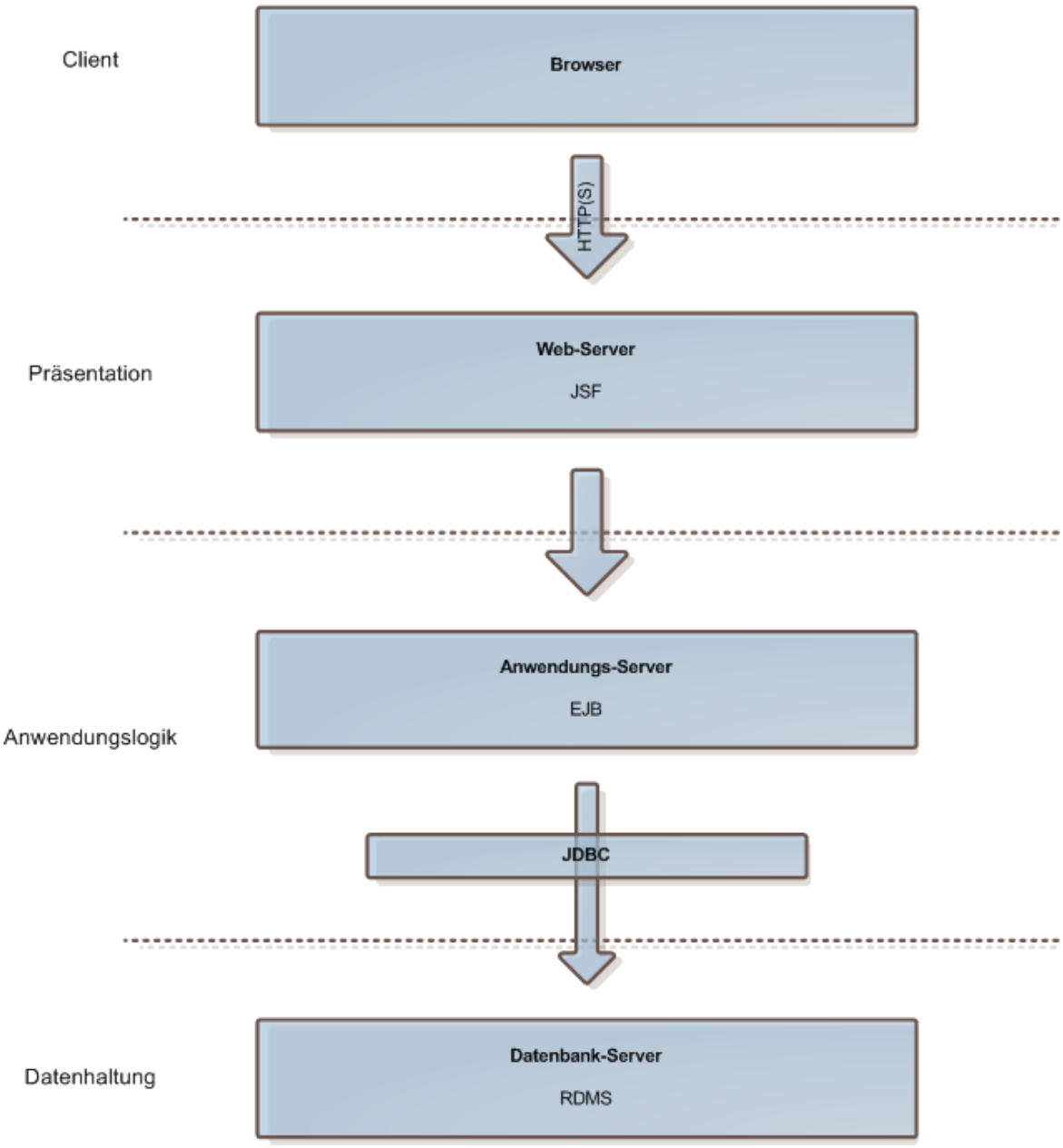

Abbildung 4.5: 4-Schichten Architektur

4.2.2 Klassenmodell

Das Paketdiagramm (siehe Abbildung 4.6) zeigt die Aufteilung der Klassen des Systems in Pakete und deren Beziehungen untereinander und zeigt daher die statische Sicht auf das System.

Das Paket *Entities* beinhaltet alle identifizierten Datenobjekte, die i.d.R. einfache Java-Objekte sind, aber vom Anwendungs-Server auch für die Persistenzabbildung genutzt werden. Diese Objekte werden durch das gesamte System transportiert und daher von den anderen Pakten importiert. Sie werden in den Schichten Anwendungslogik und Präsentation genutzt. Als Beispiel sei hier die Klasse *Account* genannt, die ein im Vertriebssystem abgebildetes Unternehmen darstellt. Die Klasse bietet nur Getter und Setter Methoden an, um Zugriff zu ihren Variablen zu ermöglichen.

Das Paket *Services* beinhaltet die Interfaces und Klassen, die für die Präsentationsschicht Methoden zur Verfügung stellen, mit Hilfe derer u. a. das Lesen aus und das Schreiben in die Datenbank realisiert werden. Nach Außen wird die Implementierung eines Services nur durch sein Interface angesprochen und bleibt somit austauschbar. Diese Interfaces bieten zudem eine Fassade, die Implementierung vor den oberen Schichten versteckt. Dieses Paket bildet die Anwendungslogik-Schicht. Als Beispiel sei hier das Interface *AccountServiceLocal* genannt, das Methoden definiert, um z. B. Accounts aus der Datenbank zu lesen, in die Datenbank zu schreiben oder aus der Datenbank zu löschen. Die Klasse *AccountServiceBean* implementiert dieses Interface und realisiert dessen Methoden.

Presentation enthält die Klassen, die in der Präsentionsschicht notwendig sind, um auf die Services zuzugreifen und die Daten anzuzeigen oder anzulegen. Die Klasse *ServiceLocator* implementiert das Service Locator Pattern (siehe: Sun Microsystems Inc., 2002) und bietet somit einen zentralen Zugriff auf die Services. Die Seiten *(JSF)*, die angezeigt werden, sind in dieser Darstellung ausgelagert, gehören jedoch zur Präsentationsschicht. Das Paket *Presentation* bildet zusammen mit den JSF Seiten die Präsentationschicht.

4.2.3 Ablaufdiagramm

Das Ablaufdiagramm (siehe Abbildung 4.7) zeigt, im Gegensatz zum Klassenmodell, die dynamische Sicht auf das System. Beispielhaft ist in der Abbildung der Aufruf eines Benutzers zur Darstellung aller Accounts dargestellt.

Der Benutzer wählt hierbei über die Oberfläche die Ansicht aller Accounts aus. In der darunter liegenden Klasse *AccountsManagerBean* wird dabei zunächst über den *ServiceLocator* die Referenz zum Service erfragt. Ist diese Referenz vorhanden, wird über das Interface *AccountServiceLocal* beim implementierenden Service nach der Liste aller Accounts gefragt.

