Hochschule Heilbronn – Universität Heidelberg

Konzeption und Realisierung einer RANDI2 - Installationssoftware

Bachelorarbeit
Medizinische Informatik
-
Andreas Kudak

Referent: Prof. Dr. Martin Haag, Hochschule Heilbronn
Korreferent: Prof. Dr.-Ing. Dirk Heuzeroth, Hochschule Heilbronn

Danksagung

Ich möchte mich bei all denjenigen bedanken, die mich bei der Erstellung dieser Bachelor-Arbeit unterstützt haben.
Mein besonderer Dank gilt Prof. Dr. Martin Haag, der mich durch seine hilfreichen Anregungen unterstützt hat und Dipl.-Inform. Med. Daniel Schrimpf, der mir mit seinem technischem Know-How wertvolle Hinweise in der Phase der Implementierung gegeben hat.

Inhaltsverzeichnis

1 Einleitung

1.1 Gegenstand und Motivation

Die heutige Medizin entwickelt sich ständig weiter. Laufend entstehen neue medizinische Fragestellungen und es werden neue Therapien und Medikamente erforscht. Um deren Wirksamkeit zu belegen und Fragestellungen zu beantworten, ist es wichtig, klinische Studien und empirische Sozialforschungen durchzuführen.

Für diese Vorgehensweisen werden unter anderen auch Versuche am Menschen durchgeführt. Hierbei ist immer mit bekannten und unbekannten personenbezogenen Störgrößen zu rechnen. Um diese auszugleichen werden die Personen in Gruppen eingeteilt. Dieses Verfahren nennt man Randomisierung. Wichtig bei dem Verfahren ist, dass keine unausgewogenen Gruppen entstehen. Um dies zu erreichen, ist es notwendig, verschiedene Parameter zu betrachten und basierend auf den Anforderungen an diese Parameter eine Randomisierung zu implementieren. Diese kann zum Beispiel nach dem Münzwurf Prinzip oder einem definierten Schema, wie der Blockrandomisation, erfolgen.

Von Bedeutung ist zudem auch, dass die Aufteilung in die Gruppen nicht nur von den Patienten anerkannt wird, sondern auch von den Personen, die die Studie ausführen. [1]

Um diese Randomisierung zu vereinfachen, wurde in Heidelberg eine Software namens RANDI entwickelt. Später machte sich der Kurs Softwareentwicklung des Studiengangs Medizinische Informatik der Universität Heidelberg und Hochschule Heilbronn das Problem der Randominisierung zu Eigen. Im Rahmen eines Praktikums entstand schließlich die Open Source Lösung RANDI2[2]. Zurzeit wird die Entwicklung der Software von verschiedenen Personen vorangetrieben.

1.2 Problemstellung und Zielsetzung

Aktuell ist die Installation, sowie die Konfiguration von RANDI2, sehr umständlich. Parameter oder sogar ganze Konfigurationsdateien werden für die vorhandene Infrastruktur des Anwenders angepasst. Obwohl jede Installation mit Eingaben bezüglich der Institution, verantwortlichen Personen, Logos und vielen anderen Einstellungen parametriert wird, verfügt das System zurzeit über keinen Mechanismus, der den Benutzer bei der Konfiguration unterstützt. Dadurch kann die Installation und Konfiguration meistens nur von erfahrenen Anwendern durchgeführt werden.

Ziel dieser Bachelorarbeit ist es, ein Modul zu entwickeln, welches den Administrator dabei unterstützt, das System bei der Installation von RANDI2 optimal einzurichten.

2 Theoretische Grundlagen

Die Software RANDI2 ist eine Java Webanwendung, die zur Ausführung einen JEE[1] Application Server oder einen Servlet Container benötigt. Des Weiteren wird eine Datenbank benötigt, um die im laufenden Betrieb anfallenden Daten zu verwalten. Im Rahmen des Softwareentwicklungsprojektes von RANDI2 werden die Tools Git[2] und Apache Maven[3] verwendet. Im Folgenden wird zuerst die Architektur von RANDI2 beschrieben, anschließend werden die Funktionen sowie Vor- und Nachteile von Git und Apache Maven dargestellt. Um Grundkenntnisse für die Entwicklung eines Stand-Alone oder Web-Client Installer zu schaffen, wird zusätzlich auf die Java Virtual Machine[4] (JVM) und die Java Platform Enterprise Edition[5] (JEE) eingegangen. Zusätzlich wird in diesem Kapitel die Architektur des Model View Controller Prinzip erläutert, an dem sich die Architektur der Installationssoftware orientiert.

2.1 Analyse der Architektur von RANDI2

Die Architektur von RANDI2 ist nach einem Drei-Schichtenmodell aufgebaut. Die untere Schicht besteht aus einer Verbindung zur Datenbank. Es können alle Datenbanken verwendet werden, die von Hibernate[6] unterstützt werden.

Die mittlere Schicht bildet den Server und die obere Schicht den Client ab.
In der Server-Schicht befinden sich die „Domain Components“ und die „Application Logic“. Für das Domain Components wird Hibernate verwendet, welches für das OR Mapping verantwortlich ist. Dies wird benötigt, um Daten von der Datenbank in das Model und wieder zurück zu wandeln. Die Datenbankverbindung befindet sich in der unteren Schicht. Die Application Logic wird durch das Java Framework Spring[7] abgebildet, das die Entwicklung mit JavaEE vereinfacht. Zudem ist mit Spring die Dependency Injection (DI) möglich. Das heißt, dass die Objekte ihre Abhängigkeiten nicht selbst verwalten müssen, sondern dies von Spring übernommen wird.

Das User Interface wird durch ICEfaces[8] gebildet. ICEfaces ist ein auf JavaServer Faces[9] basierendes Framework. Das User Interface geht von der Server- in die Client-Schicht über. Mit Hilfe dieses Frameworks wird die Benutzeroberfläche dargestellt.

---

[1] Java Platform, Enterprise Edition: http://www.oracle.com/technetwork/java/javaee/overview/index.html

[2] Git: www.git-scm.com

[3] Apache Maven: www.maven.apache.org

[4] JVM: www.java.com

[5] JEE: www.oracle.com

[6] Hibernate: www.hibernate.org

[7] Spring: www.springsource.org

[8] ICEfaces: www.icefaces.org

[9] JSF: http://www.oracle.com/technetwork/java/javaee/javaserverfaces-139869.html

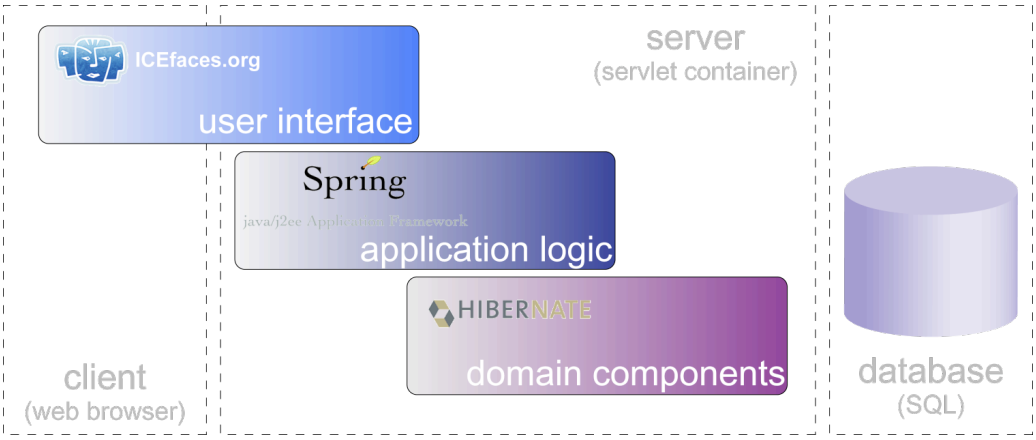

Abbildung 1: [3] Schichtenmodell zu RANDI2

2.2 Git

Git ist eine Open Source Software zur verteilten Versionsverwaltung von Dateien. Zu der Entstehung von Git kam es, als der Entwickler Linus Benedict Torvalds[10] im April 2005 eine Software benötigte, die Programmcode zwischen den Entwicklern des Linux Kernels[11] verwaltet. Die Anforderungen an diese Software waren die Unterstützung verteilter Arbeitsabläufe, eine hohe Sicherheit gegen unabsichtliche und böswillige Verfälschung und eine hohe Effizienz.[5]

Eine unabsichtliche oder böswillige Veränderung des Codes von RANDI2 ist nicht erwünscht. Diese Gefahr besteht jedoch, da es sich um eine Open Source Software handelt. Git bietet sich daher für die Entwicklung von RANDI2 an, da es unter anderem Schutz vor Verfälschung bietet und hervorragend zur Versionsverwaltung geeignet ist.

Git registriert Änderungen in einer Datei oder einem Verzeichnis, sobald diese erfolgen. Dies funktioniert über ein SHA-1[12] Verfahren. Ziel dieses Verfahren ist es, eine Nachricht in einen 40 Zeichen langen Hashwert so zu verschlüsseln, dass keine andere Nachricht mit diesem Hashwert identisch ist. Dadurch entsteht eine hohe Sicherheit gegen Verfälschung und zum anderen wird der Hashwert als eindeutige Referenzierung der Daten genutzt. Das SHA-1 Verfahren gilt zurzeit zwar noch als relativ sicher, es ist allerdings mit einer hohen Anzahl an Operationen und entsprechender Hardware möglich, solange eine neue Nachricht zu erstellen, bis sie denselben Hashwert hat, wie die Ursprungsnachricht. Das Verfahren kann also mit einem hohen Aufwand umgangen werden, für die Bedürfnisse von RANDI2 reicht es jedoch aus.[4]

Die hohe Effizienz von Git resultiert aus der Tatsache, dass das komplette Repository lokal auf den Clients gespeichert wird. Ein Vergleich zweier Versionen einer Datei ist so schnell möglich, weil zum einem ein Abgleich über den Hashwert erfolgt und zum anderen alle Versionen der Datei offline verfügbar sind. Dadurch entfällt die

[10] Linus Benedict Torvalds : * 28. Dezember 1969 in Helsinki, Finnland[4]

[11] Linux Kernel: www.kernel.org

[12] SHA-1: Secure Hash Algorithm

Zugriffszeit auf den Server. Das komplette Speichern des Repositories auf jedem Client bietet zudem Schutz vor Datenverlust. Wird der Server beschädigt, können die Dateien von den Clients wieder auf den Server übertragen werden.[4]

Ein weiterer großer Vorteil von Git ist das einfache Erstellen von Branches. Ein Branch ist eine Verzweigung, die von dem eigentlichen Repository abgeleitet wird. Möchte ein Entwickler z.B. eine Neuentwicklung unabhängig von der Hauptentwicklung erstellen, generiert er einen Branch, der eine 1:1 Kopie des Workspace[13] ist. Auf diesem Branch kann der Entwickler nun unabhängig von der Hauptentwicklung arbeiten.[4]

Nehmen wir an es liegt eine Version 1.1 vor, die bereits in Betrieb ist. Nun soll eine neue Version entwickelt werden, die eine große Änderung des Codes der Version 1.1 erfordert. Während der Entwicklungszeit wird nun ein Fehler in der Version 1.1 entdeckt, der schnell behoben werden muss. Die Entwickler können jetzt einfach auf den Code von Version 1.1 zugreifen, ohne die Änderungen des Codes für 1.2 rückgängig zu machen.[4]

All diese Punkte haben Git zu einer völlig neuen Versionsverwaltung gemacht. Git hat sich bewährt und wird heutzutage vielfach von großen, vor allem Open Source, Projekten genutzt. So gibt es auf Github[14] alleine zum Linux Kernel 2.6 über 450 Weiterentwicklungen von Open Source Entwicklern[6]. Das große Interesse an Git & Github verspricht auch in Zukunft Support und weitere Entwicklung.

2.3 GitHub

Damit der Entwickler seine Repositories über Git auf einen Server unterbringen und sie für andere zugänglich machen kann, braucht er einen Hosting-Dienst. Ein passender Hosting-Dienst für Software Entwicklungsprodukte, die Git verwenden, ist GitHub. Dieser Dienst arbeitet webbasiert und ist für Open Source Projekte kostenlos. Der Quellcode von RANDI2 befindet sich zurzeit auf GitHub. Der in dieser Arbeit entstehende Installer wird aus den zuvor erarbeiteten Gründen ebenfalls auf GitHub mittels Git verwaltet[15].

GitHub unterscheidet sich von anderen Hosting-Diensten, indem er in der URL nicht das Projekt an erster Stelle aufführt, sondern den aktuellen Benutzer. Dadurch entstand ein soziales Netzwerk, welches sich auch in dem Firmenslogan „SOCIAL CODING“ von GitHub widerspiegelt.

---

[13] Workspace: Arbeitsbereich auf den die Entwickler programmieren.

[14] Github: www.github.com

[15] Installationssoftware: git@github.com:akudak/RANDI2-Installer.git

2.4 Apache Maven

An der Entwicklung von RANDI2 sind laufend neue Programmierer beteiligt. Dies erfordert ein hohes Maß an Standards in der Entwicklung, welche mit der Unterstützung von Apache Maven durchgesetzt werden. Maven ist ein Projekt Management-Framework, das die Entwickler bei immer wieder kehrenden Aufgaben, wie zum Beispiel dem Kompilieren oder Testen von Quellcode, unterstützt.
Vor der Einführung von Maven mussten diese Prozesse oft vorher noch entwickelt werden. Die Entwicklung und Nutzung dieser Prozesse war sehr zeitaufwändig, da zum einen die Entwickler eingearbeitet werden mussten und zum anderen war es notwendig, die erstellten Prozesse zu pflegen. Zudem kommt hinzu, dass die Prozesse so speziell waren, dass sie in der Regel nicht für andere Projekte wiederverwendet werden konnten. Maven hingegen arbeitet nach dem Prinzip „Konvention vor Konfiguration“, das heißt je mehr Standards man einhält, desto geringer wird der Aufwand für die Konfiguration. Allgemein gültige Konventionen, die in einem Projekt befolgt werden, führen dazu, dass Aufgaben vielfach standardisiert werden können und nicht von der individuellen Umsetzung abhängen. So wird z.B. in einem Projekt eine gemeinsame Verzeichnisstruktur festgelegt. Mit Hilfe eines Plugins können nun alle Entwickler eine Aufgabe innerhalb dieser Verzeichnisstruktur erledigen. Gäbe es keine einheitliche Verzeichnisstruktur, wäre es nicht möglich, mit nur einem Plugin zu arbeiten. Jeder Entwickler würde sein eigenes Plugin benötigen. [7,8]

Zu den wichtigsten Aufgaben von Maven gehören das Kompilieren, das Packen von Jar-Dateien, aber auch das Testen und Erzeugen von Dokumentationen und Reports, die es ermöglichen, die Qualität der Software zu erhöhen. Maven verwendet ein lokales Repository, in dem alle Libraries abgelegt werden. Die Verknüpfungen speichert Maven in einer Baumstruktur. Wird eine Library hinzugefügt, die zusätzlich eine andere Library benötigt, werden diese automatisch von Maven heruntergeladen. Gleiches Prinzip gilt für das Löschen von nicht mehr benötigten Libraries. Diese Funktion sorgt dafür, dass ein Projekt nicht unnötig überladen wird. [7,8]

Als Fazit kann festgestellt werden, dass die Verwendung von Maven, speziell bei großen Softwareprojekten, Entwicklungszeit und damit Kosten einspart. Ein weitere Vorteil von Maven sind die Standards, die befolgt werden müssen. Sie sorgen für eine strukturierte Modellierung und helfen neuen Entwicklern beim Einstieg in das Projekt.

2.5 JVM - Java Virtual Machine

Viele Programmiersprachen, wie C, Cobol, C++, usw., arbeiten plattformabhängig. Soll eine Anwendung auf verschiedenen Plattformen mit unterschiedlichen Prozessoren und Betriebssystemen ausgeführt werden, so muss der Quellcode den Befehlssätzen des Prozessors angepasst werden. Diese Eigenschaft ist nachteilig, wenn ein Programm vielen Anwendern zur Verfügung gestellt werden soll.

Java-Entwicklungen haben dagegen den großen Vorteil, dass sie plattformunabhängig arbeiten („write once, run anywhere“[16]). Dies ist möglich, da der Java Compiler Maschinencode für einen allgemeinen Prozessor, der Java Virtual Machine (JVM) erzeugt. Diese JVM ist für die gängigsten Betriebssysteme verfügbar und interpretiert den Java-Bytecode in entsprechende Anweisungen für den eingesetzten Prozessor.

Technisch gesehen gehört die JVM zur Java Runtime Environment[17] (JRE), die für die Ausführung des Java-Bytecodes verantwortlich ist und zusätzlich noch Java-Klassenbibliotheken beinhaltet.

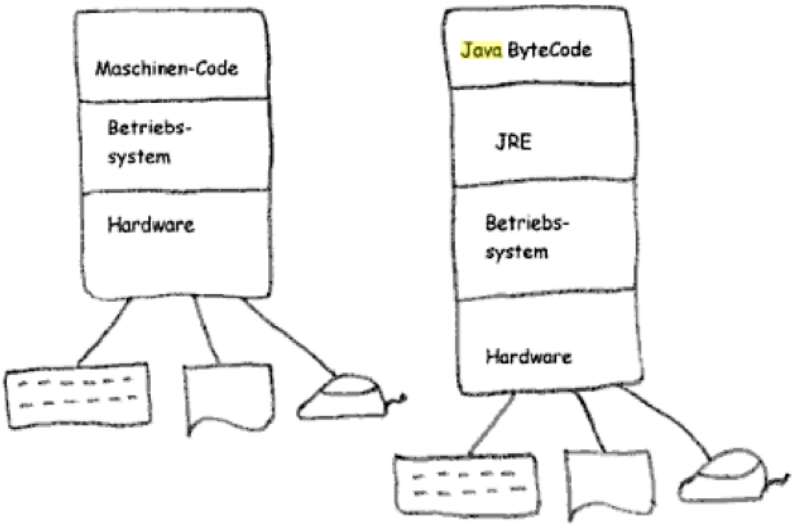

Abbildung 2: [9] Vergleich eines normalen Systems und eines JRE Systems

Da die JRE als Verbindungsstück zwischen Java Bytecode und Betriebssystem immer zusätzlich gestartet werden muss, haben Java-Entwicklungen durch eine längere Laufzeit einen Nachteil gegenüber plattformabhängigen Entwicklungen.
Dieser Nachteil ist aber im Fall der Installationssoftware für RANDI2 gegenüber der Plattformunabhängigkeit zu vernachlässigen.

[16] Übersetzung: einmal schreiben, überall ausführen

[17] JRE: www.java.com

2.6 JEE - Java Platform Enterprise Edition

Java Platform Enterprise Edition (JEE) ist die Spezifikation einer Architektur für in Java programmierte Anwendungen. JEE unterscheidet sich von der Java Standard Edition[18] (JSE), indem es noch weitere APIs[19] beinhaltet, die vor allem für Web-Anwendungen benötigt werden.

Zusätzlich benötigt jede JEE Anwendung zur Laufzeit einen JEE Application Server oder einen Servlet Container, die verschiedene Funktionalität wie Sicherheit, Kommunikation und Persistenzdienste anbieten. Server Tools für JEE Anwendungen werden von diversen Herstellern angeboten. Auch hier gibt es Open Source (z.B. Apache oder Sun) und kommerzielle (z.B. von IBM oder SAP) Lösungen.