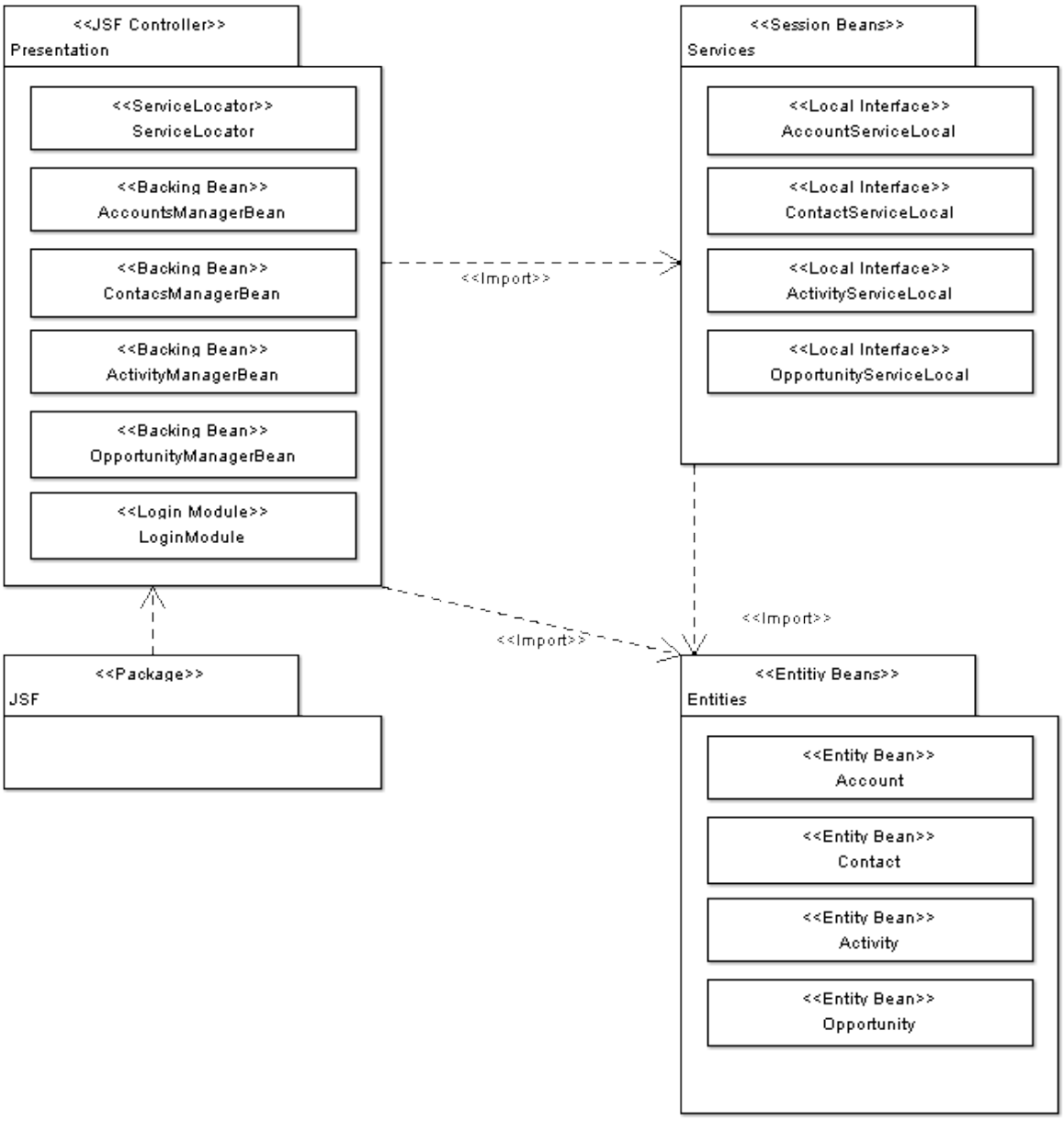

Abbildung 4.6: Pakete

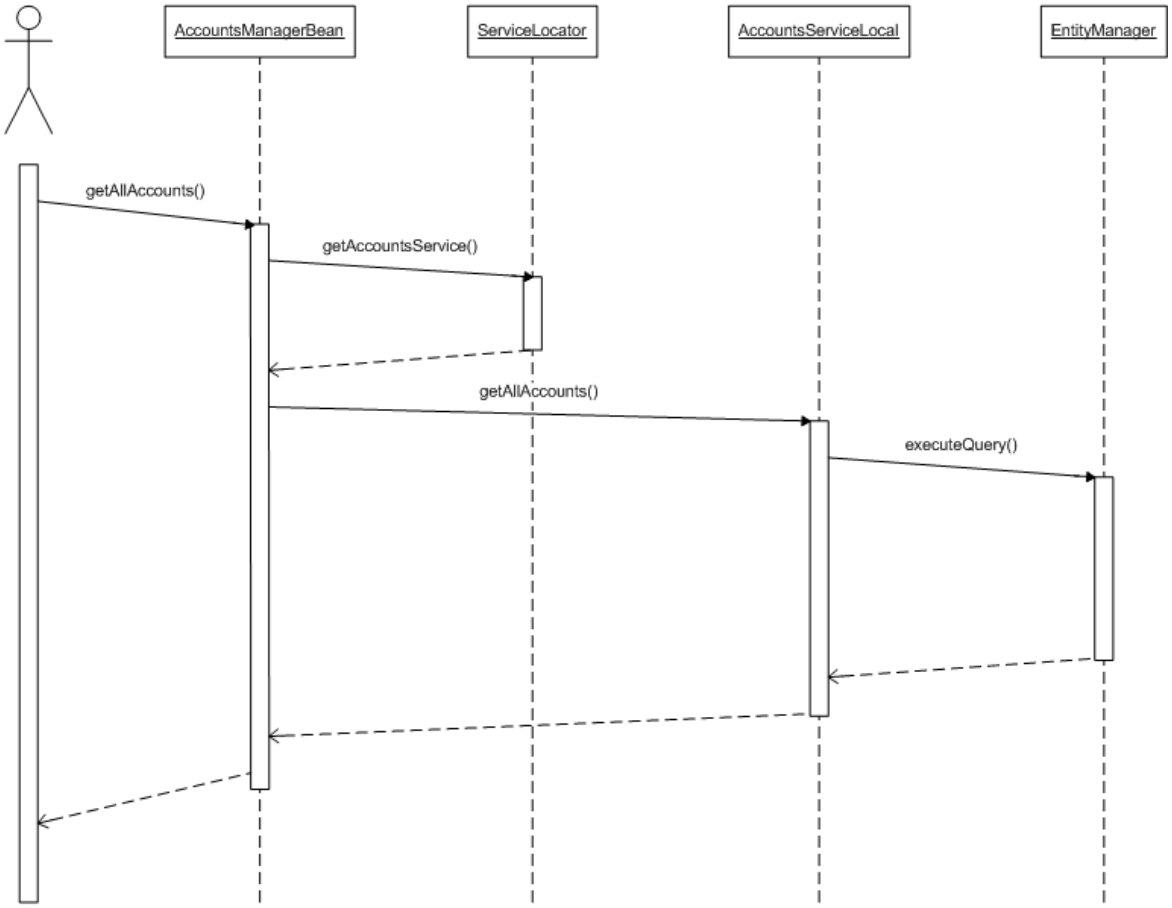

Abbildung 4.7: Ablaufdiagramm

Dieser Service erlangt die Liste der Accounts über den EntitiyManager, der von Java EJB angeboten wird.

4.3 Oberflächenstruktur

Die Oberfläche des Vertriebssystem soll so entworfen werden, dass sie dem Corporate Design der novomind AG entspricht und eine hohe Bedienbarkeit bietet. Da das System aber als Open Source zur Verfügung gestellt werden soll, muss die Oberfläche leicht austauchbar bleiben. Daher wird eine Templateengine und CSS zur Trennung von Inhalt und Darstellung genutzt. Zur Strukturierung wird die Oberfläche in folgende Bereiche eingeteilt (siehe Abbildung 4.8).