Das Bereitstellen eines Servers ist sicherlich ein kleiner Nachteil gegenüber einer JVM Entwicklung, jedoch ist der Server bei Projekten, die für viele Anwender entwickelt werden, unabdinglich um Sicherheit, gutes Transaktionsmanagement und Kommunikation zwischen Komponenten sicher zu stellen.

2.7 Model View Controller

Eine bewährte Architektur in der Softwareentwicklung ist das Model-View-Controller Prinzip. Hierbei werden die drei Einheiten voneinander getrennt, was zum einen eine hohe Wiederverwendbarkeit bedeutet und zum anderen ein schnelles Austauschen der Komponenten ermöglicht. Der Wechsel von der geplanten Stand-Alone Anwendung zu einer eventuellen Web Anwendung kann z.B. ohne Änderung des Controllers erfolgen.

Das Model beschreibt das Datenmodell. In dieser Komponente werden die Daten gespeichert, die in der Präsentationsschicht angezeigt werden. Außerdem beinhaltet es Methoden, mit denen die Daten abgerufen und geändert werden.

Die Komponente View ist die Darstellungsschicht. Sie zeigt die im Model gespeicherten Daten an. Erfolgt eine Änderung dieser Daten, werden alle relevanten Elemente in der View aktualisiert. Die Komponente View nimmt zudem Benutzerinteraktionen, z.B. Eingaben in Textfeldern oder Aktionen der Buttons, entgegen. Diese Ereignisse werden jedoch nicht von der View selbst interpretiert, sondern an den Controller weitergegeben.

Der Controller erhält Interaktionen und Daten von der Komponente View. Die Aufgabe des Controllers ist es dann, die Methodenaufrufe im Model passend umzusetzen. Zusätzlich kann der Controller, je nach Aktion des Benutzers oder dem Programmzustand, den View anpassen.

---

[18] JSE: www.java.com

[19] API: Programmierschnittstelle

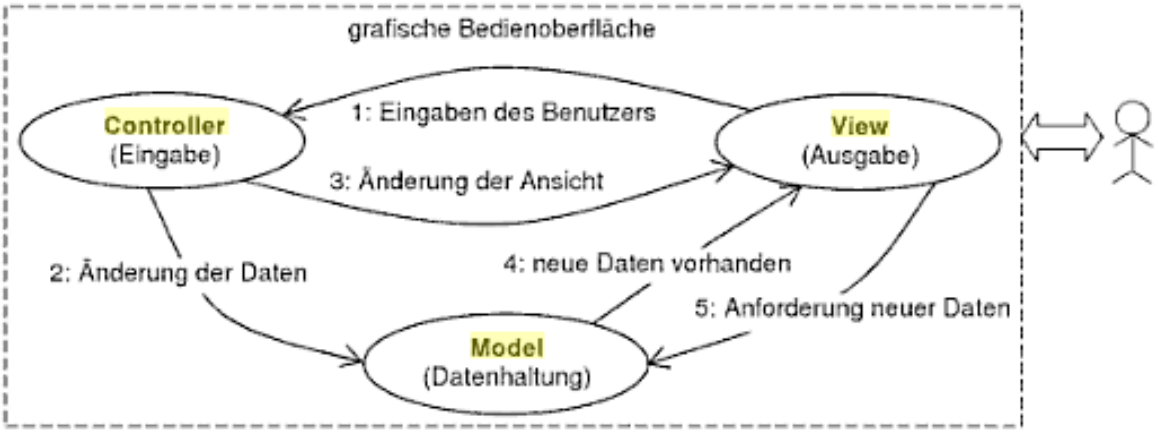

Abbildung 3: [10] MVC Prinzip

In Abbildung 3 ist das Zusammenspiel der einzelnen Komponenten in einer Model-View-Controller Architektur dargestellt. Der Anwender gibt nach Aufforderung des Programms seine persönlichen Daten in die Komponente View ein. Von dort aus werden die Daten weiter an den Controller geleitet (Schritt 1), der die Daten im Model ändert (Schritt 2). Hier werden sie in einem Objekt gespeichert. Der Controller spricht nun die Komponente View an, z.B. um anzuzeigen, dass die Daten erfolgreich gespeichert wurden (Schritt 3). Die neuen Daten werden dabei noch einmal angezeigt. Das Model gibt die Information, dass neue Daten vorhanden sind, an die Komponente View weiter (Schritt 4). Die Komponente View fordert darauf hin die neuen Daten an (Schritt 5). Anschließend werden diese angezeigt.

3 Anforderungsanalyse

3.1 Installer Ist - Analyse

Zurzeit ist eine Installation von RANDI2 sehr umständlich. Es muss ein Java Application Server oder ein Servlet Container installiert werden. Zudem wird ein Datenbankmanagementsystem (DBMS[20]) benötigt, das installiert und konfiguriert werden muss.
Für die Konfiguration müssen Benutzer mit bestimmten Rechten und Tabellen angelegt werden. In der bisherigen Installation wird ein Benutzerkonto mit den Rechten eines Administrators und einem Standardpasswort angelegt. Aktuell gibt es für den Administrator keinen Zwang dieses Passwort zu ändern, wodurch ein hohes Sicherheitsrisiko entsteht. Zusätzlich ist das Benutzerkonto mit fiktiven Daten bestückt. Auch hier wird der Benutzer nicht zum Ändern der Daten aufgefordert, was z.B. beim Versenden an die fiktive E-Mail Adresse zu Informationsverlust führen kann. Gleiches gilt für das Zentrum einschließlich des Passworts, mit dem sich neue Benutzer am Zentrum authentifizieren. Auch Informationen wie Ansprechpartner oder Logo der Firma müssen in einer Konfigurationsdatei im RANDI2 Verzeichnis durch den Administrator aufwendig von Hand geändert werden.

3.2 Installer - Sollzustand

Die Installationssoftware soll den Anwender dabei unterstützen, RANDI2 inklusive aller benötigten Programme zu installieren und zu konfigurieren. Bereits bei der Entwicklung der Software wurde besonderer Wert darauf gelegt, dass die Installation von RANDI2 einfach auszuführen ist und möglichst wenige Ansprüche an das System stellt.

Das nachfolgende Use Case Diagramm ‚RANDI2 Installer' enthält die Use Cases, die der Anwender auslöst, um die Installation und Konfiguration von RANDI2 durchzuführen. Der Use Case ‚Konfiguration' ist Teil des Use Cases Installation und kann nicht vom Anwender separat ausgelöst werden.

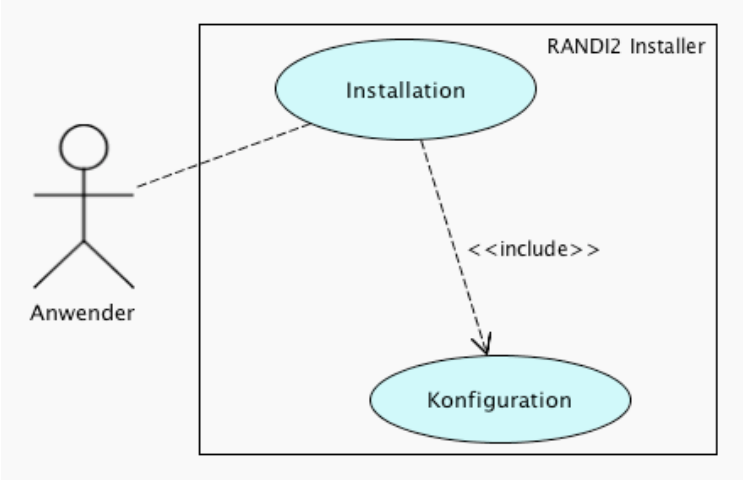

Abbildung 4: Use Case RANDI2 Installer

[20] Datenbankmanagmentsystem: Eine Verwaltungssoftware für die Datenbank

3.2.1 Use Case - Installation

Der Use Case ‚Installation' beschreibt die Installation aller benötigten Komponenten, die zum Betrieb von RANDI2 notwendig sind. Um die komplette Installation zu durchlaufen, muss der Rechner mit dem Internet verbunden sein. Dies ist notwendig, da während der Installation diverse Software-Programme herunter geladen werden müssen. Um die Installation durchzuführen, werden folgende Schritte ausgeführt.

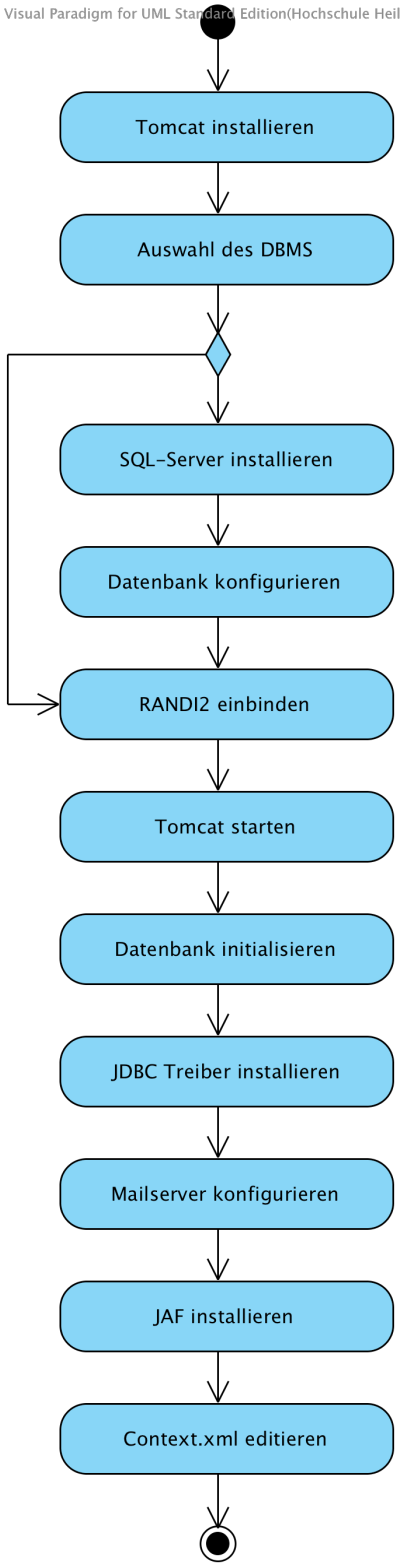

Abbildung 5: Aktivitäten - Installation

RANDI2 ist eine Webanwendung, die auf einem Server oder einem Servlet Container betrieben werden muss. Der Installer ist auf den JEE Application Server Tomcat ausgelegt. Dazu wir nach Programmstart die Installation von Tomcat vorgeschlagen. Sollte bereits eine Version von Tomcat installiert sein, kann der Benutzer auch nur den Pfad zum Tomcat Home-Verzeichnis angeben.

Im zweiten Schritt erfragt der Installer, ob bereits ein Datenbankmanagementsystem (DBMS) installiert ist. Sollte dies der Fall sein, fährt der Installer automatisch mit Schritt vier fort. Hat der Benutzer noch kein DBMS installiert, geht der Installer zu Schritt drei über.

Sollte der Anwender noch kein DBMS installiert haben, wird im dritten Schritt XAMPP herunter geladen und installiert. XAMPP liefert eine MySQL Datenbank, die für RANDI2 verwendet werden kann.

Im vierten Schritt wird die Datenbank, die die beim Betrieb von RANDI2 anfallenden Daten verwaltet, erstellt. Zudem wird ein Benutzer angelegt, der die Schreib- und Leserechte auf der Datenbank hat. Der Datenbankname und der Benutzername können frei gewählt werden.

Im fünften Schritt wird RANDI2 heruntergeladen und eingebunden.

Im sechsten Schritt wird Tomcat gestartet.

Die Initialisierung der Datenbank erfolgt im siebten Schritt über das ebenfalls heruntergeladene SQL-Skript.

Im achten Schritt wird der passende JDBC Treiber heruntergeladen und in Tomcat installiert. Dies ist notwendig, damit innerhalb der Programmiersprache Java eine Datenbankschnittstelle existiert. Der Installer entscheidet hier automatisch zwischen dem MySQL und PostgreSQL Treiber.

Im neunten und zehnten Schritt der Installation werden die Java Mail API und das JavaBeans Activation Framework[21] (JAF) heruntergeladen und ins Tomcat/lib Verzeichnis kopiert, damit im laufenden Betrieb von RANDI2 E-Mails versendet werden können.

Im elften und letzten Schritt schließt der Installer, mit der Bearbeitung der Datei Context.xml die Installation ab. Die Context.xml ist eine Tomcat Konfigurationsdatei, in der sich der JDBC Pfad und die Zugangsdaten zur Datenbank und zum Mail Server befinden.

Da die einzelnen Aktivitäten weitere Sub-Aktivitäten beinhalten, werden diese im Folgenden näher erläutert. Zusätzlich befinden sich im Anhang weitere Beschreibungen der Aktivitäten.

---

[21] JAF: http://www.oracle.com/technetwork/java/javase/downloads/index-135046.html

3.2.1.1 Aktivität - Tomcat installieren

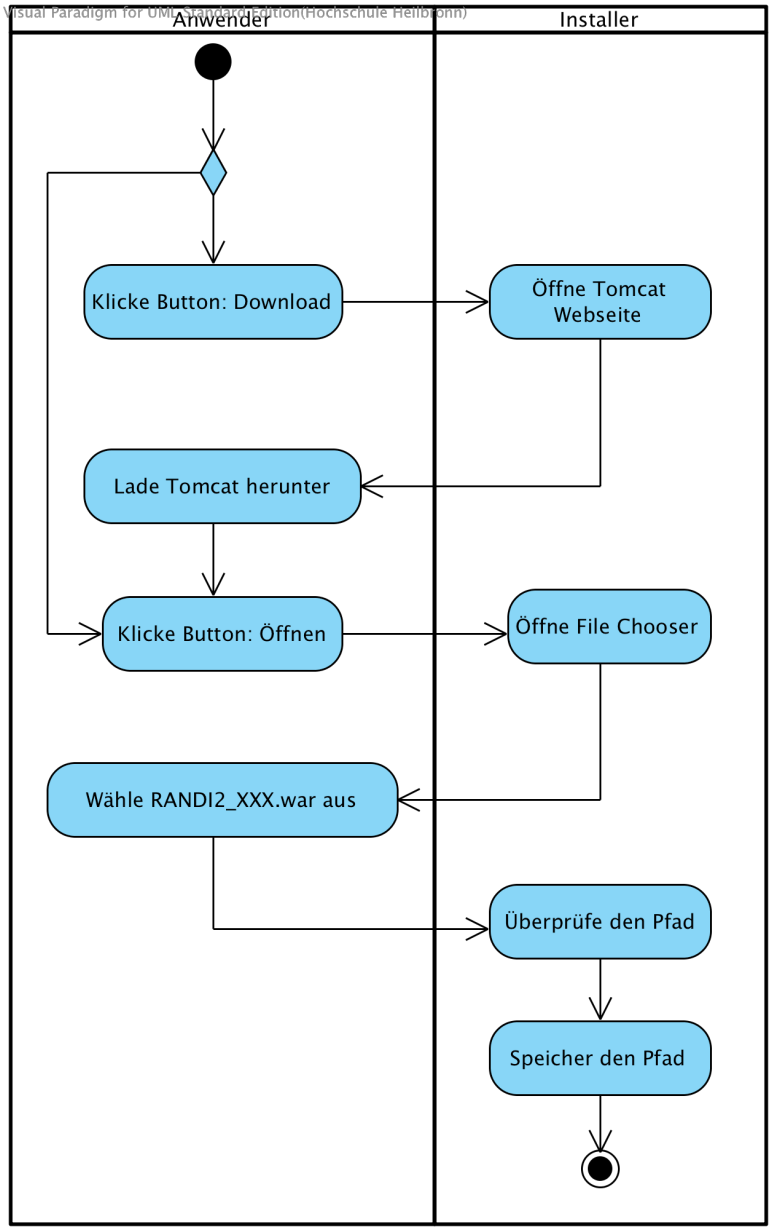

Abbildung 6: Aktivitätsdiagramm - Tomcat installieren

Der Installer bietet dem Anwender an, Tomcat 7 als JEE Application Server herunterzuladen und zu installieren. Optional kann der Anwender auch einen anderen Application Server oder einen Servlet Container benutzen. Allerdings muss er dann viele Schritte selber durchführen, weshalb dringend empfohlen wird, einen Tomcat Server zu verwenden. Falls der Anwender sich dafür entscheidet, Tomcat zu verwenden, öffnet der Installer, nach Drücken des Download-Buttons, einen Webbrowser mit der Webseite „http://tomcat.apache.org/“.
Der Anwender muss nun die Installationsdatei von Tomcat herunterladen und anschließend Tomcat installieren. Falls Tomcat bereits installiert ist, kann der Anwender den Download überspringen und direkt den Pfad zum Tomcat Home Verzeichnis angeben.

3.2.1.2 Aktivität - Auswahl des DBMS

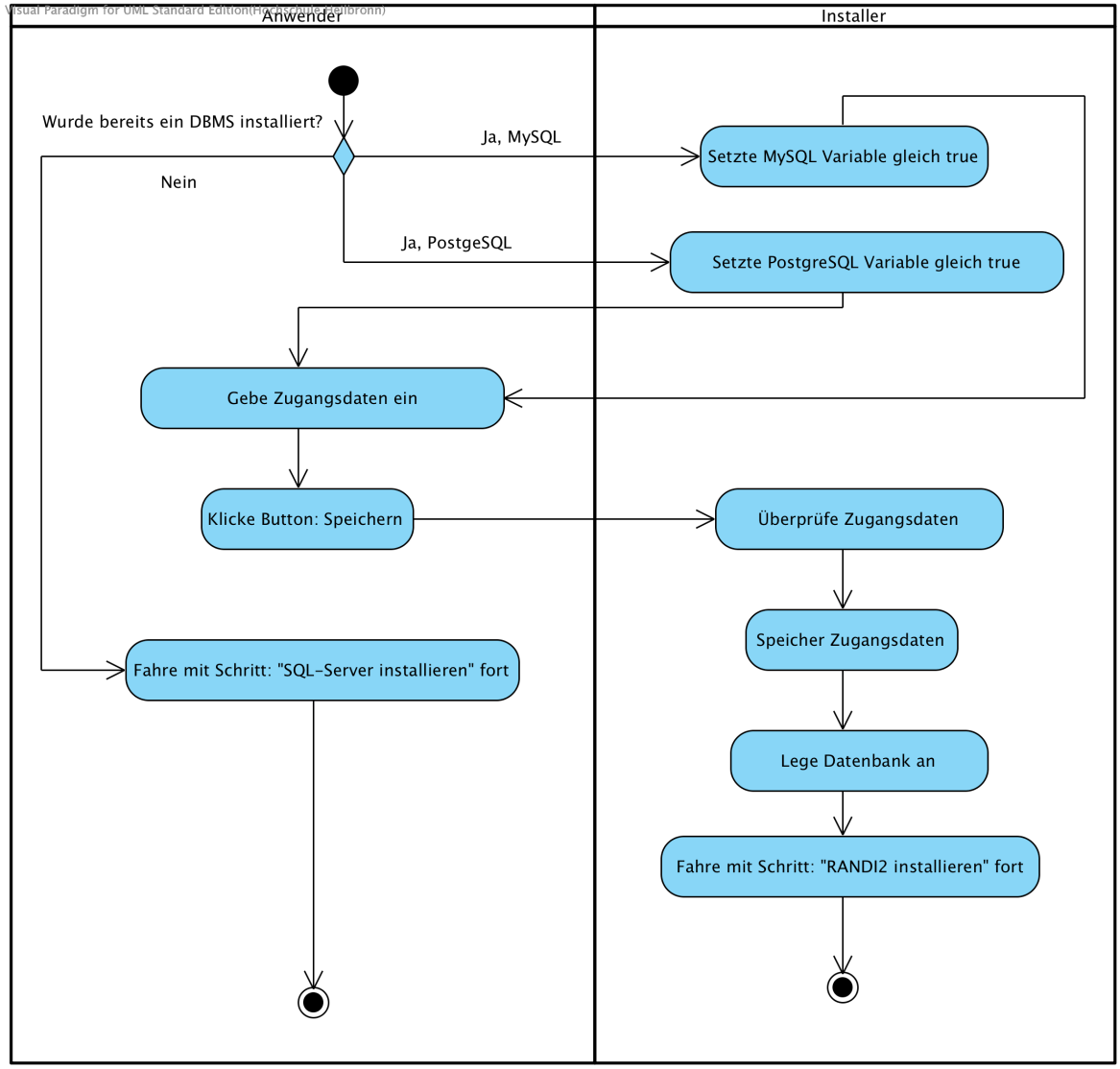

Abbildung 7: Aktivitätsdiagramm - Auswahl des DBMS

Der Anwender wird im zweiten Schritt gefragt, ob bereits ein SQL-Server installiert wurde und wenn ja, welcher Art. Er hat dabei die Auswahl zwischen MySQL und PostgreSQL. Allerdings muss die Sprache Hibernate unterstützen.
Sollte bereits ein SQL-Server installiert sein, wird der Benutzer aufgefordert, die Zugangsdaten inklusive des Datenbanknamens einzugeben. Ein weiterer Datenbankbenutzer wird hierbei nicht angelegt. Danach fährt der Installer automatisch mit dem Schritt RANDI2 installieren fort. Sollte noch kein SQL-Server installiert sein, so wählt der Anwender „Nein“ aus und der Installer fährt mit dem Schritt SQL-Server installieren fort.