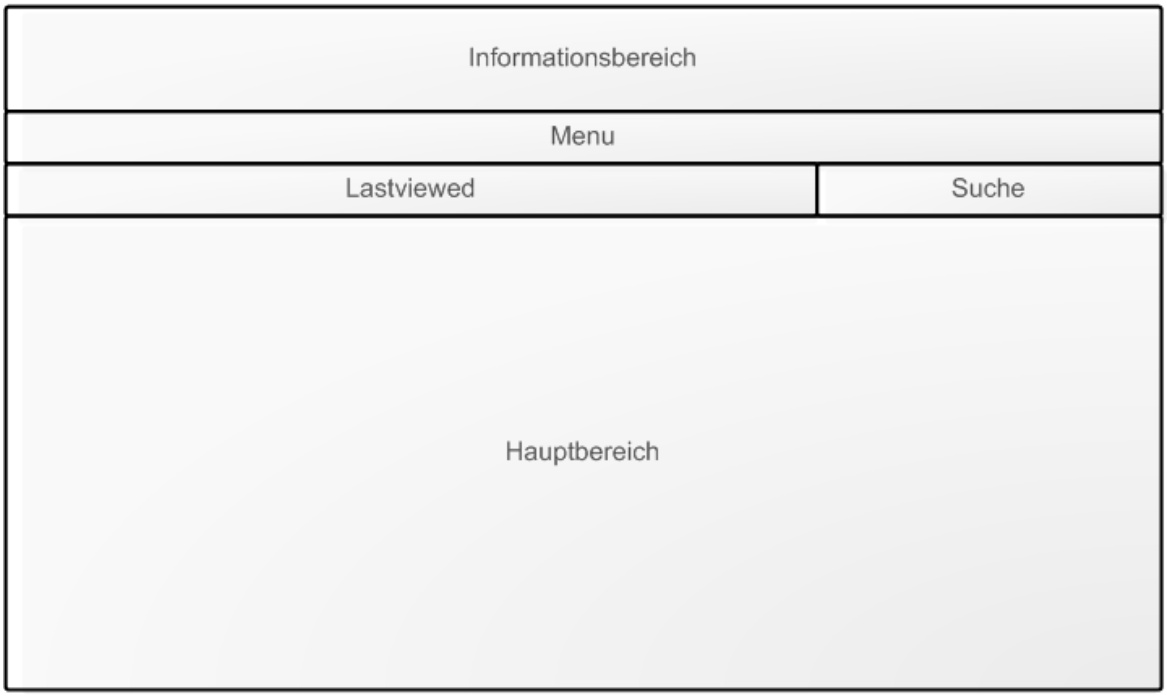

Abbildung 4.8: Struktur der Bedienoberfläche

Der obere Bereich dient zum Anzeigen des Unternehmenslogos und der Daten des angemeldeten Benutzers. Es wird der Name des Benutzers angezeigt, aber auch das Unternehmen und die Business Unit, der der Benutzer zugeordnet ist. Außerdem befinden sich in diesem Bereich die Schaltflächen zum Abmelden und zur Hilfefunktion. Darunter befindet sich das Menü, das in Reiter unterteilt ist. Hier hat der Benutzer die Möglichkeit, die verschiedenen

Komponenten (Accounts, Contacts, Activities, Opportunities und Reports / Selektion) auszuwählen. Unter dem Menü werden dem Benutzer die letzten besuchten Entitäten angezeigt. Dies dient vor allem dem schnelleren Auffinden von häufig besuchten Entitäten. Daneben befindet sich das Eingabefeld der globalen Suchen, die es dem Benutzer ermöglicht, innerhalb aller Datensätze zu suchen. Der Hauptbereich beinhaltet kontextabhängig alle Informationen, die angezeigt werden sollen.

5 Prototyp

Das entworfene Konzept eines Vertriebssystem wird nun in einem explorativem Prototypen umgesetzt. Der Prototyp soll in erster Linie die Umsetzbarkeit des Konzepts beweisen. Dabei wird nur die grundlegende Funktionalität implementiert. Abbildung 5.1 zeigt die Startseite des entwickelten Prototypen.

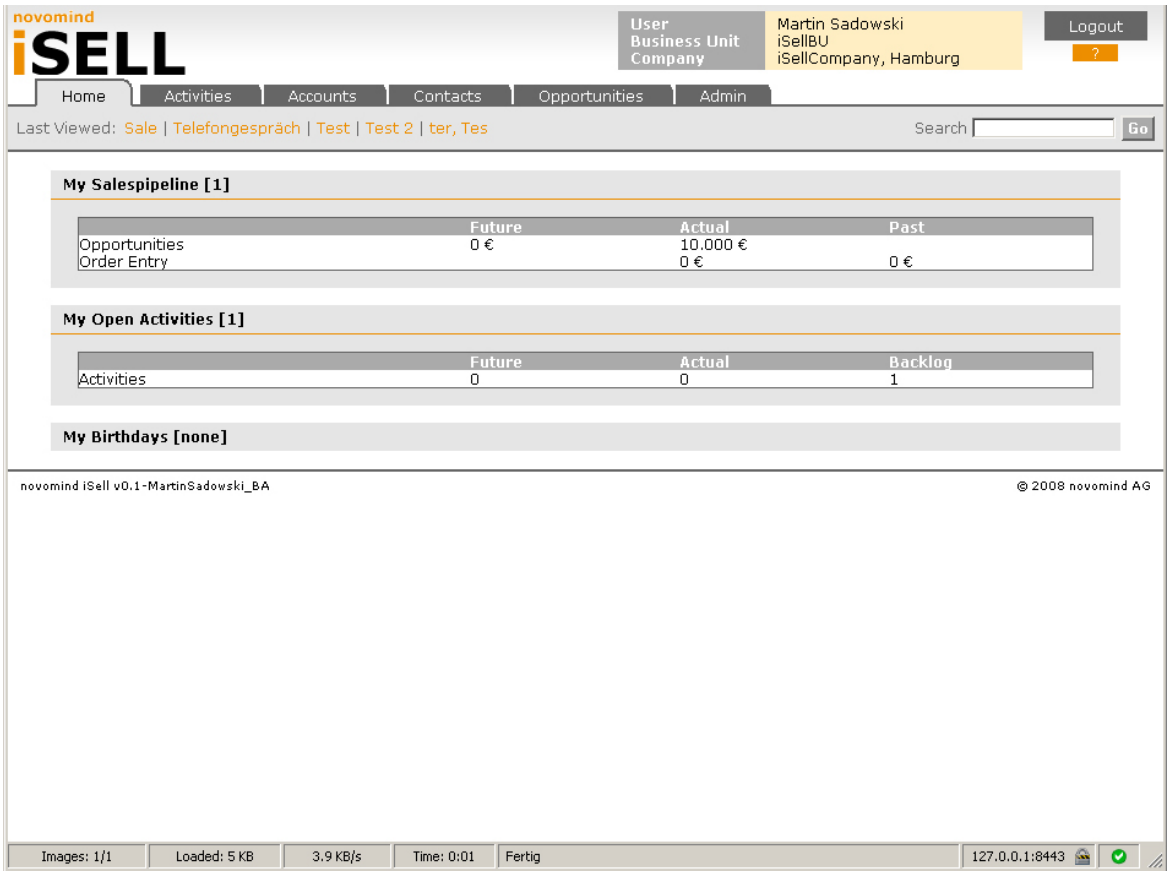

Abbildung 5.1: Screenshot Prototyp

5.1 Zielsetzung

Der Prototyp wird mit dem Ziel entwickelt, die Umsetzbarkeit des in den vorigen Kapiteln entwickelten Konzeptes zu beweisen. In fachlicher Hinsicht sollen mögliche Lücken im Konzept aufgedeckt und geschlossen werden. In technischer Hinsicht soll, durch Entscheidungen bei der Implementierung, die Grundlage für ein marktreifes Produkt entwickelt werden.
Um dieses Ziel zu erreichen, werden in dem Prototypen nur lesende Funktionen möglich sein. Dadurch wird die Aufmerksamkeit der zukünftigen Benutzer nicht auf fehlerhafte Eingabemasken gelenkt und sie können die Vollständigkeit des Konzepts besser beurteilen. Hilfreich für diese Beurteilung ist es, die Vertriebsdaten des Altsystems im Prototypen zugänglich zu machen. Diese Daten stehen als Dateien im CSV-Format (Comma-Separated Values) zur Verfügung und werden über eine Importschnittstelle in das System eingelesen.

5.2 Realisierung

Der Prototyp wurde als Java EE Anwendung mit webbasierter Oberfläche realisiert. Zum Betreiben des Prototypen ist folgende Ablaufumgebung notwendig.
Als Application Server wird JBoss (Red Hat Inc.) in der Version 4.2.1 verwendet. Der JBoss Application Server ist ein Open Source Projekt, das von Red Hat Inc. unterhalten wird, und bietet eine Implementierung eines Application Servers nach dem Java EE-Standard.
Des Weiteren wird als Datenbank eine MySQL Installation in der Version 5.0 vorausgesetzt.
Auf der Client-Seite wird ein moderner Web-Browser benötigt, der JavaScript fähig ist.

5.3 Ausgewählte Aspekte der Realisierung

Im Folgenden sollen nun ausgewählte Aspekte der Umsetzung des Prototypen näher erläutert werden.

5.3.1 Navigation

Die Abbildung 5.2 gibt eine Übersicht über die Möglichkeiten der Navigation innerhalb des entwickelten Prototypen. Nach dem Login beginnt der Benutzer auf der Startseite (Index). Die Startseite wird im nächsten Abschnitt näher erläutert. Von der Startseite aus hat der Benutzer die Möglichkeit, in jede der Komponenten Accounts, Contacts, Activities oder Opportunities zu wechseln. Auf der darauf folgenden Seite sieht der Benutzer eine Liste der in

der Datenbank vorhandenen Objekte mit der Möglichkeit, diese Liste über einen Filter einzuschränken. Mit einem Klick auf den Namen des jeweiligen Objekts gelangt der Benutzer zur Detailansicht. Hier sieht er sämtliche gespeicherten Informationen zu dem Objekt.
Im unteren Bereich der Detailansicht sind die Beziehungen des ausgewählten Objekts zu anderen aufgelistet. So sieht der Benutzer z. B. in der Detailansicht eines Accounts die Contacts, die Activities und die Opportunities, die diesem Account zugeordnet sind.
Zudem kann der Benutzer auf jeder Seite die globale Suche erreichen. In dem entwickelten Prototypen wurde die globale Suche nur innerhalb der Accounts und Contacts realisiert. Von den Suchergebnissen gelangt der Benutzer über einen Klick auf den jeweiligen Namen, direkt zur Detailansicht.

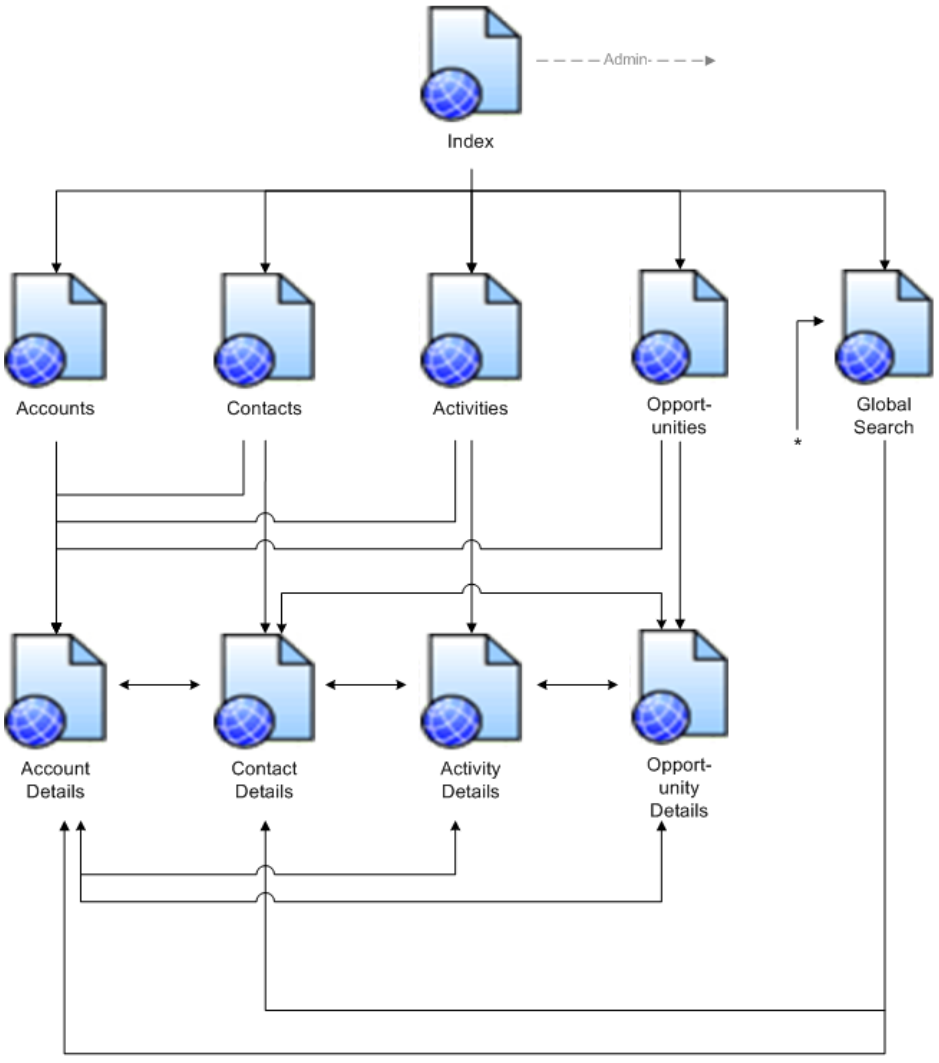

Abbildung 5.2: Navigation innerhalb des Prototypen

5.3.2 Startseite

Die Startseite des Prototypen wurde so entwickelt, dass sie einen guten Einstieg in das Vertriebssystem bietet, aber auch einen gewissen didaktischen Wert hat. Die Startseite ist hervorragend geeignet, um wichtige Informationen anzuzeigen, denn jeder Benutzer des Vertriebssystem muss seine Arbeit auf dieser Seite beginnen. So werden in dem entwickelten Prototypen folgende Informationen auf der Startseite angezeigt.
Der Benutzer sieht seine persönliche Sales Pipeline (siehe Abbildung 5.3). Diese besteht aus aufsummierten erwarteten Auftragseingängen von Opportunities, für die der Benutzer verantwortlich ist. Es werden die Opportunities geordnet nach folgenden Kriterien angezeigt. Unterschieden wird zwischen bereits abgeschlossenen und noch offenen Opportunities. Bei bereits abgeschlossenen wird nochmals unterschieden, ob sie in dem aktuellen Quartal (Actual Order Entry) oder in den vorigen drei Quartalen (Past Order Entry) abgeschlossen wurden. Dabei werden nur solche Opportunities berücksichtigt, die positiv abgeschlossen wurden. Zu finden sind diese in der unteren Zeile der Sales Pipeline (siehe Abbildung 5.3). Die noch offenen Opportunities werden unterteilt, in Opportunities, die noch in diesem Quartal fällig werden (Actual Opportunities), und Opportunities, die darüber hinaus fällig werden (Future Opportunities). Diese werden in der ersten Zeile der Sales Pipeline angezeigt (siehe Abbildung 5.1). Der Benutzer gelangt mit einem Klick auf die Summe zu einer Listenansicht der jeweiligen Opportunities.

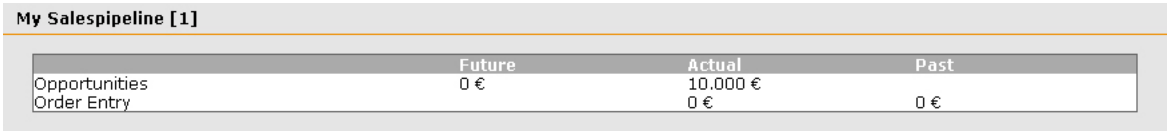

Abbildung 5.3: Sales Pipeline

Des Weiteren wird die Anzahl der offenen Aktivitäten, für die der angemeldete Benutzer zuständig ist, untergliedert in Aktivitäten, die innerhalb von 7 Tagen fällig sind (Actual), offene Aktivitäten, die bereits fällig waren (Past) und Aktivitäten, die in mehr als 7 Tagen fällig werden (Future), angezeigt. Der Benutzer erhält mit Hilfe dieser Informationen einen Überblick über die zu erledigenden Aufgaben. Mit einem Klick auf die Zahl erhält er eine Listenansicht der jeweiligen Aktivitäten.
Außerdem werden die nächsten Geburtstage der Kontaktpersonen, für die der angemeldete Benutzer verantwortlich ist, angezeigt. Der Benutzer hat die Möglichkeit, durch Klicken auf den Namen der Person, zur Detailseite zu gelangen und sämtliche Kontaktdaten einzusehen.