3.2.1.3 Aktivität - SQL-Server installieren

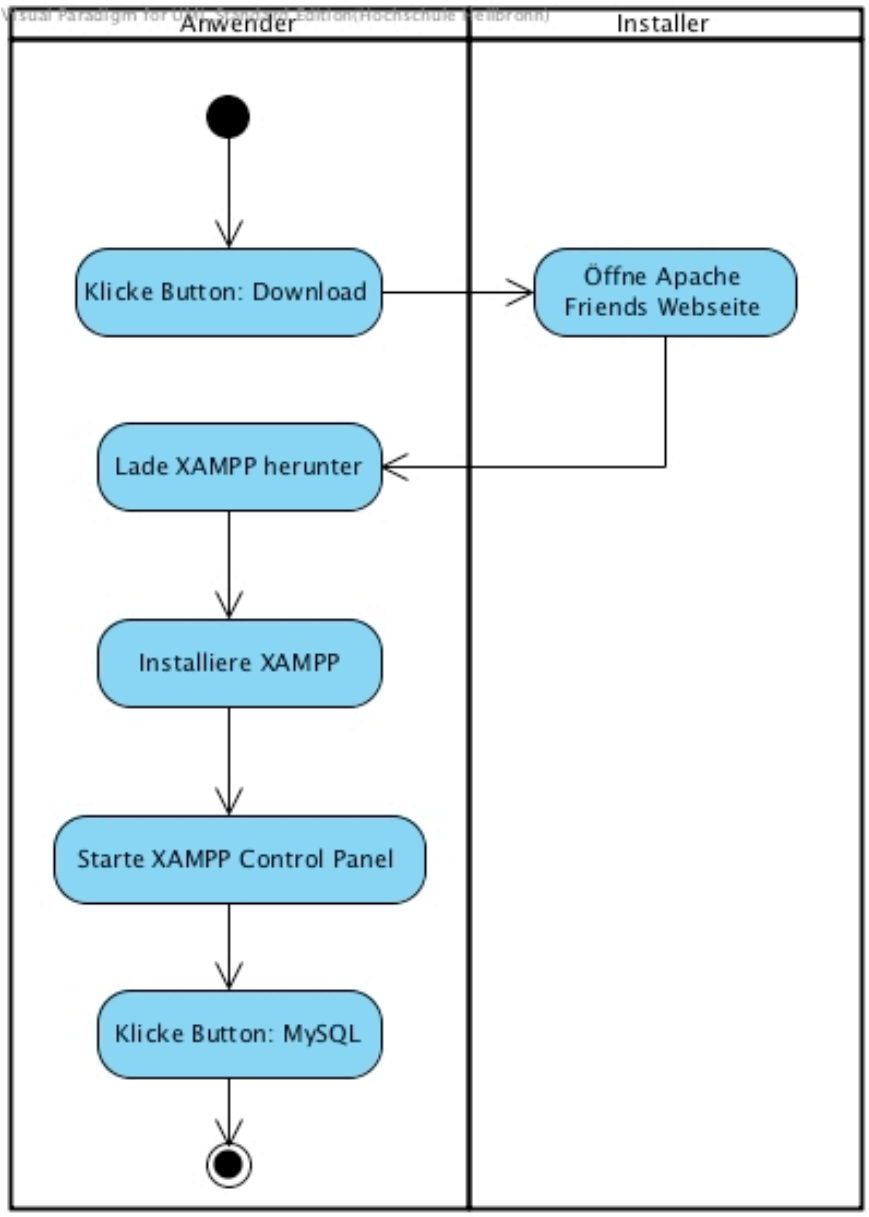

Abbildung 8: Aktivitätsdiagramm - SQL-Server installieren

Dieser Schritt wird nur ausgeführt, wenn der Anwender im vorherigen Schritt angegeben hat, dass noch kein SQL-Server installiert wurde.

Der Installer bietet den Anwender nun an, einen MySQL Server herunterzuladen. Über den Download-Button gelangt der Anwender zur Webseite „http://www.apachefriends.org/de/xampp.html“, wo er XAMPP herunterladen kann. Nach dem erfolgreichen Laden von XAMPP muss der Anwender die Installation ausführen und anschließend mit Hilfe des Control Panels MySQL starten.

3.2.1.4 Aktivität - Datenbank konfigurieren

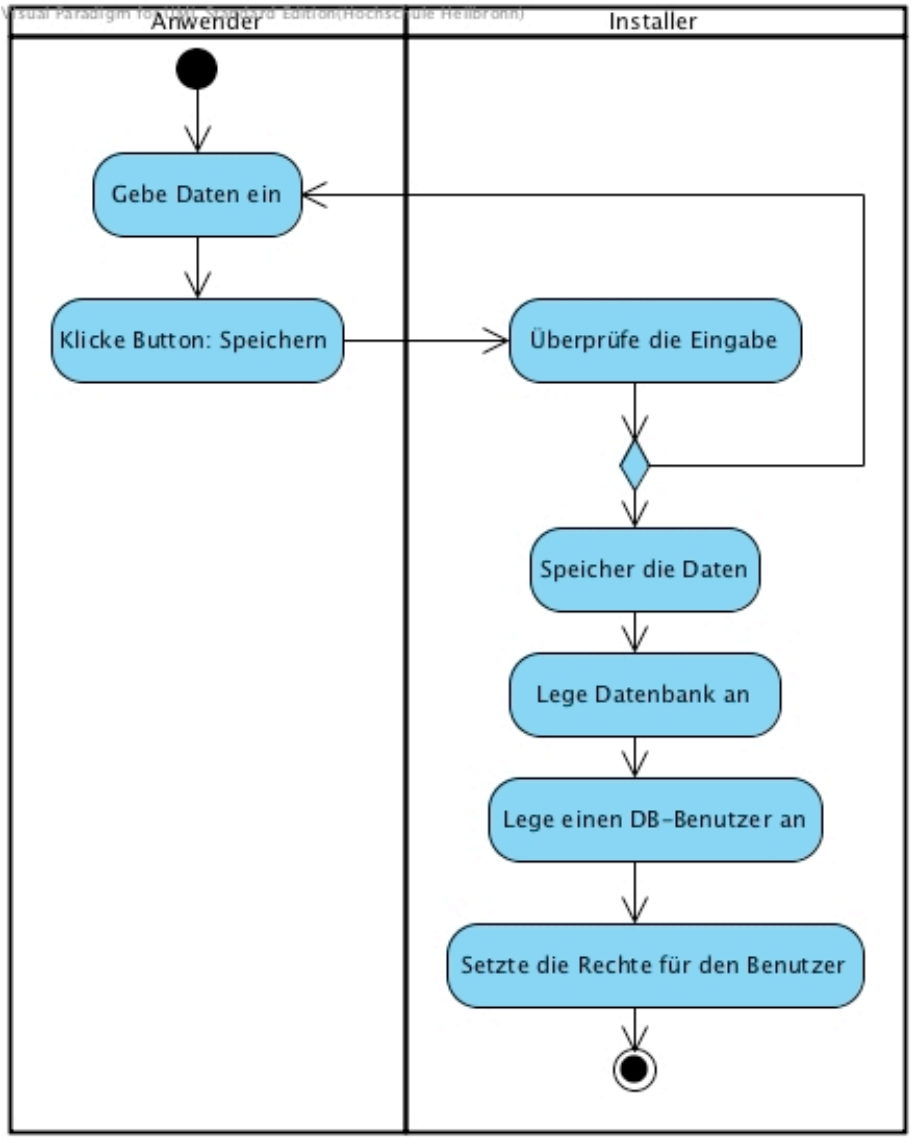

Abbildung 9: Aktivitätsdiagramm- Datenbank konfigurieren

Dieser Schritt wird nur ausgeführt, wenn der Anwender angegeben hat, dass noch kein SQL-Server installiert wurde.

Die Installationssoftware fordert den Anwender nun auf, das Login und Passwort für den Datenbank-Benutzer einzugeben. Zusätzlich werden die Angaben zu dem Server benötigt, auf dem das DBMS installiert ist. Befinden sich das DBMS und der Installer auf demselben System, lautet die IP-Adresse des Servers: 127.0.0.1. Diese wird auch standardmäßig vorgegeben.
Der Datenbankname kann ebenfalls frei gewählt werden. Zudem richtet der Installer die notwendigen Tabellen ein und fügt einen Benutzer, mit den Daten, die vom Anwender eingegeben wurden, hinzu.
Der nächste Schritt besteht aus dem Einbinden der RANDI2.war Datei. Es wird dabei nicht mehr unterschieden, ob er Anwender bereits einen SQL-Server installiert hat oder nicht.

3.2.1.5 Aktivität - RANDI2 einbinden

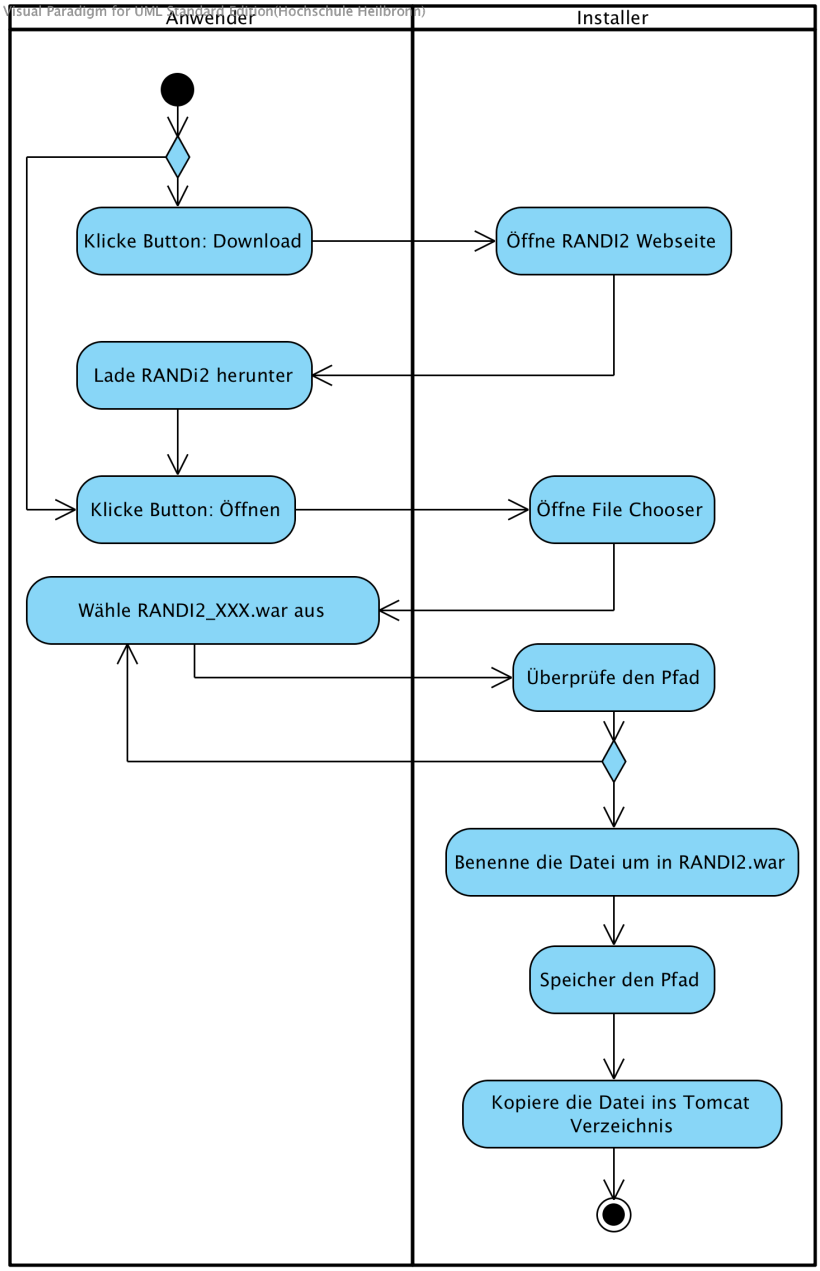

Abbildung 10: Aktivitätsdiagramm - RANDI2 installieren

Um RANDI2 einzubinden, bietet der Installer dem Anwender an, die Datei RANDI2 herunterzuladen. Dazu öffnet der Installer einen Webbrowser mit der Webseite „http://www.randi2.org“. Der Anwender muss die Datei selbstständig herunterladen und auf seinem Rechner speichern.
Sobald der Anwender den Öffnen-Button drückt, öffnet sich ein FileChooser. Über den FileChooser kann der Anwender den Pfad zur RANDI2_XXX.war Datei auswählen. Sollte der Anwender RANDI2 bereits zuvor heruntergeladen haben, kann er direkt den Pfad zur Datei auswählen.

Nachdem ein korrekter Pfad angegeben wurde, benennt der Installer die Datei RANDI2_XXX.war um und kopiert sie ins Tomcat/webapps Verzeichnis.

3.2.1.6 Aktivität - Tomcat starten

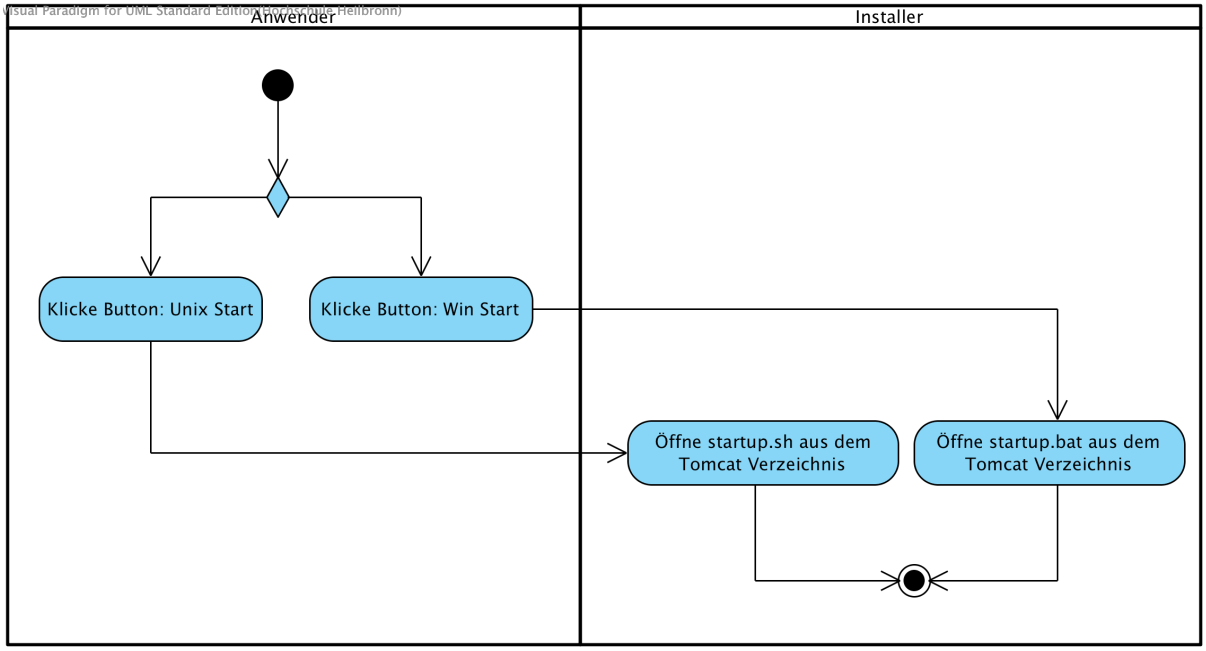

Abbildung 11: Aktivitätsdiagramm - Tomcat starten

In diesem Schritt wird die passende startup-Datei aus dem Tomcat/bin/ Verzeichnis geöffnet, welche den Tomcat Server startet. Dadurch wird die Datei RANDI2.war im Tomcat/webapps Verzeichnis entpackt.

3.2.1.7 Aktivität - Datenbank initialisieren

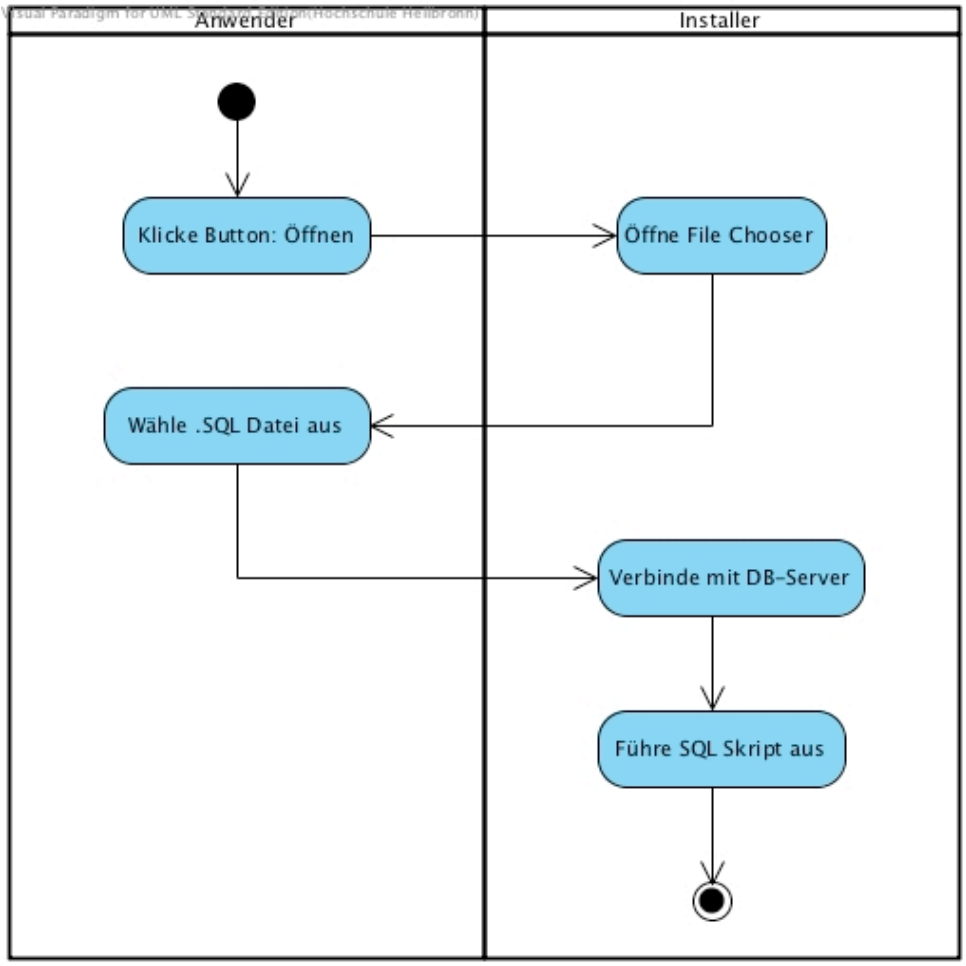

Abbildung 12: Aktivitätsdiagramm- Datenbank installieren

Im nächsten Schritt fragt der Installer den Anwender nach dem Ablageort der Datei randi2_XXX_InitData.sql oder psql_randi2_insert.sql. Welche Datei verwendet wird ist abhängig davon, ob eine MySQL oder PostgreSQL Datenbank installiert ist. Das MySQL-Skript wurde zuvor zusammen mit der RANDI2.war Datei heruntergeladen. Das PostgreSQL-Skript liegt dem Installer bei.