5.3.3 Anbindung der Datenbank

Die Anbindung der Datenbank an den entwickelten Prototypen erfolgt mittels einer XML-Datei. In dieser Datei werden die Parameter, die für die Verbindung zu der Datenbank nötig sind, definiert, sowie ein Name festgelegt, über den die Datenbank referenziert werden kann. Das Listing 5.1 zeigt die verwendete Anbindung einer MySQL Datenbank an den entwickelten Prototypen.

```
1  <?xml version="1.0" encoding="UTF-8"?>
2
3  <datasources>
4
5    <!-- MySQL DataSource -->
6    <local-tx-datasource>
7      <jndi-name>iSellEJBDS</jndi-name>
8
9      <connection-url>jdbc:mysql://127.0.0.1:3306/isell_ba</connection-url>
10
11     <driver-class>com.mysql.jdbc.Driver</driver-class>
12
13     <user-name>isell</user-name>
14     <password>novomind</password>
15
16     <exception-sorter-class-name>
17       org.jboss.resource.adapter.jdbc.vendor.MySQLExceptionSorter
18     </exception-sorter-class-name>
19
20     <metadata>
21       <type-mapping>mySQL</type-mapping>
22     </metadata>
23   </local-tx-datasource>
24
25 </datasources>
```

Listing 5.1: Anbindung der verwendeten MySQL Datenbank

Über die Konfigurationsdatei *persistence.xml* (siehe Listing 5.2) wird ein Persistence Context definiert und mit der zuvor definierten Datenbank verbunden (Listing 5.2, Zeile 7).

```
1  <?xml version="1.0" encoding="UTF-8"?>
2  <persistence version="1.0"
3    xmlns="http://java.sun.com/xml/ns/persistence"
4    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
5    xsi:schemaLocation="http://java.sun.com/xml/ns/persistence http://java.sun.com/xml/ns/
         persistence/persistence_1_0.xsd">
6    <persistence-unit name="iSellEJB">
7      <jta-data-source>java:/iSellEJBDS</jta-data-source>
8      <mapping-file>META-INF/orm.xml</mapping-file>
9      <class>
10     de.novomind.isell.entities.Account</class>
11
12 ...
13
14   </persistence-unit>
15 </persistence>
```

Listing 5.2: Auszug aus der persistence.xml Datei

Mit Hilfe des definierten Persistence Context wird über Annotations festgelegt, dass ein Entity in dieser Datenbank persistent gehalten werden soll (siehe Listing 5.3, Zeile 5).

```
1  @Stateless
2  @Local
3  public class AccountServiceBean implements AccountServiceLocal {
4
5    @PersistenceContext(unitName="iSellEJB")
6    protected EntityManager em;
7
8  ...
9  }
```

Listing 5.3: Verwendung des Persistence Context

5.4 Probleme und erforderliche Maßnahmen

Datenbankunabhängigkeit

Bei der Planung des Vertriebssystem wurde Datenbankunabhängigkeit angestrebt. Aus zeitlichen Gründen wurde dies bei der Entwicklung des Prototypen nicht weiter berücksichtigt. Dieser Aspekt soll nun näher untersucht werden. Dazu soll versucht werden, den entwickelten Prototypen mit einer anderen Datenbank, als der während der Entwicklung verwendeten MySQL Datenbank, zu betreiben. Hierzu wird die ebenfalls als Open Source verfügbare Datenbank PostgreSQL verwendet. Zunächst muss die bisherige Anbindung der MySQL Datenbank abgeändert werden. Das Listing 5.4 zeigt die Definition einer DataSource für eine PostgreSQL Datenbank. Als JNDI Name wird derselbe wie zuvor verwendet. Dadurch

sind für einen ersten Test keine Veränderungen am Code nötig, da die Datenbank nur über diesen Namen referenziert wird.

```
<?xml version="1.0" encoding="UTF-8"?>

<datasources>

  <!-- PostgreSQL DataSource -->
  <local-tx-datasource>
    <jndi-name>iSellEJBDS</jndi-name>

    <connection-url>jdbc:postgresql://localhost:5432/isell_ba</connection-url>

    <driver-class>org.postgresql.Driver</driver-class>

    <user-name>isell</user-name>
    <password>novomind</password>

  </local-tx-datasource>

</datasources>
```

Listing 5.4: Anbindung der PostgreSQL Datenbank

Der Start des JBoss Application Servers erfolgt noch erfolgreich, jedoch erhält man beim ersten Zugriff auf den Prototypen eine Exception. Dies deutet schon darauf, dass der entwickelte Prototyp nicht datenbankunabhängig ist. Nach genaueren Analysen wurde die Ursache hierfür gefunden. In den Abfragen, die zur Selektion von Entitäten aus der Datenbank genutzt werden, wurden teilweise MySQL spezifische Funktionen benutzt. Diese Abfragen sind in der EJB Query Language formuliert und beziehen sich nicht auf Datenbanktabellen sondern auf Objekte. Obwohl die Nutzung dieser Abfragesprache eine Datenbankunabhängigkeit suggeriert, muss trotzdem darauf geachtet werden, keine Datenbank spezifischen Funktionen zu nutzen.
In dem entwickelten Prototypen kann dieses Problem behoben werden, indem alle Abfragen, die MySQL spezifische Funktionen nutzen, welche insbesondere Datums- und Zeitfunktionen sind, angepasst werden. Auf die Nutzung dieser Funktionen kann verzichtet werden, wenn die Berechnungen eine Ebene höher stattfinden, also zuvor im Java Code ausgeführt werden. Nach diesen Anpassungen sollte die Datenbankunabhängigkeit vollständig erreicht sein.

Performanz

Die Problematik der Performanz wurde zu Beginn der Entwicklung nicht ausführlich genug beachtet und erst in einer späteren Phase der Entwicklung bemängelt. Bei einer Fortführung des Projektes besteht in dieser Hinsicht noch Handlungsbedarf. Das System muss einem

ausführlichen Test unterzogen und problematische Bereiche identifiziert und verbessert werden. Zum einen kann sicherlich die Konfiguration des JBoss Application Servers optimitert werden, eine Analyse des entwickelten Codes sollte aber dennoch stattfinden. Weiteres Potential bietet die Umstellung auf eine schnellere Datenbank, oder die Verlagerung der Datenbank auf einen separaten Server.

Bedienoberfläche und Usability

Die Bedienoberfläche wurde während der Entwicklung des Prototypen immer in Rücksprache mit den zukünftigen Benutzern der novomind AG entwickelt. Es wurden zwar moderne Techniken wie CSS und eine Template-Engine eingesetzt, das volle Potential wurde aber noch nicht ausgenutzt. Für eine Veröffentlichung des Vertriebssystem muss eine neutrale Bedienoberfläche entwickelt werden. Außerdem steht die Entwicklung der Mehrsprachenfähigkeit noch aus. Sobald das Vertriebssystem in einem produktiven Stadium ist, empfiehlt es sich, die entwickelte Bedienoberfläche einem ausführlichen Usability Test zu unterziehen und in einem Vergleichstest mit etablierten Systemen Schwachstellen aufzudecken.

Funktionsüberprüfung

Um die Entwicklung zu einem marktreifen Produkt fortzuführen, ist eine ausführliche Funktionsüberprüfung nötig. Diese sollte zudem automatisiert ablaufen, um genauere Ergebnisse und eine schnellere Rückmeldung zu erlangen. Dazu greift man bei der Java Entwicklung im Allgemeinen zu Unit Tests mit Hilfe von JUnit (JUnit, 2008). Da diese Tests allerdings nur bedingt für den entwickelten Prototypen geeignet sind und z. B. nicht die Bedienoberfläche testen können, muss eine andere Lösung gefunden werden. Es soll hier vor allem das Framework JSFUnit (JBoss.org, 2008) erwähnt werden, da es ein JBoss Projekt ist und somit perfekt auf den eingesetzten Application Server zugeschnitten ist. Aus Zeitgründen konnte dieses Framework nicht mehr eingesetzt werden.
Der entwickelte Prototyp konnte nur mittels definierten Testfällen und der Überprüfung auf erwartete Reaktionen getestet werden. Da die zukünftigen Benutzer des System sehr schnell in die Entwicklungsphase einbezogen wurden, war ein Feedback sehr schnell verfügbar und Fehler wurden zügig aufgedeckt. Die Tatsache, dass die Daten des Altssystems in dem entwickelten Prototypen schon frühzeitig zugänglich waren, war hierbei sehr hilfreich, da die zukünftigen Benutzer sich so besser im System orientieren konnten. Sie mussten sich nicht auf Testdaten einstellen, sondern kannten die Daten und konnten sich so auf das Testen der Funktionalität konzentrieren.