3.2.1.8 Aktivität - JDBC Treiber installieren

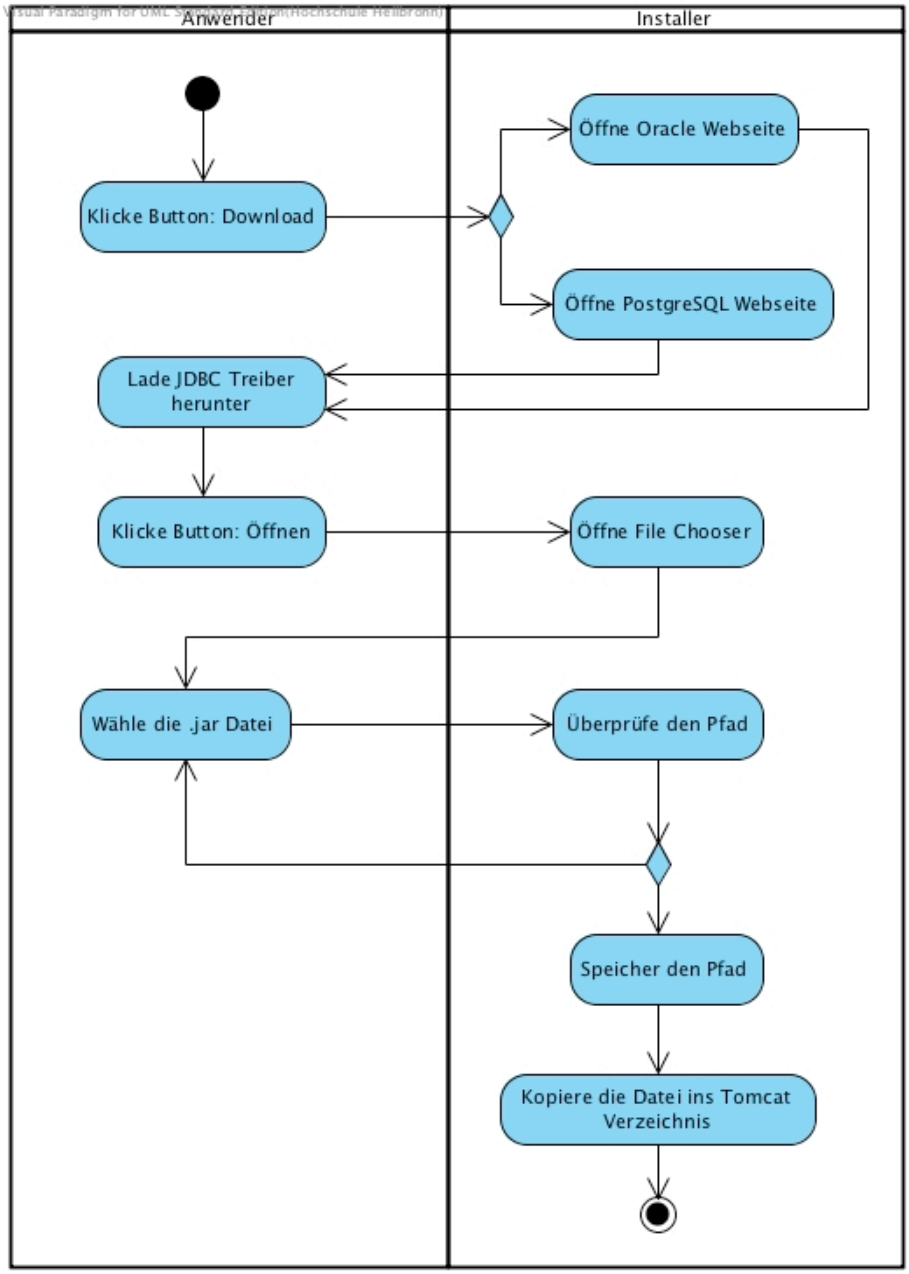

Abbildung 13: Aktivitätsdiagramm . JDBC Treiber installieren

In diesem Schritt installiert der Installer die JDBC Treiber. Dazu kann der Anwender die Datei nach Drücken des Download-Buttons von der im Webbrowser geöffneten Internetseite herunterladen. Der Installer wählt dabei automatisch die passenden Webseite (http://www.mysql.com/products/connector/“ für MySQL oder http://jdbc.postgresql.org/download.html für PostgreSQL).
Nach dem erfolgreichen Laden kann, durch das Drücken des Öffnen-Buttons, wieder über den FileChooser der Pfad ausgewählt werden. Auch hier besteht kein Zwang, die Datei erneut zu laden.

3.2.1.9 Aktivität - Mailserver konfigurieren

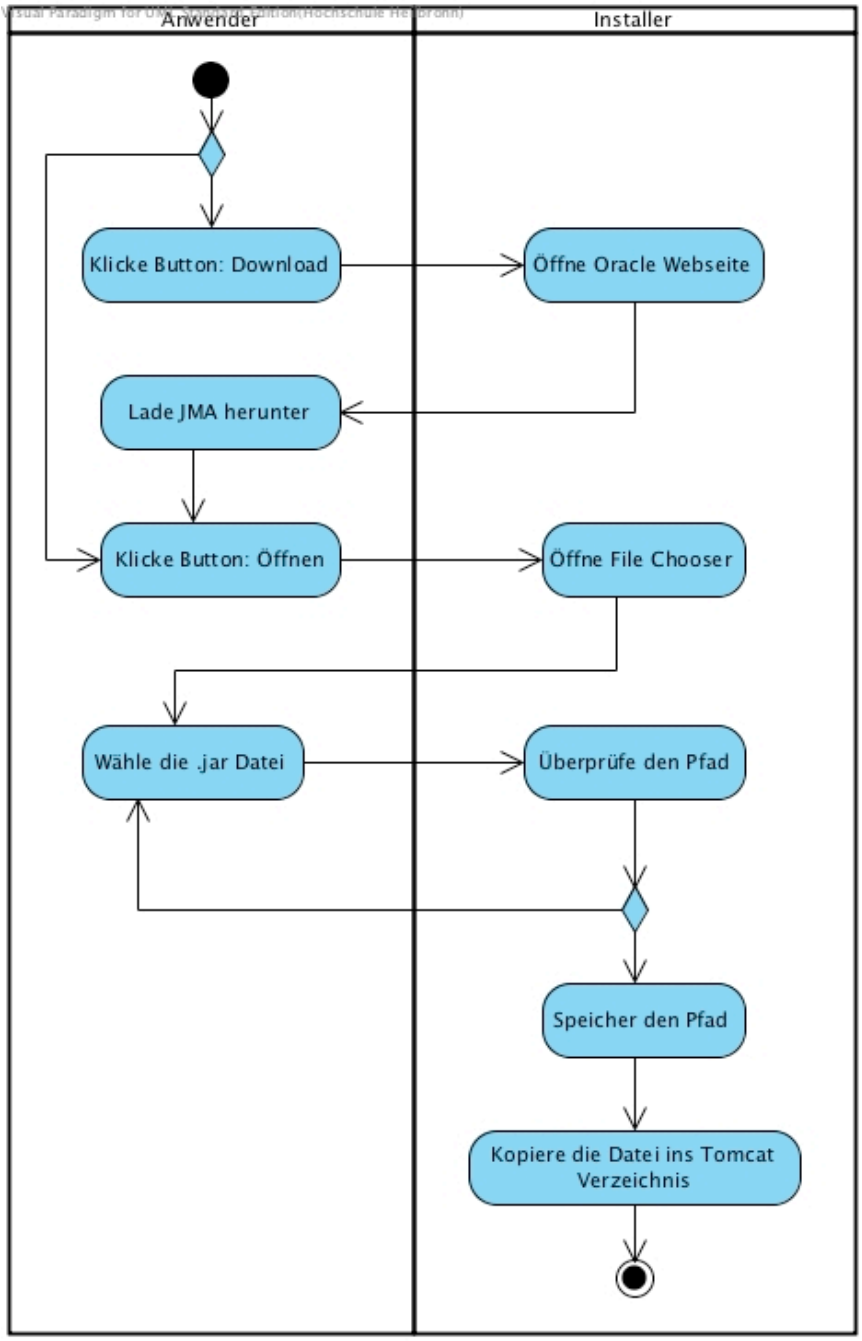

Abbildung 14: Aktivitätsdiagramm - Mailserver konfigurieren

Für die Installation der Java Mail API[22] (JMA) bietet der Installer dem Anwender an, die JMA herunterzuladen. Dazu öffnet der Installer einen Webbrowser mit der Webseite „http://www.oracle.com/technetwork/java/javaee/index-138643.html“. Der Anwender muss die Java Mail API herunterladen und auf seinem Rechner speichern. Anschließend gibt er den Pfad zur heruntergeladenen Datei an. Dies macht der Anwender, indem er den Button zum Öffnen drückt. Über den FileChooser kann er nun die Datei auswählen. Sollte der Anwender die JMA bereits zuvor geladen haben, muss er sie nicht erneut laden. Er kann direkt den Pfad zur Datei wählen.

[22] JMA: www.oracle.com

3.2.1.10 Aktivität - JavaBeans Activation Framework installieren

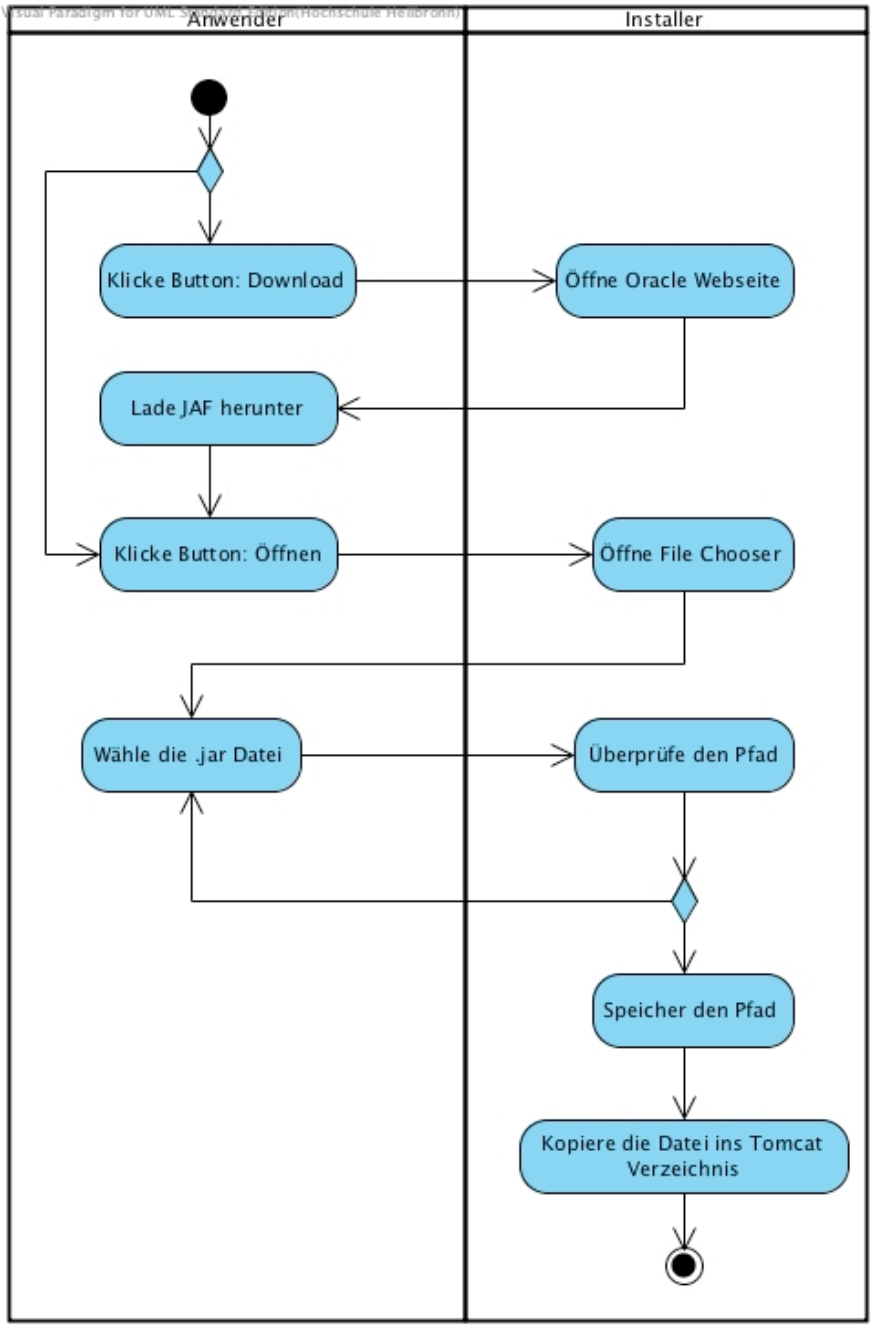

Abbildung 15: Aktivitätsdiagramm - JavaBeans Activation Framework installieren

Als nächstes bietet der Installer dem Anwender an, das JavaBeans Activation Framework (JAF) zu installieren. Dazu wird nach Drücken des Download-Buttons im Webbrowser die Internetseite „http://www.oracle.com/technetwork/java/javase/downloads/index-135046.html“ geöffnet.
Der Anwender muss hier die JAF Datei herunterladen und anschließend, wie auch in den vorherigen Schritten, den FileChooser öffnen, um den Pfad anzugeben.
Auch hier besteht kein Zwang, die Datei ein zweites Mal herunterzuladen.

3.2.1.11 Aktivität - Context.xml editieren

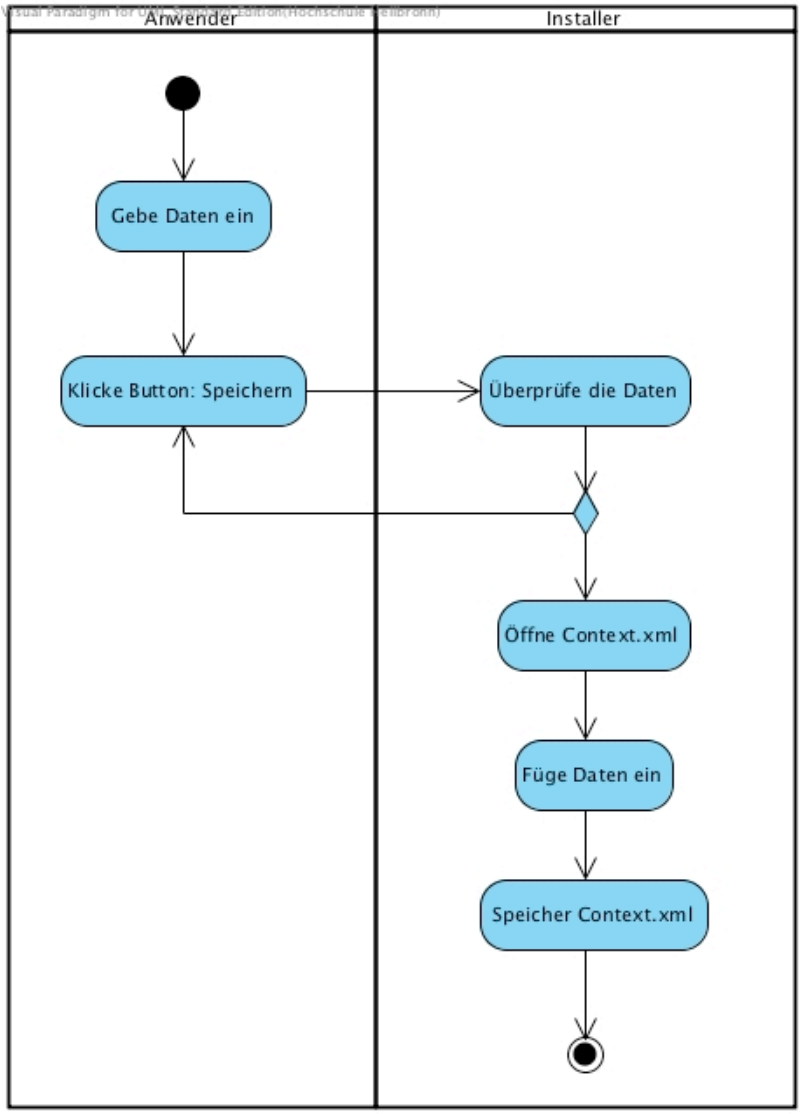

Abbildung 16: Aktivitätsdiagramm - Context.xml editieren

Nun wurden alle benötigten Programme und Frameworks installiert. Als letzten Schritt der Installation muss der Anwender die Zugangsdaten für den Mail-Server eingeben.

Nach Drücken des Speichern-Buttons öffnet der Installer die Datei context.xml, die sich im Installationsverzeichnis von Tomact befindet. Der Installer hinterlegt in dieser Datei folgende Informationen:

- Benutzername des Datenbank-Anwenders
- Datenbankname
- Zugangsdaten des Mail-Servers
- Ablageort des JDBC Treibers
- Ablageort der Java Mail API Datei
- Ablageort des JavaBeans Activation Frameworks

Die Installation von Tomcat, des DBMS mittels XAMPP, die JNDI Konfiguration, die JDBC Datenbanktreiber und RANDI2 sind hiermit beendet, sodass nun die Konfiguration über die Installationssoftware erfolgen kann.

3.2.2 Use Case - Konfiguration

Der Use Case ‚Konfiguration' dient dazu, die Installation von RANDI2 zu konfigurieren. Um diesen Use Case auszuführen, muss die Installation der benötigten Komponenten erfolgreich durchgeführt sein. Die einzelnen Aktivitäten werden auch hier detailliert in Aktivitätsdiagrammen beschrieben. Ebenfalls befinden sich im Anhang Beschreibungen zu den einzelnenAktivitäten.

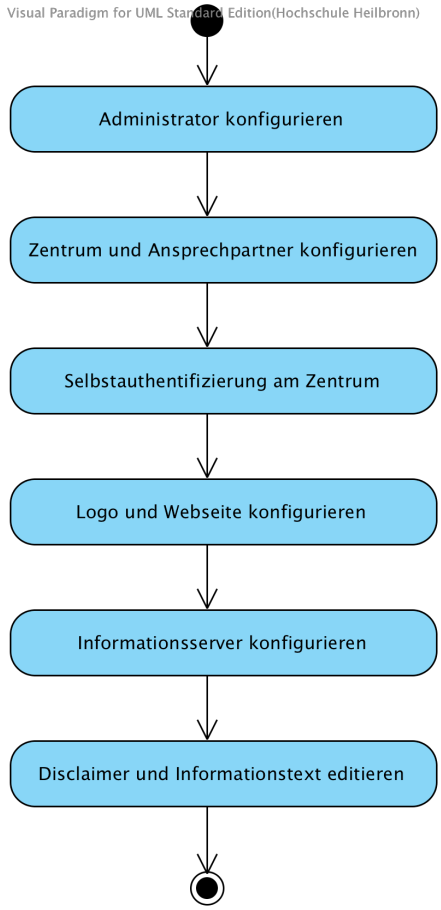

Abbildung 17: Aktivitäten - Konfigurieren

Im ersten Schritt der Konfiguration wird der Administrator konfiguriert.

Im zweiten Schritt werden Daten zum Zentrum und Daten eines Ansprechpartners abgefragt. Anschließend werden diese in der Datenbank gespeichert.

Im dritten Schritt kann der Anwender entscheiden, ob sich neue Benutzer selbst am Zentrum authentifizieren dürfen. Sollte dies der Fall sein, legt der Anwender ein Passwort an, mit dem sich die neuen Benutzer registrieren können.

Im vierten Schritt kann ein Logo ausgewählt und eine Webseite angegeben werden.

Im fünften Schritt werden Daten zum Informationsserver erfragt, wie z.B. die Absender E-Mail Adresse.

Im sechsten Schritt gibt der Benutzer, jeweils in deutscher und englischer Sprache, einen Informationstext und einen Disclaimer ein.

3.2.2.1 Aktivität - Administrator konfigurieren

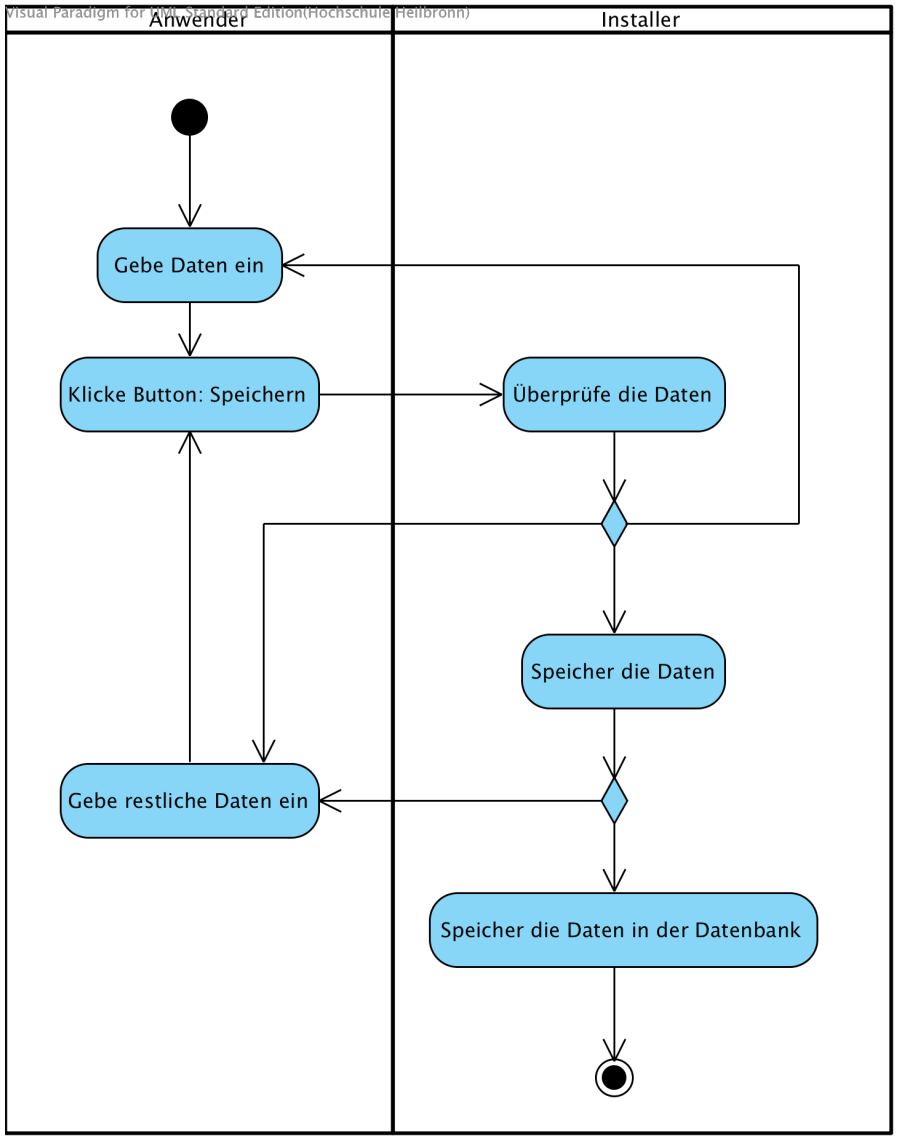

Abbildung 18: Aktivitätsdiagramm - Administrator konfigurieren

Als erster Schritt der Konfiguration erfolgt die Eingabe der Daten für das Benutzerkonto des Administrators. Die Eingabe findet über zwei Schritte statt.

Zuerst muss der Anwender den Akademischen Grad, den Vornamen, den Nachnamen, den Benutzernamen und das Geschlecht des Administrators angeben. Dabei sind alle Felder, bis auf der Akademischen Grad, Pflichtfelder.

Durch das Drücken des Speichern-Buttons werden diese Daten überprüft und bei erfolgreicher Eingabe in ein Objekt, jedoch nicht in der Datenbank, gespeichert.

Waren alle Eingaben gültig, kann der Anwender über den Weiter-Button zur nächsten Eingabemaske gelangen. Hier müssen als Pflichtfelder die E-Mail Adresse, Fax-, Mobil- und Telefonnummer, sowie ein Passwort eingegeben werden.
Drückt der Anwender, nach der Eingabe der Daten, den Speichern-Button, werden die Daten nach gelungener Überprüfung in die Datenbank eingetragen.

3.2.2.2 Aktivität - Zentrum und Ansprechpartner konfigurieren

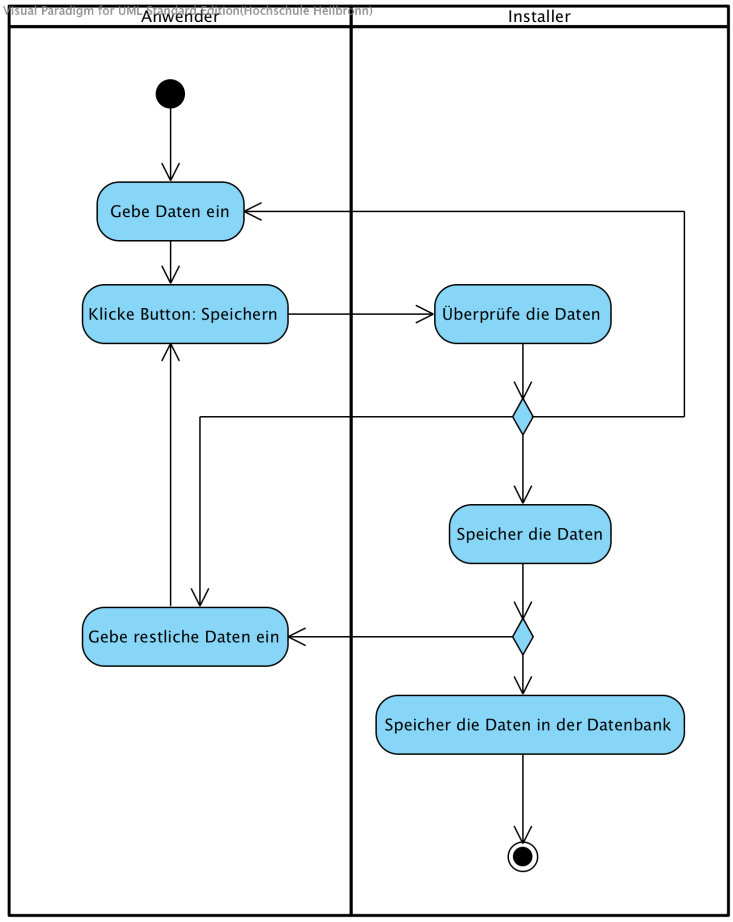

Abbildung 19: Aktivitätsdiagramm - Zentrum und Ansprechpartner konfigurieren

Nachdem der Administrator erfolgreich bearbeitet wurde, wird der Benutzer dazu aufgefordert, ein Zentrum und einen Ansprechpartner anzugeben. Das Prinzip und die Eingabemaske sind dabei der vorhergehenden Bearbeitung des Administrators angelehnt.

Im ersten Schritt wird der Anwender, bezüglich der Pflichtfelder Name, Land, Stadt, Postleitzahl und Straße des Zentrums, befragt. Durch das Drücken des Speichern-Buttons werden die Daten in ein Objekt, jedoch nicht in der Datenbank, gespeichert.

Mit Betätigen des Weiter-Buttons wird die Eingabemaske geöffnet, die den Benutzer auffordert, Angaben zum Ansprechpartner des Zentrums zu machen.

Gefordert sind Akademischer Grad, Vorname, Nachname, E-Mail Adresse und Geschlecht. Auch hier sind, wie beim Administrator, alle Felder außer dem Akademischen Grad Pflichtfelder.

Nach dem Drücken des Speichern-Buttons werden die Daten überprüft und der Weiter-Button wird durch das erfolgreiche Bearbeiten des Schrittes frei geschaltet.

Im letzten Schritt zum Ansprechpartner muss der Anwender eine Telefon-, Mobilfunk- und Faxnummer angeben. Es handelt sich bei allen Feldern um Pflichtangaben.

3.2.2.3 Aktivität - Selbstauthentifizierung am Zentrum

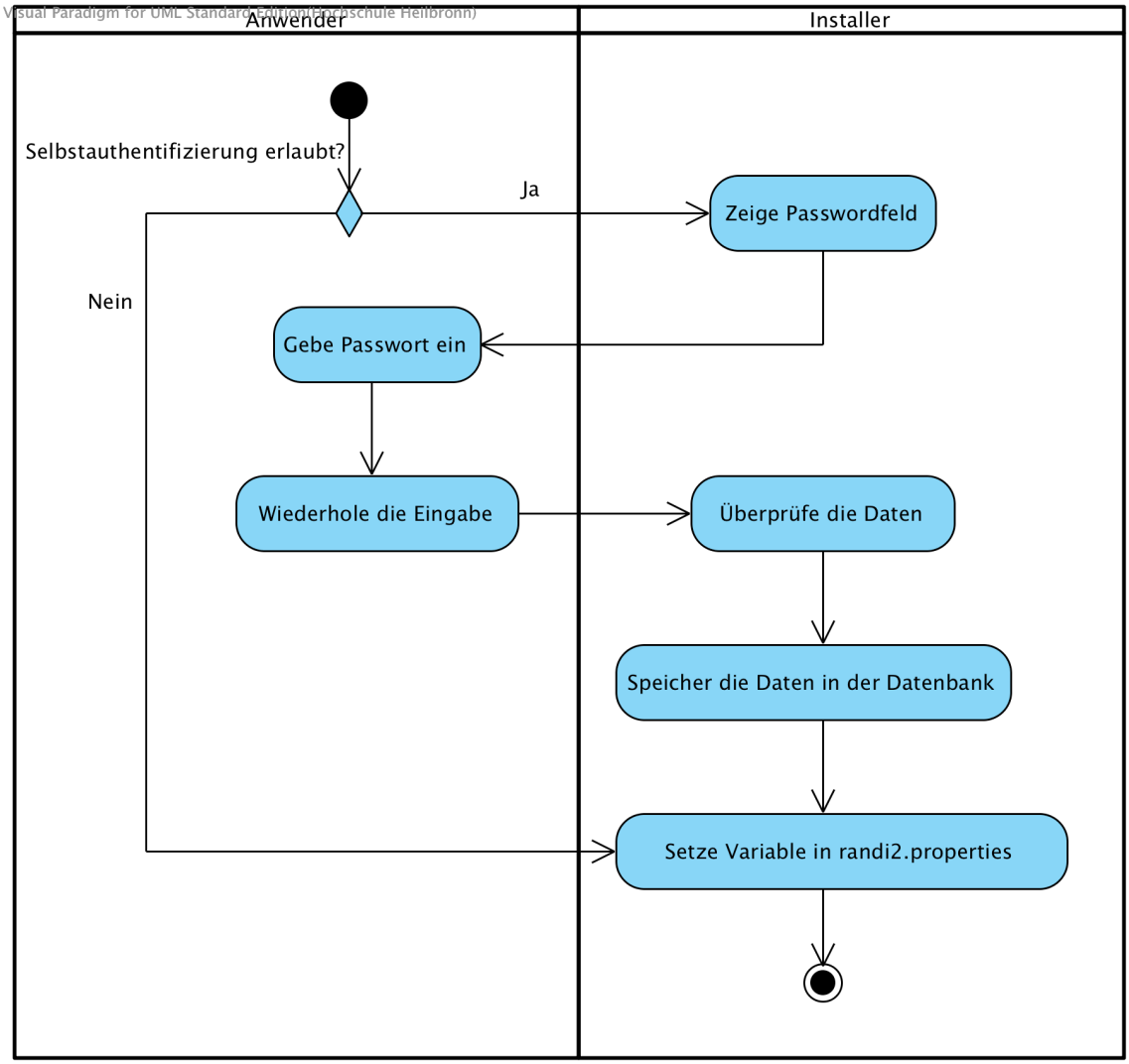

Abbildung 20: Aktivitätsdiagramm - Selbstauthentifizierung am Zentrum

Ebenfalls kann er wählen, ob sich der Benutzer selber am Zentrum authentifizieren kann oder nicht. Sollte der Anwender sich für ja entscheiden, wird er aufgefordert, ein Passwort festzulegen. Mit diesem Passwort bestätigen die Anwender, dass sie zum Zentrum gehören.

3.2.2.4 Aktivität - Logo und Webseite konfigurieren

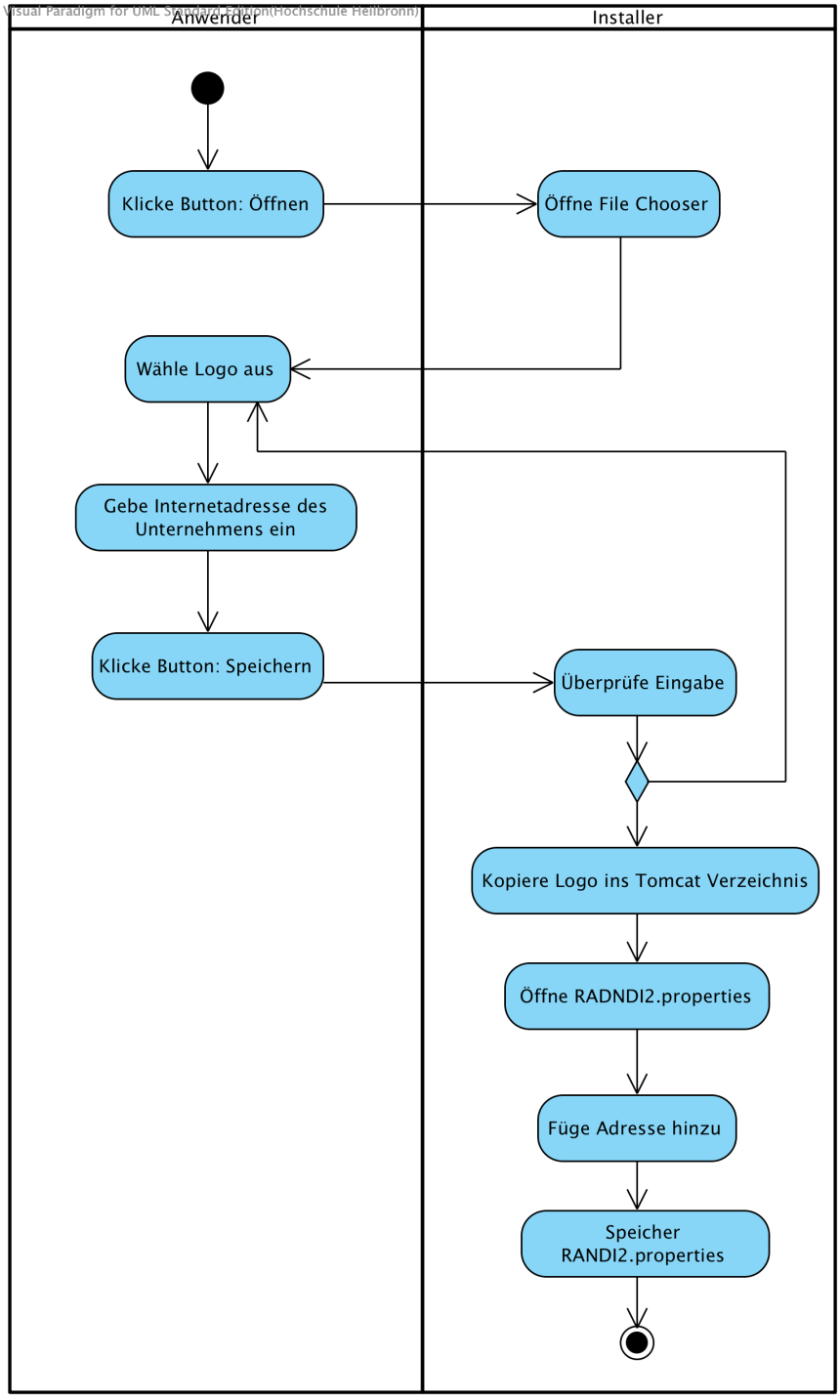

Abbildung 21: Aktivitätsdiagramm - Logo und Webseite konfigurieren

In diesem Schritt geht es darum, ein Logo und die Webseite des Unternehmens zu erfragen. Durch Drücken des Öffnen-Buttons wird der FileChooser angezeigt, mit dem der Benutzer ein Logo auswählen kann. Die Internetadresse kann der Anwender in das vorgesehene Feld eintragen. Nach Drücken des Speichern-Buttons wird die Adresse gespeichert. Das Logo wird in „hostingInstLogo.png“ umbenannt und in das Verzeichnis Tomcat/webapps/RANDI2/resources kopiert.