6 Methodische Abstraktion

Das in dieser Arbeit ausgearbeitete Konzept eines Vertriebssystems beschreibt nur in gewissen Teilen eine Standardlösung und ist eher als branchenspezifische Software einzuordnen. Es bietet nur Funktionen eines operativen CRM und ist dabei auch in diesen Funktionen eingeschränkt. So bietet es z. B. nicht die in CRM-Systemen sonst übliche Verwaltung von Leads, oder das Kampagnenmanagement. Das Vertriebssystem kann durchaus um diese Funktionalität erweitert werden, dies gehörte jedoch nicht zur Planung. Das System kann daher als branchenspezifische Software in die Branche der Softwareanbieter eingeordnet werden, wobei es aber z. B. durch die Möglichkeit benutzerspezifische Felder anzulegen, angepasst und so durchaus in anderen, verwandten, Branchen eingesetzt werden kann.
Es ist jedoch in jedem Fall als System für die Business-to-Business (B2B)-Kommunikation ausgelegt, da es die erforderliche Funktionalität für die Business-to-Consumer (B2C)-Kommunikation nicht implementiert. So ist das Vertriebssystem in den analytischen Funktionen nicht stark genug ausgeprägt.

Das System ist für den Einsatz in kleinen und mittleren Unternehmen (KMU) mit durchschnittlich 50 Beschäftigten ausgelegt. Besonders geeignet ist das Vertriebssystem dabei für Unternehmen, die innerhalb ihrer Unternehmensgruppe ein oder mehrere Tochterunternehmen beinhalten. In solch einem Umfeld kann das System durch die Abbildung der Unternehmensstruktur helfen, vorhandene Insellösungen abzuschaffen und eine ganzheitliche Sicht auf den Kunden zu ermöglichen.

In dem aktuellen Entwicklungsstand kann eine Veröffentlichung des Systems als Open Source Software nicht erfolgen. Um dies zu ermöglichen, muss das System lokalspezifisch konfigurierbar gemacht werden. So muss es in erster Linie sprachlich anpassbar sein. Alle ausgegebenen Texte müssen mittels einer Sprachdatei austauschbar sein und ggf. durch das einsetzende Unternehmen selber definiert werden. Neben einer Änderung der Sprache müssen zusätzlich die lokalspezifischen Parameter, wie die Währung, das Datums- oder das Zeitformat änderbar sein. Darüber hinaus muss das Aussehen des Vertriebssystem änderbar sein. Das einsetzende Unternehmen muss die Möglichkeit haben, das Logo, das oben rechts angezeigt wird, auszutauschen und das System farblich, aber in Teilen auch in seiner Struktur, zu verändern. Die hierfür erforderlichen Technologien wurde bei der Umsetzung des Prototypen bereits verwendet.

7 Zusammenfassung

Abschließend soll an dieser Stelle eine Zusammenfassung der Arbeit beschrieben und ein Fazit gezogen werden. Zudem folgt ein Ausblick auf die Möglichkeiten zur Weiterentwicklung des Vertriebssystems.

7.1 Fazit

In dieser Arbeit sollte das Konzept für ein neues Vertriebssystem für die novomind AG erarbeitet werden und in einem explorativem Prototypen die Umsetzbarkeit nachgewiesen werden. In Analysen wurde das Altsystem untersucht und Anforderungen an ein neues Vertriebssystem zusammengetragen. Die Anforderungen wurden mit bereits am Markt verfügbaren Open Source Vertriebssystemen verglichen, um ein evt. einsetzbares oder erweiterbares System zu finden. Es konnte jedoch kein geeignetes System hierfür gefunden werden, weshalb eine Neuentwicklung angestrebt wurde. Die Alleinstellungsmerkmale, wie die Möglichkeit eine Unternehmensstruktur abzubilden und die Möglichkeit zur fein granularen Verteilung von Zugriffsrechten, machen das System also geeignet für eine Veröffentlichung als Open Source Software.

Der Schwerpunkt dieser Arbeit lag auf der Ausarbeitung einer Architektur und der Realisierung eines explorativen Prototypen eines Vertriebssystems, das zum Einsatz bei der novomind AG geeignet ist. So wurde in dieser Arbeit eine Architektur entworfen, die auf modernen Technologien basiert und so eine hohe Wartbarkeit aufweist. Sie ist zudem leicht um weitere Komponenten oder um Schnittstellen zu anderen Systemen erweiterbar.

Durch das Feedback der zukünftigen Benutzer des Vertriebssystems, das schon sehr früh innerhalb der Entwicklungsphase eingeholt wurde, ist die Akzeptanz gegenüber dem System sehr hoch. Die Bereitstellung der Daten des Altsystems war hierbei sehr nützlich und kann für jegliche Projekte dieser Art dringend empfohlen werden. Zudem wurde so ein Scheitern des Projektes durch das Verfehlen der Anforderungen ausgeschlossen.

Die Ziele dieser Arbeit in Hinblick auf die Ausarbeitung eines Konzepts für ein neues Vertriebssystem für die novomind AG und die Umsetzung eines Prototypen konnten also erfüllt

werden. Auch wenn die Realisierung des Prototypen aus zeitlichen Gründen zum Teil eingeschränkt werden musste, bietet er trotzdem eine sehr gute Basis für die weitere Entwicklung.

7.2 Ausblick

Der entwickelte Prototyp muss zunächst ingenieurmäßig weiterentwickelt werden, so dass er alle gestellten Anforderungen erfüllt. Dazu müssen u. a. alle Funktionen zur Datenmanipulation hinzugefügt werden. Bis zur Einführung des Vertriebssystem bei der novomind AG muss sichergestellt sein, dass es mindestens die Funktionalität bietet, um eine problemlose Arbeit damit zu ermöglichen. Des Weiteren sollten die im Kapitel 5.4 aufgeführten Maßnahmen durchgeführt werden.
Die Komponente „Reporting und Selektion“ wird von Fahim Aleaf in seiner Arbeit (Aleaf, 2008) bereits vorbereitet. Diese wird sich in die vorhandene Umgebung einfügen und den Benutzern die Möglichkeit bieten, aus der Fülle von Daten die wichtigen auszuwählen. Dies wird vor allem für das Marketing notwendig sein.

Darüber hinaus sind noch zahlreiche Weiterentwicklungsmöglichkeiten denkbar. So könnte das Vertriebssystem um Schnittstellen zu dem ERP-System oder dem E-Mail-Management System der novomind AG erweitert werden. Die Kundenkommunikation wäre so im Sinne eines kommunikativem CRM integriert und würde eine Steigerung der Effizienz bedeuten. Eingehende E-Mails würden sofort dem richtigen Kunden zugeordnet und die Speicherung einer Kommunikationshistorie wäre leicht möglich.

Denkbar wäre zudem eine Verknüpfung mit einer Telefonanlage (CTI) (siehe Wikipedia, 2008a). So könnte der angerufene Vertriebsmitarbeiter bereits auf seinem Telefon alle wichtigen Informationen zu dem anrufendem Kunden angezeigt bekommen, um sich auf das Telefonat vorzubereiten. Es wäre außerdem leichter möglich, ein Protokoll des Gesprächs im Vertriebssystem anzufertigen, da das System wüsste, mit welchem Kunden der Benutzer telefoniert und z. B. die Eingabemaske zur Erstellung einer Aktivität direkt öffnen könnte.