3.2.2.5 Aktivität - Informationsserver konfigurieren

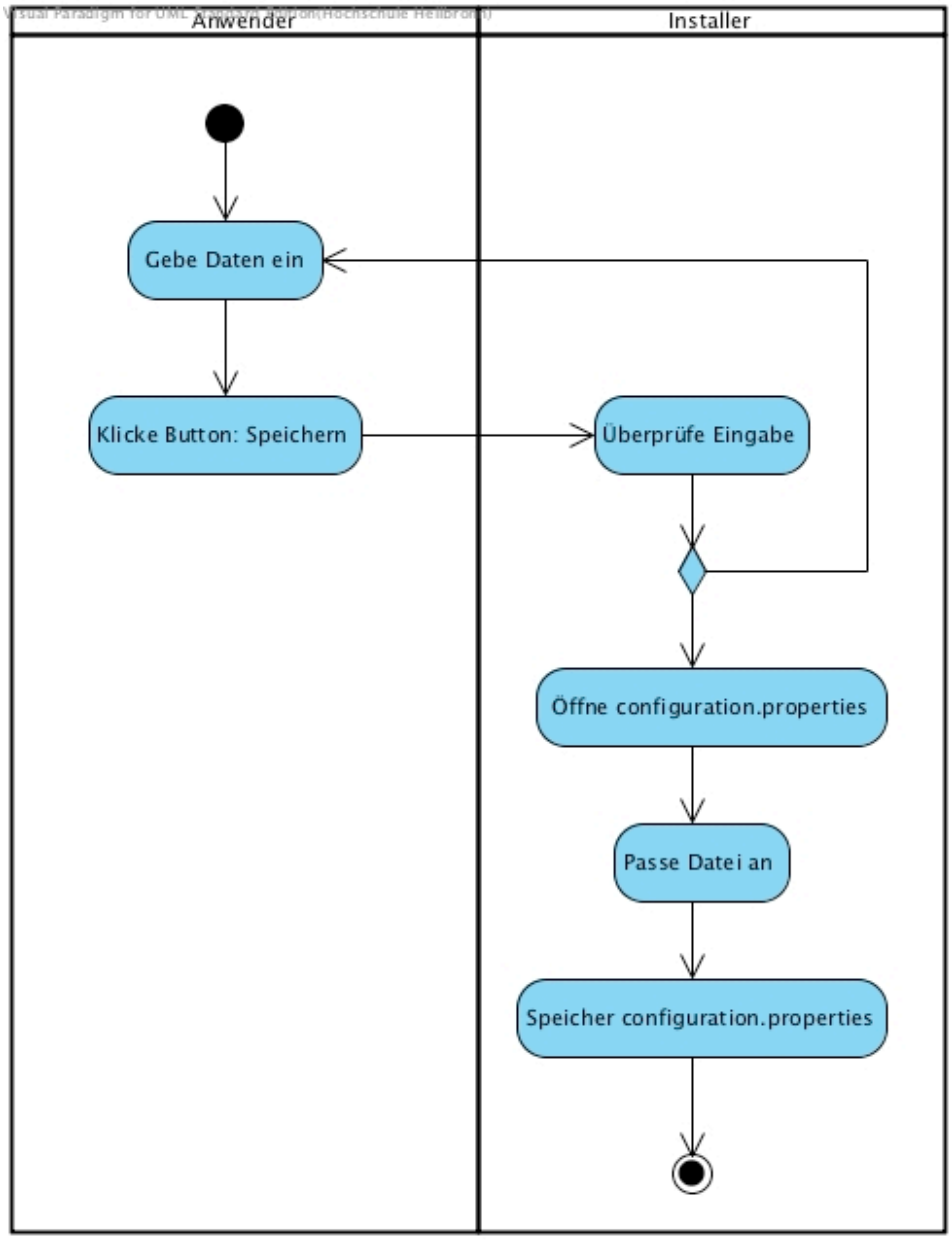

Abbildung 22 Aktivitätsdiagramm - Informationsserver konfigurieren

Der Anwender wird nun aufgefordert, weitere Informationen (Absender E-Mail Adresse, Webseite und die Gruppe) anzugeben. Diese Daten werden, nach einer Überprüfung auf Korrektheit, in der Datei configuration.properties gespeichert.

3.2.2.6 Aktivität - Disclaimer und Informationstext editieren

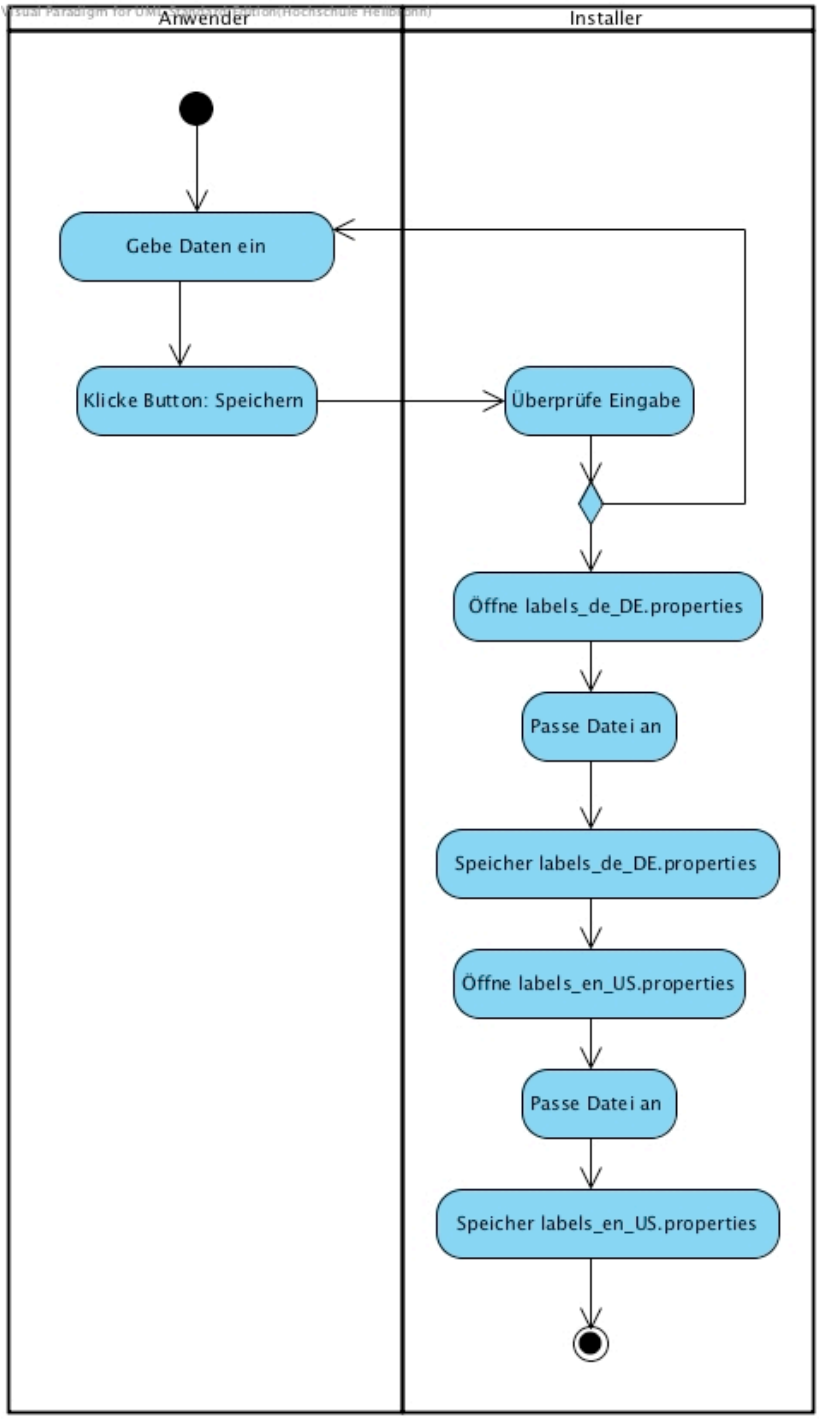

Abbildung 23: Aktivitätsdiagramm - Disclaimer und Informationstext editieren

Zum Schluss der Konfiguration gibt der Anwender noch einen Informationstext und einen Disclaimer ein. Beides wird auf Deutsch und auf Englisch verfasst.
Die Installation inklusive der Konfiguration von RANDI2 ist nun beendet. Der Anwender kann RANDI2 mit seinem Browser unter „127.0.0.1:8080/RANDI2/“ starten. Als Login sind die Administrator Daten zu verwenden.

3.3 Anwendungsbereiche

Angewandt wird die Installationssoftware für RANDI2 dort, wo RANDI2 betrieben werden soll. Die Zielgruppe bezieht sich auf Unternehmen und Forschungsgruppen, die eine freie und kostenlose Randomisierungs-Software benötigen.

3.4 Betriebsbedingungen

Für den Betrieb der Installationssoftware muss Java in der aktuellsten JRE Version vorliegen. Während der Installation müssen einige Tools und Libraries heruntergeladen werden, dazu ist ein Webbrowser inklusive Internetverbindung notwendig. Es besteht jedoch die Möglichkeit, die benötigen Dateien vorab oder auf einem anderen Weg auf dem System zu installieren, so dass keine Internetverbindung und kein Webbrowser erforderlich sind.

Für den eigentlichen Betrieb von RANDI2 sind weitere Bedingungen notwendig. Diese können den RANDI2 Betriebsbedingungen[23] entnommen werden.

[23] www.randi2.org

4 Architektur

Für die Architektur der Installationssoftware gibt es zwei in Frage kommende Lösungen. Zum einen besteht die Möglichkeit, ihn als Stand-Alone Anwendung zu entwickeln. Die andere Lösung ist eine Server-orientierte Webanwendung, wie sie auch von RANDI2 verwendet wird.
Letzteres bietet den Vorteil, dass der Installer dem Design und der Art von RANDI2 angepasst werden kann. Es können sogar zum Teil Klassen übernommen werden. Durch ein einheitliches Design merkt der Benutzer kaum, dass es sich um zwei unterschiedliche Programme handelt, die von unterschiedlichen Personen entwickelt wurden. Dieses engere Zusammenspiel ist als großer Vorteil zu sehen. All diese Punkte werden von einer einzelnen Stand-Alone Anwendung nicht erfüllt.

Als wesentlicher Nachteil steht der Webanwendung jedoch entgegen, dass der Benutzer bereits einen JEE Applikation Server oder einen Servlet Container installiert haben muss. Was wiederum dem Ziel widerspricht, den Installer in der Bedienung so einfach wie möglich zu halten.
Letztes Ziel kann mit einer einfachen Stand-Alone Anwendung, die mit einem Klick gestartet werden kann, gut realisiert werden. Der Benutzer braucht keine großen Computerkenntnisse. Lediglich eine Version von Java wird benötigt, welche auf den meisten System bereits installiert ist. Aus diesen Gründen verwendet der Installer eine Stand-Alone Architektur.

4.1 Modell

Folgendes Diagramm zeigt ein UML Beziehungsdiagramm mit allen Klassen, Packages und den Beziehungen der Installationssoftware. Der Installer lehnt sich an das Model View Controller Prinzip an, welches im Kapitel 2.7 beschrieben wurde.

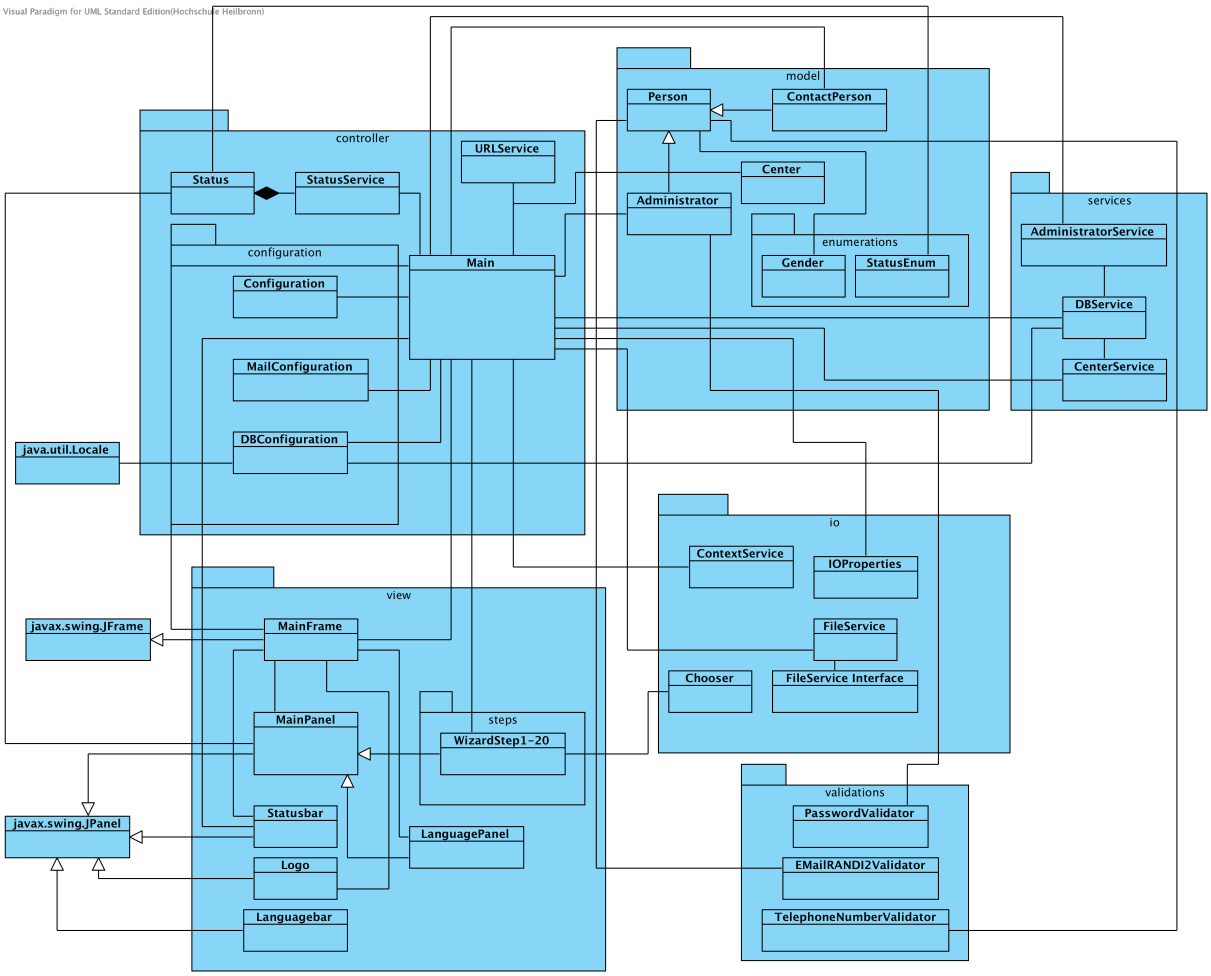

Abbildung 24: Klassendiagramm

4.1.1 Das Package Model

Das Package Model ist für die Datenverwaltung der Installationssoftware zuständig. Es beinhaltet die Klassen *ContactPerson* und *Administrator*, welche von der Klasse *Person* erben. Die Klassen besitzen Parameter, inklusive der *getter()* und *setter()* Methoden, wie Telefonnummer, Vor- Nachname oder E-Mail Adresse. Zudem befinden sich im Package Model die Klasse *Center* und die Enumerationen *Gender* und *StatusEnum*. *Gender* kann die Werte MALE und FEMALE annehmen. *StatusEnum* die drei möglichen Werte des Status (SUCCESS, FAIL und UNMACHINED).

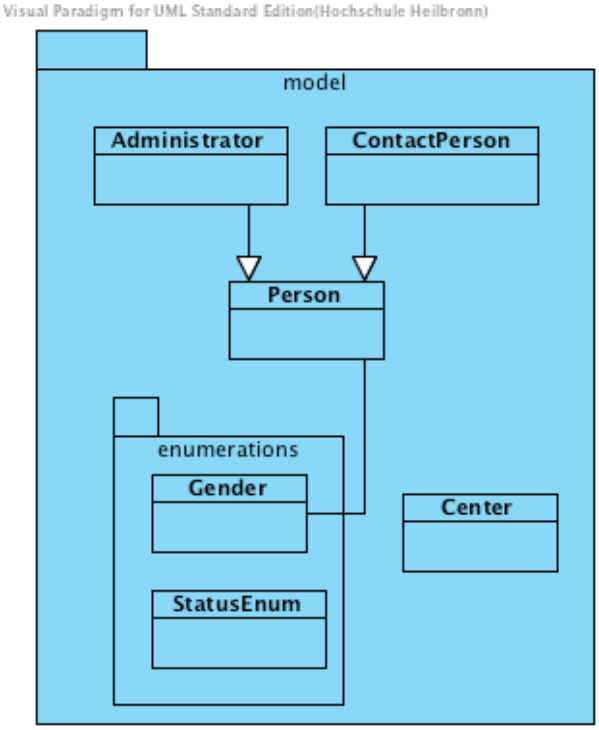

Abbildung 25: Package Model

4.1.2 Das Package View

Die Klassen des Packages View verwendet die Grafikbibliothek Java Swing[24] zum Aufbau und zur Darstellung der grafischen Benutzeroberfläche. Die Hauptklasse in diesem Package ist die Klasse *MainFrame*. Diese Klasse erbt von der Klasse *JFrame* und initialisiert ein Fenster. Auf dem *MainFrame* befinden sich unter anderem Objekte der Klassen *Statusbar*, *LanguagePanel* und *Logo*. All diese Klassen erben von der Klasse *JPanel*. Diese Objekte werden über die gesamte Lebenszeit des *MainFrame* auf diesem dargestellt.
Außerdem befinden sich Objekte der Klassen *WizardSteps1-20*, welche eine Instanz der Klasse *JPanel* erzeugen, auf dem MainFrame. Diese Objekte werden je nach Status der Installation angezeigt, wobei immer nur ein Objekt zeitgleich angezeigt werden kann.

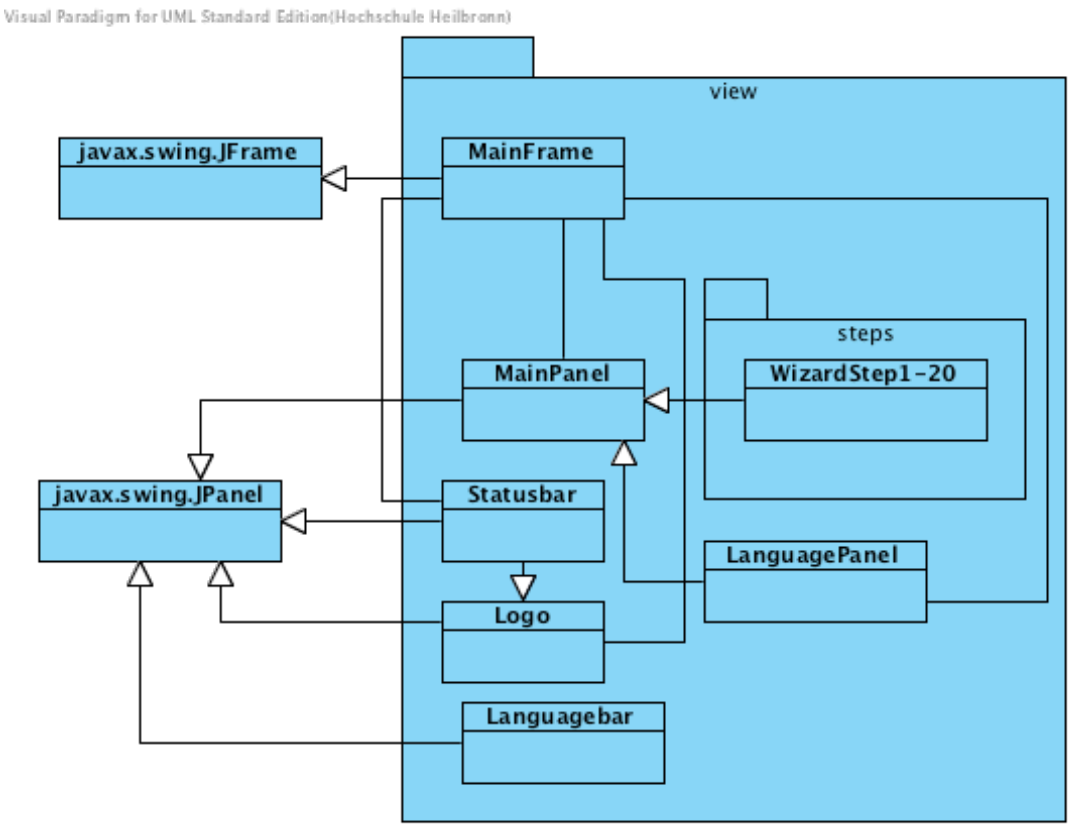

Abbildung 26: Package View

[24] Swing: http://java.sun.com/javase/technologies/desktop/

4.1.3 Das Package Controller

Das Package Controller ist für die Verwaltung der Installationssoftware zuständig. Zu dem Package gehört unter anderem die Klasse *Main*, die alle View Objekte erzeugt und deren Eingaben entgegennimmt. Zusätzlich verwaltet die Klasse *Main* Objekte, wie den *Administrator* oder *Center*, die wiederum die eingegebenen und verarbeiteten Daten speichern.
Ebenfalls von zentraler Bedeutung sind die Klassen des Packages controller.configuration. Sie speichern die Konfigurationen, die der Benutzer während des Installationsprozesses vornimmt. Dazu gehören z.B. der Tomcat Pfad oder die SQL-Server Zugangsdaten.
Für den aktuellen Stand der Installation ist die Klasse *StatusService* zuständig. Sie besitzt Objekte der Klasse *Status*. Diese verwalten den Fortschritt der Installation. Dazu speichern die Objekte, ob ein Installationsschritt erfolgreich war, fehlgeschlagen ist oder noch nicht bearbeitet wurde.
Die Klasse *URLService* öffnet einen Browser mit bestimmten Internetseiten damit der Anwender die benötigten Tools herunterladen kann.

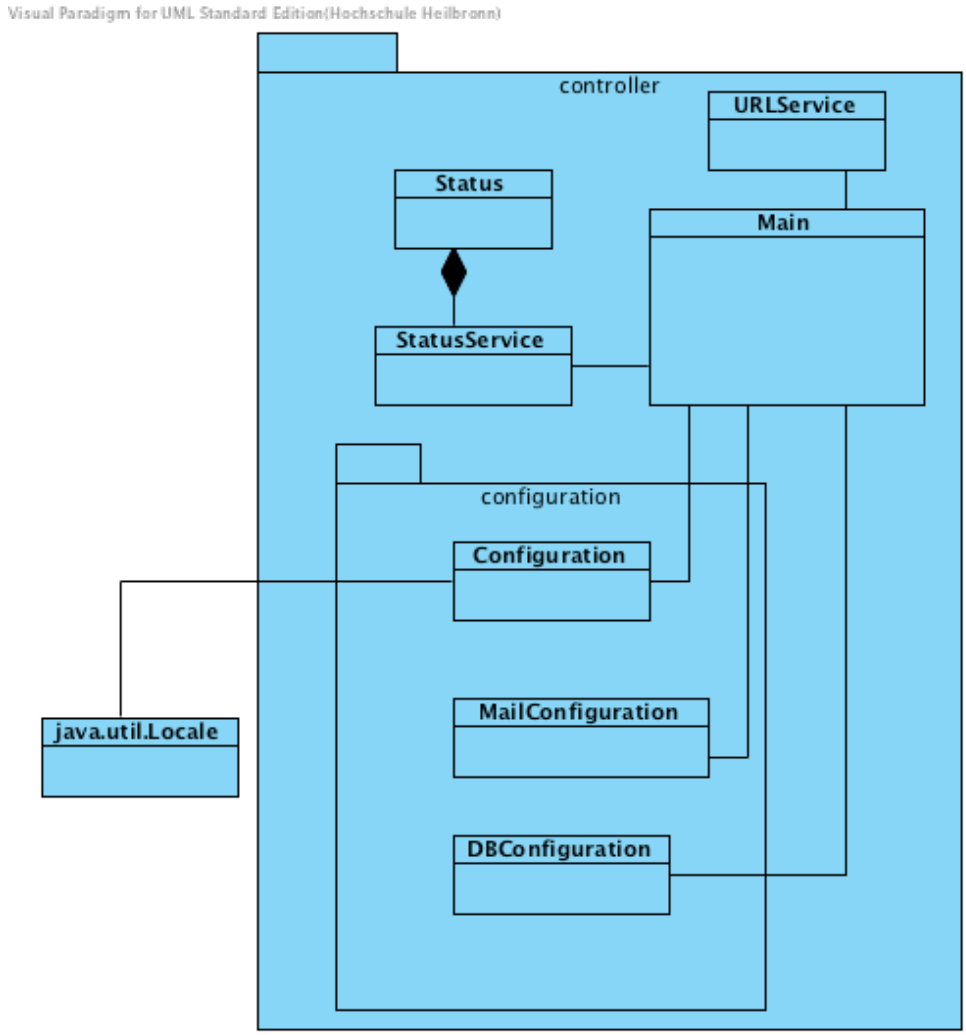

Abbildung 27: Package Controller

4.1.4 Das Package Service

Das Package Service verwaltet die Klassen, die für die Datenbankanbindung verantwortlich sind. Die zentrale Klasse dieses Packages ist die Klasse *DBService*. Mit Hilfe dieser Klasse baut der Installer eine Verbindung zum SQL-Server auf und legt die Datenbank, sowie den Benutzer an. Die Klassen *AdministratorService* und *CenterService* werden dazu verwendet, um Daten in der Datenbank persistent zu speichern. Dieser Vorgang erfolgt mit Hilfe der Klasse *DBService*.

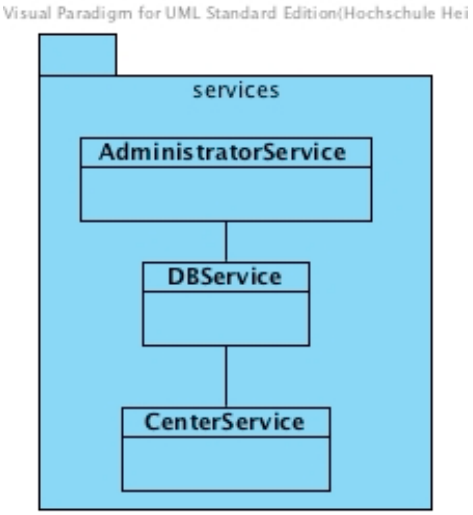

Abbildung 28: Package Service

4.1.5 Das Package IO

Im Package IO werden die notwendigen Daten für die Konfiguration von RANDI2 in Properties- (IOProperties) oder XML- (ContextService) Dateien gespeichert. Ebenfalls heraus zu stellen ist die Klasse *FileService*. Sie ermöglicht das Kopieren und Umbenennen von heruntergeladenen Dateien in ein Zielverzeichnis. Dies wird z.B. benötigt, um die RADNI2.war-Datei in den Tomcat/webapps Ordner zu kopieren. Die Klasse *Chooser* öffnet den dafür notwendigen Dialog, um die Dateien auszuwählen.

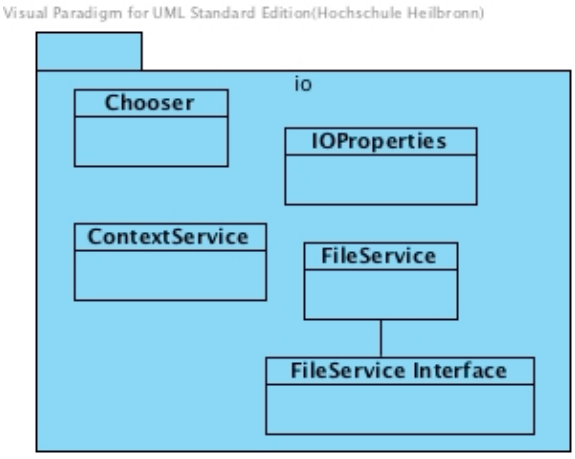

Abbildung 29: Package IO

4.2 Oberflächenentwurf

Der RANDI2 Installer hat bewusst eine einfach gehaltene Oberfläche. Folgendes Bild zeigt einen möglichen Zustand des Installer.

Abbildung 30: Eingabemaske RANDI2

In der oberen Hälfte befinden sich standardmäßig das RANDI2 Logo, eine deutsche Flagge und eine Flagge der Vereinigten Staaten von Amerika. Durch das Drücken auf eine Flagge kann die jeweilige Sprache ausgewählt werden.
Unterhalb des Logos und der Flaggen befindet sich eine Statusbar. Diese zeigt für jeden Installationsschritt den jeweiligen Status an. Der aktuelle Status wird dabei hervorgehoben.

Folgende Zustände sind möglich:

Symbol	Zustand
	Schritt fehlgeschlagen. Dies tritt z.B. ein, wenn die Datenbankverbindung nicht erfolgreich war.
	Schritt noch nicht ausgeführt oder übersprungen.
	Schritt erfolgreich durchgeführt.

Tabelle 1: Zustandstabelle

Unterhalb der Statusbar befindet sich das Hauptfenster, welches abhängig vom Installationsschritt, die notwendigen Informationen anzeigt bzw. abfragt. Im Hauptfenster muss der Benutzer Daten eingeben oder benötigte Programme herunterladen.

Im unteren Teil des Fensters werden die Statusinformationen angezeigt. Zurzeit liegt kein Fehler vor, deswegen wird der Text „Aktueller Status: OK“ angezeigt. Sollte ein Fehler auftreten, z.B. weil nicht alle Pflichtfelder ausgefüllt wurden, wird der aktuelle Kreis in der Statusbar rot und in der Statusinformation erscheint ein Text in dem steht, dass der Benutzer alle Pflichtfelder ausfüllen soll.

Rechts neben der Statusinformation sind ein Zurück- und ein Weiter-Button angeordnet, mit denen der Benutzer durch die Installation navigieren kann. Ein Aufrufen des nächsten Schrittes ist allerdings nur möglich, wenn der aktuelle Schritt erfolgreich absolviert wurde.

4.2.1 Hauptfenster

Der Installer verwendet zwei Hauptfenster Typen, deren Funktionen im Folgenden näher beschrieben werden.
Der erste Hauptfenster Typ ist in Abbildung 30 dargestellt und zeigt eine Eingabemaske. In dieser Eingabemaske gibt der Benutzer die erforderlichen Daten ein. Durch das Drücken des Speichern-Buttons, werden die Daten nach erfolgreicher Überprüfung in ein Objekt gespeichert oder in die Datenbank geschrieben. Möchte der Benutzer die eingetragenen Daten ändern, so muss er sie in den Textfeldern korrigieren und erneut den Speichern-Button drücken.

Der zweite Hauptfenster Typ ist in Abbildung 31 dargestellt und zeigt eine Download-Maske. Mit Hilfe dieser Maske lädt der Benutzer das benötigte Tool herunter. Anschließend wählt er den Pfad zur heruntergeladenen Datei aus.
Wurde das Tool bereits heruntergeladen, so besteht kein Zwang dies erneut durchzuführen. Der Nutzer gibt in diesem Fall den Pfad direkt an. Bei einer korrekten Auswahl des Pfads wird der Zustand, trotz fehlendem Downloads, erfolgreich abgeschlossen. Der Anwender kann somit den nächsten Schritt aufrufen.

Abbildung 31: Download-Maske RANDI2

5 Implementierung

Dieses Kapitel beschreibt die Implementierung der in Java geschriebenen Installationssoftware. Dafür wird zum einen auf die allgemeine Implementierung und zum anderen auch auf speziellen Code eingegangen. Um eingebundene Frameworks zu verwalten und auf Grund der weiteren Vorteile, die im Kapitel 2.4 über Maven erarbeitet wurden, wird Maven auch bei der Entwicklung der Installationssoftware eingesetzt. Die Versionsverwaltung wird über Git mittels Github [25] gesteuert.

5.1 Implementierung der Main-Klasse

Die *Main*-Klasse wird beim Start der Installationssoftware als erstes ausgeführt. Sie erstellt gleich zu Anfang alle nötigen Objekte der verschiedenen Klassen, wie *StatusService*, *DBService*, *IOProperties*, *Person* usw..
Die *Main*-Klasse hat zudem Methoden, welche von der View gestartet werden und weitere Vorgänge ausführen.
Gibt z.B. der Anwender die neuen Kontaktdaten des *Administrators* ein, wird beim Klicken des Speicher-Buttons, welcher in der View erstellt wird, in der *Main*-Klasse die Methode *editAdmin()* aufgerufen. Diese wiederum ruft eine Methode in der Klasse *AdminService* auf. Die benötigten Parameter erhält das Objekt der Klasse *AdminService* dabei von der *Main*-Klasse, die ein Objekt der Klasse *Administrator* beim Methodenaufruf mitliefert.

5.2 Implementierung der Oberfläche

Als Hauptframework für die Oberflache verwendet die GUI[26] Swing. Dieses Framework stellt ein *JFrame* und die *JPanels* zur Verfügung, auf denen die ebenfalls aus Swing stammenden Objekte wie Textfelder oder Button eingefügt werden.

Die GUI ist so aufgebaut, dass in der *Main*-Klasse ein Objekt, der Klasse *MainFrame*, erzeugt wird. Dieses Objekt bleibt während der kompletten Installation bestehen. Es initialisiert direkt Objekte der Klasse *Statusbar*, *Logo* und *Languagebar*, die ebenfalls die ganze Zeit auf dem *MainFrame* bestehen bleiben.
Die Positionen dieser Objekte auf dem *MainFrame* ändern sich während der Ausführung des Programms nicht, sodass diese bei der Initialisierung fest gesetzt werden können.
Ebenfalls fest auf dem *MainFrame* platziert sind die beiden JButtons Zurück und Weiter, sowie die Statusinformation in Form eines nicht veränderbaren *JTextFields*, das unten links auf der Oberfläche angezeigt wird.

In der Mitte des *MainFrames* befinden sich die einzelnen Oberflächen der Installationsschritte. Realisiert werden diese über die vom *MainPanel* abgeleiteten Klassen *WizardStep1*, *WizardStep2*, [...], *WizardStep20*. Die Superklasse *MainPanel* wird, wie die Subklassen *WizardSteps*, am Anfang des Programms in der *Main*

---

[25] GitHub: git@github.com:akudak/RANDI2-Installer.git

[26] GUI: Graphical User Interface dt. Grafische Benutzeroberfläche

Klasse angelegt. Jedoch werden diese Objekte erst bei Bedarf auf dem *MainFrame* platziert. Nach Ende des Installationsschrittes wird der *WizardStep* wieder entfernt. Das Objekt bleibt jedoch bestehen, sodass es jederzeit wieder eingefügt werden kann. Dies hat den Vorteil, dass pro *WizardStep* immer nur ein Objekt existiert. Durch das *MainPanel*, welches Größe, Position und Hintergrundfarbe vorgibt, werden die *WizardSteps* vereinheitlicht.
Auf den einzelnen *WizardSteps* können dann nach Bedarf *JButton*, *JLabels* oder *JTextFields* eingefügt werden.
Eingegebener Text in *JTextFields* oder *JPasswordFields* wird erst übernommen, nachdem der Speicher-Button gedrückt wurde. Dieser wird bei Bedarf unten rechts auf dem *WizadStep* angezeigt.

5.3 Implementierung der Datenbankzugriffe

Für die Verwaltung der Zugriffe auf die Datenbank ist die Klasse *DBService* verantwortlich. Sie enthält unter anderem die Methoden *getFirstConnection()* und *getConnection()*. Beide Klassen geben eine Connection zum SQL-Server zurück. Der Unterschied der beiden Methoden besteht darin, dass sich die Methode *getFirstConnection()* mit keiner Datenbank verbindet. Dies ist Anfangs von Bedeutung, da zuerst noch eine Datenbank vom Installer erstellt werden muss. Die Methode *getConnection()* verbindet sich dagegen direkt mit der Datenbank. Diese Methode wird benötigt, um Daten in der angelten Datenbank zu schreiben oder zu lesen.

Die Tabellen in der Datenbank werden in der Methode *executeSQLDBScript()* initialisiert. Hierfür wird ein SQL-Skript eingelesen. Dabei wird jede Zeile einzeln eingelesen, Kommentare und leere Zeilen werden übersprungen. An dem Semikolon, welches sich laut Syntax von MySQL und PostgreSQL immer am Schluss einer Zeile befindet, erkennt die Methode, dass die Zeile vollständig ist und sendet sie an den SQL-Server.

Um Parameter in der Datenbank zu aktualisieren, werden die Klassen *AdministratorService* und *CenterService* verwendet. Beide Klassen benutzen für die Update-Anweisung ein Prepared Statements, um ungültige Eingaben und böswillige SQL-Injektionen zu vermeiden.
Zurzeit ist es so, dass sich in den Tabellen Objekte befinden, da immer dasselbe SQL-Skript verwendet wird.
Es ist jedoch nicht auszuschließen, dass sich das SQL-Skript im Laufe der Entwicklung von RANDI2 ändert. Aus diesem Grund wird eine INSERT-Anweisung ausgeführt, sofern die Tabellen leer sind. Eine UPDATE-Anweisung in einer leeren Tabelle würde zu einem Fehler führen.

5.4 Implementierung der Statusbar

Jeder Schritt des Installer besitzt einen eigenen Status, der im *MainPanel* erstellt wird. So haben die abgeleiteten Klassen *WizardStep* gleich von Anfang an ein Objekt der Klasse *Status*. Der Zustand des *Status* wird mittels Enumerationen interpretiert. Dabei bedeutet der Wert „FAIL“, dass der Schritt fehlgeschlagen ist. Der Wert „UNMACHINED“ beutet, dass dieser Installationsschritt noch nicht bearbeitet wurde und der Wert „SUCCESS“ steht dafür, dass der Schritt erfolgreich abgeschlossen wurde. Zudem weiß der *Status*, ob er gerade aktiv ist oder nicht. Hierbei gilt, dass immer nur ein *Status* aktiv sein kann. Das Wissen über den *Status*, der zurzeit aktiv ist, ist für das Setzten der Parameter des *Status* von Bedeutung. Schlägt z.B. ein Datenbankzugriff fehl, so wird über *StatusService* der aktuelle *Status* als fehlgeschlagen gesetzt.

Um eine Übersicht über alle Zustände zu erlangen, gibt es die Klasse *StatusService*, die eine ArrayListe mit allen Zuständen verwaltet. Über ihre Methoden können auch der aktuelle und der nächste *Status* abgerufen werden.
Das Setzten des aktuellen *Status* erfolgt beim Betätigen der Weiter- oder Zurück-Buttons. Hier wird jeweils der nächste bzw. vorherige *Status* in der ArrayList ausgewählt. Sollte sich die ArrayList am Anfang oder Ende befinden, ist der jeweilige Button durch *JButton.setEnable(false)* gesperrt, sodass kein ungültiger Status ausgewählt werden kann.

5.5 Implementierung der IOProperties

In der Klasse *IOProperties* werden die verschiedenen Properties Dateien von RANDI2, sowie vom Installer eingelesen und geschrieben.

Java.util.Properties stellt die Klasse Properties zur Verfügung, sodass zum Einlesen der Datei die Methode *properties.load(new FileInputStream(path)):Properties* verwendet werden kann.
Über *properties.setProperty()* und *properties.getProperty()* können die einzelnen Werte der Properties-Datei geschrieben bzw. gelesen werden.
Das Speichern der Properties Dateien erfolgt dann wieder über *propeties.store(new FileOutputStream(path))*.