Diese Erweiterungen sind vor allem mit Blick auf die Veröffentlichung des Vertriebssystems als Open Source Software wünschenswert und würden den Erfolg des Systems durchaus steigern. Es kann zudem, bei einer erfolgreichen Veröffentlichung, damit gerechnet werden, dass zusätzlich eine Weiterentwicklung durch eine Entwicklergemeinde erfolgt, die wiederum der novomind AG zu Gute kommen kann. Eine Veröffentlichung des Vetriebssystems als Open Source Software sollte jedoch erst dann erfolgen, wenn genügend eigene Erfahrungen damit gesammelt sind und das System einen marktreifen Status erlangt hat.

A CD-ROM

Dieser Arbeit liegt eine CD-ROM mit folgendem Inhalt bei:

- `/arbeit`: beinhaltet die digitale Version dieser Arbeit.
  - `/arbeit/latex`: beinhaltet die LaTeX-Version dieser Arbeit.
  - `/arbeit/pdf`: beinhaltet diese Arbeit im PDF-Format.
- `/prototyp`: beinhaltet den entwickelten Prototypen.
  - `/prototyp/db`: beinhaltet die Skripte zum Erstellen der Datenbank.
  - `/prototyp/data`: beinhaltet Beispieldaten im CSV-Format, die mit Hilfe der eingebauten Import-Schnittstelle importiert werden können.
  - `/prototyp/ear`: beinhaltet den Prototypen als .ear Datei.
  - `/prototyp/eclipse`: beinhaltet den Prototypen in Form von Eclipse Projekten.
  - `/prototyp/jboss`: beinhaltet eine Kopie des JBoss Application Servers, in die der Prototyp bereits integriert ist.
- `/usability`: beinhaltet die Videos des Usability Tests

B Aufgaben für den Usability Test

Für den durchgeführten Usability Test mussten die Probanden die folgenden Aufgaben ausführen.

Aufgaben für die Usability-Untersuchung der Open Source CRM Systeme

Aufgabenbereich: Lead erstellen, Aktivitäten planen, Opportunity erstellen

1. **Einloggen in das System**

   Loggen Sie sich mit folgenden Daten in das System ein.

   Benutzername: tester
   Passwort: novomind

2. **Überprüfen der anfallenden Termine für heute**

   Überprüfen Sie die für heute anfallenden Termine und lassen Sie sich eine Übersicht für diese Woche anzeigen.

3. **Anlegen eines Lead**

   Sie haben auf einer Messe einen Interessenten kennen gelernt. Dessen Visitenkarte wollen Sie nun als neuen Lead, also als potentiellen Kunden, in das System eingeben.

   B-i-B GmbH
   Big in Business GmbH
   Münchener Straße 17
   89409 München

   Oliver Müller
   - Leiter Einkauf -
   Tel.: 089 / 123456 - 7

E-Mail: o.müller@b-i-b.de

4. **Telefonanruf mit „Oliver Müller“ planen**

   Um bei dem neuen Interessenten nachzuhaken, wollen Sie einen Telefonanruf für:
   Fr. 16.11.2007 13:45 Uhr
   planen. Thema soll das Interesse des neuen Kontakts an dem Produkt abc sein.

5. **Ausweichtermin für Telefonat**

   Sie haben „Oliver Müller“ leider nicht erreicht und planen deshalb einen Ausweichtermin für das Telefonat für:
   Mo. 19.11.2007 10:00 Uhr

6. **Lead in Account und Contact umwandeln**

   Der neue Kontakt hat reges Interesse an dem Produkt abc, deswegen wandeln Sie den Lead in einen richtigen Unternehmenskontakt mit Ansprechpartner um. Bei dem Telefongespräch haben Sie erfahren, dass das Unternehmen ein geschätztes E-Mail Volumen von 15.000 E-Mails pro Monat hat.

7. **Follow-Up Aktivität erstellen**

   Der Kunde hat darum gebeten ihm Informationsmaterial zu abc zukommen zu lassen. Sie legen dazu eine Aktivität im System an. Das Informationsmaterial soll bis spätestens 23.11.2007 beim Kunden sein.

8. **Neuen Kontakt hinzufügen**

   Bei dem Telefonat haben Sie außerdem erfahren, wer die verantwortliche Projektleiterin bei dem Kunden ist:

   Name: Frau Franziska Becker
   Tel.: 089 / 123456 - 8
   E-Mail: f.becker@b-i-b.de

   Sie wollen nun Frau Becker als Ansprechpartner in das System eingeben.

9. **Zweigstelle von „B-i-B“ eintragen**

   Als nächstes möchten Sie die Zweigstelle des Unternehmens im System aufnehmen.

   Erfassen der Zweigstelle: B-i-B Dresden
   Adresse: Dresdner Hauptstraße 44a
   01234 Dresden

10. **Neue Opportunity anlegen**

    Da der Kunde sehr großes Interesse am Kauf von abc hat, wollen sie eine Opportunity im System anlegen. Die Entscheidung sollte bis zum 25.01.2008 getroffen sein und der voraussichtliche Zahlungseingang wird 35.000 € betragen.

11. **Planen eines Meetings**

    Um dem Kunden das Produkt abc näher zu bringen wird ein Meeting vereinbart. Sie wollen dieses Meeting nun im System eingeben. Stattfinden soll es am 11.12.2007 12:00 Uhr. Teilnehmen werden:

    Oliver Müller, Franziska Becker (B-i-B)
    Nico Müller und Sie (novomind)

12. **Ändern der Phase einer Opportunity**

    Der Kunde hat sich für ihr Produkt entschieden. Sie ändern den Status der Opportunity auf „cloased won“.

13. **Ausloggen aus dem System**

    Loggen Sie sich bitte aus dem System aus.

C Vergleichsmatrix

Die Tabelle C.1 zeigt die Vergleichsmatrix, die auflistet, ob eine Anforderung in einem CRM System erfüllt ist (+), nicht erfüllt ist (-), nur teilweise erfüllt ist (o) oder ob es nicht ersichtlich war, ob diese erfüllt wird (?).

	SugarCRM	vTiger CRM	OpenTaps
Administration			
Mandantenfähigkeit	-	-	?
Businessprofil abbildbar	-	-	-
Userverwaltung	+	+	+
Rollenverwaltung	+	+	+
Benutzerdefinierte Felder	+	+	?
Datenexport	+	+	?
Datenimport	+	+	?
Accounts			
Accounts verwalten	+	+	+
Inhaberschaft änderbar	+	+	+
multiple Adressvergabe	o (nur 2)	o (nur 2)	+
Zweigstellen möglich	+	-	+
Übersicht sortierbar	+	+	-
gefilterte Liste direkt exportierbar	o (nur CSV)	o (nur XLS)	-
Contacts			
Contact verwalten	+	+	+
Inhaberschaft änderbar	+	+	+
multiple Adressvergabe	o (nur 2)	o (nur 2)	+
Contact umhängen	+	+	+
Übersicht sortierbar	+	+	-
gefilterte Liste direkt exportierbar	o (nur XLS)	-	-
vCard Export	+	-	-

Activities			
Activities verwalten	+	+	+
Inhaberschaft änderbar	+	+	+
Opportunities			
Opportunities verwalten	+	+	+
Inhaberschaft änderbar	+	+	-
Übersicht sortierbar	+	+	+
Ansicht in Sales Pipeline	+	+	-
Sonstiges			
Verlauf der aufgerufenen Entitäten	+	-	+
Mehrsprachenfähigkeit	+	+	+
Hilfefunktion	+	o (link zu Wiki)	o

Tabelle C.1: Vergleichsmatrix

Glossar

Account Ein im Vertriebssystem abgespeichertes Unternehmen.

Activity Eine im Vertriebssystem abgespeicherte Aktivität.

Business Unit Geschäftseinheit innerhalb eines Unternehmens.

Contact Eine Kontaktperson, die einem Account angehört.

CRM Customer Relationship Management

CSS Cascading Style Sheets

CSV Comma Separated Values

EJB Enterprise Java Beans

ERP Enterprise Resource Planning

Java EE Java Enterprise Edition

JDBC Java Database Connectivity

JMS Java Message Service

JNDI Java Naming and Directory Interface

JSF Java Server Faces

Lead Ein potentieller Geschäftskontakt.

MDB Message Driven Bean

Opportunity Eine im Vertriebssystem abgespeicherte Geschäftschance.

POJO Plain old Java Object

Literaturverzeichnis

[Aleaf 2008] ALEAF, Fahim: *Entwicklung eines Reportmoduls für Ad-hoc Abfragen als Komponente eines webbasierten Open Source Vertriebssystems*. Hamburg, Hochschule für Angewandte Wissenschaften Hamburg, Bachelorarbeit in Verbereitung, 2008

[Balzert 2000] BALZERT, Helmut: *Lehrbuch der Software-Technik*. Bd. 2. Software-Entwicklung. Heidelberg ; Berlin : Spektrum, Akad. Verlag, 2000. – ISBN 3-8274-0480-0

[Brendel 2003] BRENDEL, Michael: *CRM für den Mittelstand*. Gabler, 2003. – ISBN 3-409-21934-X

[Dangelmaier u. a. 2004] DANGELMAIER, Wilhelm ; UEBEL, Matthias F. ; HELMKE, Stefan: Grundrahmen des Customer Relationship Management-Ansatzes. In: UEBEL, Matthias F. (Hrsg.) ; HELMKE, Stefan (Hrsg.) ; DANGELMAIER, Wilhelm (Hrsg.): *Praxis der Customer Relationship Management*. 2. Aufl. Wiesbaden : Gabler, 2004, S. 4–16. – ISBN 3-409-21890-4

[Gawlik u. a. 2002] GAWLIK, Tom ; KELLNER, Joachim ; SEIFERT, Dirk: *Effitiente Kundenbindung mit CRM*. Galileo, 2002. – ISBN 3-89842-246-1

[hipergate 2007] HIPERGATE: *hipergate*. 2007. – URL `http://www.hipergate.org/`

[Hippner u. a. 2006a] HIPPNER, Hajo ; HOFFMANN, Onno ; RIMMELSPACHER, Udo ; WILDE, Klaus D.: IT-Unterstützung durch Customer Relationship Management-Systeme am Beispielt von mySAP CRM. In: HIPPNER, Hajo (Hrsg.) ; WILDE, Klaus D. (Hrsg.): *Grundlagen des CRM*. 2. Aufl. Wiesbaden : Gabler, 2006, S. 75–96. – ISBN 3-409-22518-8

[Hippner u. a. 2006b] HIPPNER, Hajo ; RENTZMANN, René ; WILDE, Klaus D.: Aufbau und Funktionalität von CRM-Systemen. In: HIPPNER, Hajo (Hrsg.) ; WILDE, Klaus D. (Hrsg.): *Grundlagen des CRM*. 2. Aufl. Wiesbaden : Gabler, 2006, S. 45–74. – ISBN 3-409-22518-8

[Hippner und Wilde 2003] HIPPNER, Hajo ; WILDE, Klaus D.: CRM - Ein Überblick. In: HELMKE, Stefan (Hrsg.) ; UEBEL, Matthias F. (Hrsg.) ; DANGELMAIER, Wilhelm (Hrsg.): *Effektives Customer Relationship Management*. 3. Aufl. Wiesbaden : Gabler, 2003, S. 3–38. – ISBN 3-409-31767-8

[Hippner und Wilde 2006] HIPPNER, Hajo (Hrsg.) ; WILDE, Klaus D. (Hrsg.): *Grundlagen des CRM*. Gabler, 2006. – ISBN 3-409-22518-8

[Ihns u. a. 2007] IHNS, Oliver ; HARBECK, Dierk ; HELDT, Stefan M. ; KOSCHEK, Holger: *EJB 3 professionell*. Heidelberg : dpunkt.verlag, 2007. – ISBN 978-3-89864-431-0

[JBoss.org 2008] JBOSS.ORG: *JSFUnit*. 2008. – URL `http://www.jboss.org/jsfunit/`

[JUnit 2008] JUNIT: *JUnit*. 2008. – URL `http://www.junit.org`

[Kracklauer u. a. 2004] KRACKLAUER, Alexander H. ; MILLS, D. Quinn ; SEIFERT, Dirk: Collaborative Customer Relationship Management (CCRM). In: KRACKLAUER, Alexander H. (Hrsg.) ; MILLS, D. Quinn (Hrsg.) ; SEIFERT, Dirk (Hrsg.): *Collaborative Customer Relationship Management*. Heidelberg : Springer, 2004, S. 25–56. – ISBN 3-540-00227-8

[Märsch 2006] MÄRSCH, Friedhelm: *EJBs und J2EE*. W3L, 2006. – ISBN 3-937137-10-6

[OpenbravoERP 2007] OPENBRAVOERP: *Openbravo ERP*. 2007. – URL `http://www.openbravo.com/`

[OpenCRX 2007] OPENCRX: *openCRX - Enterprise Open Source CRM*. 2007. – URL `http://www.opencrx.org/`

[OpenTaps 2007] OPENTAPS: *opentaps - Open Source ERP + CRM*. 2007. – URL `http://www.opentaps.org/`

[Red Hat Inc.] RED HAT INC.: *Jboss Application Server*. – URL `http://www.jboss.org/`

[SalesPortal 2007] SALESPORTAL: *SalesPortal*. 2007. – URL `http://www.2d-salesportal.com/`

[Schmer 2008] SCHMER, André: *Weiterentwicklung, Evaluation und Einführung eines prototypischen Open Source Vertriebssystems*. Hamburg, Hochschule für Angewandte Wissenschaften Hamburg, Bachelorarbeit in Verbereitung, 2008

[Sourceforge.net 2008] SOURCEFORGE.NET: *Sourceforge.net - CRM Projects*. 2008. – URL `http://sourceforge.net/softwaremap/trove_list.php?form_cat=579`. – Abruf: 03.05.2008

[SugarCRM 2007] SUGARCRM: *SugarCRM - Commercial Open Source CRM*. 2007. – URL `http://www.sugarcrm.com/`

[SugarCRM 2008] SUGARCRM: *SugarCRM - About Us*. 2008. – URL `http://www.sugarcrm.com/crm/about/about-sugarcrm.html`. – Abruf: 07.06.2008

[Sun Microsystems Inc. 2002] SUN MICROSYSTEMS INC.: *Core J2EE Patterns*. 2002. – URL http://java.sun.com/blueprints/corej2eepatterns/Patterns/. – Abruf: 26.05.2008

[Sun Microsystems Inc. 2008a] SUN MICROSYSTEMS INC.: *Java 2 Platform, Enterprise Edition (J2EE) Overview*. 2008. – URL http://java.sun.com/j2ee/overview.html. – Abruf: 26.05.2008

[Sun Microsystems Inc. 2008b] SUN MICROSYSTEMS INC.: *Java Message Service (JMS)*. 2008. – URL http://java.sun.com/products/jms/. – Abruf: 26.05.2008

[Uebel u. a. 2004] UEBEL, Matthias F. (Hrsg.) ; HELMKE, Stefan (Hrsg.) ; DANGELMAIER, Wilhelm (Hrsg.): *Praxis des Customer Relationship Management*. Gabler, 2004. – ISBN 3-409-21890-4

[vTiger 2007] VTIGER: *vTiger CRM*. 2007. – URL http://www.vtiger.de/

[Wikipedia 2008a] WIKIPEDIA: *Computer Telephony Integration*. 2008. – URL http://de.wikipedia.org/wiki/Computer_Telephony_Integration. – Abruf: 05.07.2008

[Wikipedia 2008b] WIKIPEDIA: *Java EE*. 2008. – URL http://de.wikipedia.org/wiki/JavaEE. – Abruf: 07.06.2008

[XRMS 2007] XRMS: *XRMS - Open Source CRM*. 2007. – URL http://xrms.sourceforge.net/

Versicherung über Selbstständigkeit

Hiermit versichere ich, dass ich die vorliegende Arbeit im Sinne der Prüfungsordnung nach §22(4) ohne fremde Hilfe selbstständig verfasst und nur die angegebenen Hilfsmittel benutzt habe.

Hamburg, 7. Juli 2008	
Ort, Datum	Unterschrift