5.6 Implementierung des ContextService

Die Klasse *ContextService* wird verwendet, um die XML Datei, die sich im Tomcat Ordner befindet, zu bearbeiten. In die Context.xml müssen die JDNI Konfigurationen und die Zugangsdaten für den SQL- und Mail-Server eingetragen werden.
Es gibt Java APIs, um XML Dateien zu editieren, jedoch werden die sogenannten Knoten dabei jeweils mit </Knotenname> beendet. Beim Verwenden dieser Endung kann es bei Tomcat zu Problemen beim Auslesen kommen. Aus diesem Grund habe ich mich entschieden, die XML Datei über *new BufferedReader(new FileReader(path))* einzulesen und dann den benötigten Text mit der Endung /> einzufügen. Anschließend wird die Datei mit einem *FileWriter* wieder ins Tomcat Verzeichnis geschrieben.

6 Test

Der Sourcecode, der Installationssoftware, wird mit Hilfe des Frameworks JUNIT[27] in der Version 4.10 getestet. Die Funktionen der Oberfläche werden durch das Durchführen des Installationsprozesses überprüft. Dabei werden gültige und ungültige Schritte ausgeführt. Besonderen Wert ist dabei auf die Funktionen der Fehlerausgabe als Text über die Statusleiste und auf das korrekte Anzeigen des Status in der Statusbar zu legen. Ebenfalls soll überprüft werden, ob leere Angaben angenommen werden.

6.1 Test des Sourcecodes

Im Folgenden werden die Tests des Sourcecodes näher beschrieben. Die Funktionsfähigkeit der Klassen wird durch jeweils eine Testklasse überprüft, welche sich unter src/test/java befinden. Die Testklasse hat dabei den gleichen Namen wie die zu testende Klasse, inklusive des Suffix Tests.

6.1.1 Test der Klasse IOProperties

Testziel:

Ziel des Tests ist es, zu überprüfen, ob die Methoden der Klasse *IOProperties* erfolgreich Informationen in die jeweiligen Properties-Dateien schreiben.

Testklasse: *IOPropertiesTest*

Vorbedingungen:

Für diesen Test müssen Properties-Dateien vorhanden sein, die denen von RANDI2 ähnlich sind.
Um eine komplette Installation von TOMCAT und RANDI2 zu vermeiden, befinden sich im Verzeichnis src/test/resources/[...]/ die Properties-Dateien.
Die Datei RANDI2.properties befindet sich demnach unter test/resources/webapps/RANDI2/RANDI2.properties.

Durchführung:

Für den Test werden die einzelnen Properties-Dateien mit Werten gefüllt und gespeichert. Anschließend werden die Dateien wieder geöffnet und die Werte ausgelesen. Diese Werte werden mit den zuvor eingetragenen und ungültigen Werten verglichen.

Erwartetes Ergebnis:

Es wird erwartet, dass bei den positiven Tests die Werte ausgelesen werden, die zuvor eingetragen wurden.

---

[27] JUnit: www.junit.org

Bei den negativen Tests wird erwartet, dass nicht die Werte ausgelesen werden, welche beim positiven Test verwendet werden.

6.1.2 Test der Klasse FileService

Testziel:

Ziel des Tests ist es, zu überprüfen, ob die Methoden *copy()* und *rename()* der Klasse FileService erfolgreich Dateien kopieren und umbenennen können.

Testklasse: *FileServiceTest*

Vorbedingungen:

Es muss eine Test Datei mit dem Namen TestDatei.txt vorhanden sein. Der Inhalt der Datei lautet: „123456“.

Durchführung:

Getestet werden die Methoden *rename()* und *copy()*. Dazu werden die Methoden einzeln mit der Test Datei aufgerufen. Anschließend wird beim Test der Methode *copy()* überprüft, ob sich im Zielordner die TestDatei.txt befindet. Beim Test der Methode *rename()* wird überprüft, ob sich im selben Verzeichnis die Datei mit neuem Namen befindet. Bei beiden Testvorgängen wird ebenfalls überprüft, ob der Inhalt „123456“ nicht verändert wurde.

Erwartetes Ergebnis:

Es wird erwartet, dass die Datei beim Test *copy()* erfolgreich kopiert wird. Der Inhalt der Datei wird auf „123456“ vermutet.

Beim Test *rename()* wird die Datei im selbem Verzeichnis unter dem Namen TestDatei2.txt vermutet. Auch hier soll der Inhalt „123456“ sein.

Schlussbedingungen:

Zum Schluss der Tests wird das Kopieren und Umbenennen wieder rückgängig gemacht, um einen weiteren Test zu ermöglichen.

6.1.3 Test der Klasse AdministratorService

Testziel:

Ziel des Tests ist es, zu überprüfen, ob die Klasse *AdministratorService* erfolgreich Werte in der Datenbank aktualisiert.

Testklasse: *AdministratorServiceTest*

Vorbedingungen:

Als Vorbedingung für diesen Test muss der SQL-Server gestartet sein.
Vor jedem Test wird die Datenbank mit einem Benutzer und den Tabellen angelegt.
Zudem wird vor dem Testvorgang ein Objekt der Klasse *Administrator* angelegt.

Durchführung:

Die Werte des Objektes *Administrator* werden in die Tabellen Person und Login geschrieben. Beim Auslesen werden die Daten auf Korrektheit überprüft.
Das Eintragen von ungültigen Werten würde bereits beim Erstellen des Objektes *Administrator* fehlschlagen, sodass dies hier nicht getestet werden muss.

Erwartetes Ergebnis:

Es wird erwartet, dass dieselben Werte ausgelesen werden, die zuvor geschrieben wurden. Außerdem soll in den Tabellen die Spalte updatedAt als Wert das Datum und die Zeit der Aktualisierung beinhalten.

Schlussbedingungen:

Die Datenbank wird wieder gelöscht.

6.1.4 Test der Klasse CenterService

Testziel:

Ziel des Tests ist es, zu überprüfen, ob die Methoden der Klasse *CenterService* erfolgreich Informationen in der Datenbank aktualisieren.

Testklasse: *AdministratorServiceTest*

Vorbedingungen:

Es muss eine Verbindung zum SQL-Server bestehen, der die Datenbank mit allen Tabellen und einen Benutzer mit passenden Zugriffsrechten besitzt.
Zudem werden Objekte der Klassen *Center* und *ContactPerson* erstellt.

Durchführung:

Die Werte des Objektes *Center* und *ContactPerson* werden in die Tabellen TrialSite und Person geschrieben. Beim Auslesen werden die Daten auf Korrektheit überprüft. Das Eintragen von ungültigen Werten würde bereits beim Erstellen der Objekte *ContactPerson* und *Center* fehlschlagen, sodass es hier nicht getestet werden muss.

Erwartetes Ergebnis:

Es wird erwartet, dass dieselben Werte ausgelesen werden, die zuvor geschrieben wurden. Außerdem soll in den Tabellen die Spalte updatedAt als Wert das Datum und die Zeit der Aktualisierung beinhalten.

Schlussbedingungen:

Die Datenbank wird wieder gelöscht.

6.1.5 Test der Klasse DBService

Testziel:

Ziel ist es, zu überprüfen, ob die Methoden *createUser(), createDatabase()* und *executeSQLDBScript()* der Klasse *DBService* erfolgreich funktionieren.

Testklasse: *DBServiceTest*

Vorbedingung:

Als Vorbedingung für die Tests muss ein MySQL-Server erreichbar sein. Standardmäßig ist er unter 127.0.0.1 zu erreichen. Der Server darf für die Tests *createUser()* und *CreateDatabase()* keine Datenbank und keine Benutzer enthalten. Eventuelle Datenbanken oder Benutzer werden vor dem jeweiligen Test gelöscht. Für den Test *executeSQLDBScript()* muss eine Datenbank mit einem Benutzer angelegt werden.

Durchführung:

Um die Methode *createDatabase()* zu testen, wird die Methode *createDatabase()* der Klasse *DBService* aufgerufen. Sie legt eine Datenbank an. Anschließend wird von der Testklasse eine Tabelle angelegt und bestückt. Können Werte in die Tabelle geschrieben werden, so gilt der Test als erfolgreich.

Für den Test der Methode *createUser()* werden zuerst alle Benutzer mit dem Namen, der später eingefügt werden soll, gelöscht.
Anschließend wird ein neuer Benutzer angelegt und der erwartete Wert des Namens wird verglichen.

Die Methode *executeSQLDBScript()* wird getestet, indem das SQL-Skript auf der Datenbank ausgeführt wird. Anschließend wird überprüft, ob die zu erwartenden Tabellen vorhanden sind.

Erwartetes Ergebnis:

Beim Test *createDatabase()* wird erwartet, dass die Datenbank erstellt wurde und erfolgreich in die Tabelle geschrieben werden kann.

Beim Test der Methode *createUser()* wird erwartet, dass der Name des neuen Benutzers identisch zu dem Namen des angelegten Benutzers ist.

Beim Test *executeSQLDBScript()* wird erwartet, dass die Tabellen der Datenbank erfolgreich angelegt wurden.

Schlussbedingungen:

Alle Datenbanken und Tabellen werden nach dem jeweiligen Test wieder gelöscht.

6.1.6 Test der Klasse Person

Testziel:

Überprüft werden soll bei diesem Test, ob die Methoden der Klasse *Person* wie erwartet reagieren. Besonderer Wert soll darauf gelegt werden, ob ungültige Eingaben von den Methoden abgelehnt werden.

Testklasse: *PersonTest*

Vorbedingung:

Es wird ein Objekt der Klasse *Person* benötigt.

Durchführung:

Die Klasse hat ausschließlich *get()*- und *set()*-Methoden.
Aus diesem Grund werden die Parameter mit gültigen und ungültigen Werten gesetzt. Als ungültige Werte werden null, ein leerer String, bei Telefonnummern die Angabe von Buchstaben und bei E-Mail Adressen das Weglassen der Domain, dem @ Zeichen oder beides gesehen.
Über die *get()* Methoden werden die gültigen Werte ausgelesen und mit den eingetragenen Werten verglichen.

Erwartetes Ergebnis:

Bei den gültigen Werten wird erwartet, dass die *get()* Methoden dieselben Werte liefern, die eingetragen wurden.
Bei Eintragen in die *set()* Methoden wird bei den gültigen Werten der Rückgabewert *true* erwartet, der bestätigt, dass die Werte erfolgreich eingetragen wurden.
Bei den ungültigen Eintragungen wird der Rückgabewert *false* erwartet. Dieser gibt an, dass die Werte nicht eingetragen werden konnten, da sie ungültig sind.

6.1.7 Test der Klasse Administrator

Testziel:

Überprüft werden soll bei diesem Test, ob die Methoden der Klasse *Administrator* richtig funktionieren. Besonderer Wert soll darauf gelegt werden, ob ungültige Eingaben von den Methoden abgelehnt werden.

Testklasse: *AdministratorTest*

Vorbedingungen:

Als Vorbedingung wird ein Objekt der Klasse *Administrator* erstellt.

Durchführung:

Hier werden nur die zusätzlichen Methoden *setUsername()*, *setPassword()* und *encode()* getestet. Alle anderen Methoden, welche von der Klasse *Person* geerbt werden, werden in *PersonTest* auf ihre Funktionsfähigkeit überprüft.

Die Methode *setUsername()* wird getestet, indem sowohl gültige Werte als auch ungültige Werte eingetragen werden. Ein ungültiger Wert ist ein leerer Benutzername oder ein Benutzername mit mehr als 100 Zeichen.

Zum Testen der Methode *setPassword()* werden ein gültiger SHA String (64 Zeichen) und zwei ungültige SHA Strings, einmal mit unter 64 und einmal mit über 64 Zeichen, gesetzt.

Für den Test *encore()* werden zwei gleiche Passwörter übergeben und mit dem zu erwartenden SHA Wert verglichen. Für den negativen Test werden zwei ungleiche Passwörter übergeben.

Erwartetes Ergebnis:

Erwartet wird, dass bei den *set()* Methoden der Rückgabewert *true* geliefert wird, sofern es sich um gültige Eingaben handelt. Bei ungültigen Eingaben wird der Wert *false* erwartet.
Beim Testen der Methode *encore()* wird der Rückgabewert *true* und nach dem Aufruf der Methode *getPassword()* ein gültiger SHA Wert erwartet, sofern zwei gleiche Passwörter eingegeben wurden.
Beim Setzten der ungleichen Passwörter wird der Rückgabewert *false* erwartet.

6.1.8 Tests der Klassen Configuration

Die Klassen *Configuration*, *DBConfiguration* und *MailConfiguration* sind von der Struktur gleich aufgebaut, weshalb sich ihre Testklassen auch nicht von der Art und Weise des Tests unterscheiden. Aus diesem Grund können die Testklassen einheitlich beschrieben werden.

Testziel:

In diesem Test soll überprüft werden, ob die Methoden der Klassen *Configuration*, *DBConfiguration* und *MailConfiguration* erfolgreich Werte speichern und wiedergeben können.

Testklassen: *ConfigurationTest*, *DBConfigurationTest* und *MailConfigurationTest*

Vorbedingung:

Als Vorbedingung wird in jeder Testklasse ein Objekt der jeweiligen Klasse erstellt.

Durchführung:

Beim Testvorgang werden die einzelnen Parameter mit gültigen und ungültigen Werten, wie null oder einem leeren Inhalt, gesetzt.
Des Weiteren werden die *get()* Methoden aufgerufen und die Ergebnisse werden mit den zuvor gesetzten Werten verglichen.

Erwartetes Ergebnis:

Beim Testen der *set()* Methoden wird erwartet, dass bei der Verwendung von gültigen Werten *true* und bei ungültigen Werten *false* zurückgeben wird.
Bei den *get()* Methoden wird erwartet, dass der zuvor gesetzte Wert zurückgeben wird.

6.2 Test der Benutzeroberfläche

In diesem Test werden die Funktionen der Benutzeroberfläche überprüft. Die Logik des Programms wird mit der Testumgebung JUNIT getestet. Ihre Funktion wird während der Oberflächentests als funktionierend vorausgesetzt.

6.2.1 Test der Pflichtfelder

Testziel:

Es soll die Funktionsfähigkeit der Pflichtfelder überprüft werden.

Vorbedingungen:

Die Funktion der *Statusbar* und der *Statusinformation* wird in diesem Test als erfolgreich angesehen. Ihre Funktionen werden in einem anderen Test überprüft.

Durchführung:

In dem positiven Test wird überprüft, ob der Zustand als erfolgreich gesetzt wird, wenn alle Pflichtfelder ausgefüllt werden. Dazu werden die Zugangsdaten des neuen Datenbankbenutzers erfolgreich eingegeben.

Um zu testen, ob bei fehlendender Eingabe in einem Pflichtfeld der Schritt fehlschlägt, wird im negativen Test das Pflichtfeld Passwort leer gelassen.

Erwartetes Ergebnis:

Beim positiven Test wird erwartet, dass der Kreis in der *Statusbar* grün wird. Zusätzlich sollte die *Statusinformation* „Aktueller Status: OK“ anzeigen.

Beim negativen Test wird erwartet, dass der Kreis der *Statusbar* sich rot verfärbt und die *Statusinformation* anzeigt, dass alle Pflichtfelder ausgefüllt werden müssen.

6.2.2 Test der Statusfunktionen

Testziel:

In diesem Testvorgang soll getestet werden, ob die *Statusinformationen* erfolgreich an den Benutzer weitergegeben werden. Es geht dabei um die *Statusinformation*, die einen Text ausgibt und den Statuskreis, der sich rot, grün oder gelb verfärbt.

Vorbedingungen:

Für den positiven Test müssen der SQL-Server gestartet und eine Datenbank mit den benötigten Tabellen angelegt sein. Zusätzlich muss ein Benutzer mit passenden Rechten vorhanden sein.
Für den negativen Test muss die Verbindung zum SQL-Server unterbrochen werden.

Durchführung:

Für den positiven Test wird ein Administrator in der Datenbank angelegt.

Für den negativen Test wird ein Datenbankbenutzer angelegt, obwohl keine Verbindung zum SQL-Server besteht.

Erwartetes Ergebnis:

Erwartet wird beim positiven Test, dass sich der Kreis der *Statusbar* grün verfärbt. Außerdem soll die *Statusinformation* anzeigen, dass der aktuelle Status OK ist.

Beim negativen Test wird erwartet, dass sich der Kreis der *Statusbar* rot verfärbt und die *Statusinformation* ausgibt: „Fehler beim Eintragen in der Datenbank.“

6.2.3 Test der Navigation

Testziel:

In diesem Test sollen die Funktionen der Navigation überprüft werden.
Die Navigation bietet die Funktion, dass der nächste Schritt erst geöffnet werden kann, nachdem der aktuelle Schritt erfolgreich abgeschlossen wurde.

Durchführung

Für den positiven Test wird, nach einer erfolgreichen Durchführung des Schrittes, der Weiter-Button gedrückt. Anschließend wird der Zurück-Button gedrückt.

Für den negativen Test wird, nach einem fehlgeschlagenen Schritt, der Weiter-Button gedrückt.

Erwartetes Ergebnis:

Beim positiven Test wird erwartet, dass nach Drücken des Weiter-Buttons der nächste Schritt geöffnet wird. Nach Betätigen des Zurück-Buttons soll der erfolgreich bearbeitete Schritt wieder angezeigt werden. Die zuvor gemachten Eingaben sollen dabei erneut angezeigt werden.

Im negativen Test wird erwartet, dass der nächste Schritt nicht geöffnet wird. Stattdessen soll die Fehlermeldung „Bitte schließen Sie den Status ab.“ in der Statusinformation angezeigt werden. Der *Status* in der *Statusbar* soll sich nicht verändern. Er ist bereits rot, da der Schritt fehlgeschlagen ist.

6.1 Testergebnisse

Die Durchführung der JUNIT Tests war für alle Tests erfolgreich. Angezeigt wurde dieses Ergebnis durch den grünen Statusbalken der JUNIT Testumgebung.

Auch bei den Tests der Oberfläche sind alle Erwartungen eingetroffen.

7 Diskussion und Ausblick

Ziel dieser Bachelorarbeit war es, ein Konzept für einen Installer der Software RANDI2 zu entwickeln. Zusätzlich wurde das Konzept in Form einer Software realisiert.
Zur Entwicklung des Konzepts wurden die Aufgaben und Probleme erörtert, die bei der vorherigen, manuellen Installation auftraten. Zudem wurden verschiedene Technologien, die zur Installation und Konfiguration von RANDI2 benötigt werden, vorgestellt. Basierend auf diesen Kenntnissen wurde anschließend eine Anforderungsanalyse ausgeführt, um die Bedürfnisse des Anwenders bzgl. der Installation und Konfiguration zu identifizieren. Des Weiteren wurde ein Architekturmodell erstellt, mit dessen Hilfe die ermittelnden Anforderungen umgesetzt wurden. Um zu verifizieren, dass der so entstandene Installer die Anforderungen korrekt umsetzt, wurden verschiedene Tests ausgeführt und protokolliert.
Als Ergebnis dieser Bachelorarbeit steht ein benutzerfreundlicher Installer zur Verfügung, der den Anwender durch die komplette Installation und Konfiguration von RANDI2 leitet.

Der vorhandene Installer kann noch erweitert werden, sodass der Anwender auswählen kann, welcher JEE Application Server oder Servlet Container für den Betrieb von RANDI2 verwendet werden soll. Eine weitere Möglichkeit wäre, dass testen und verwenden anderer Datenbanksysteme wie z.B. HSQLDB.

8 Literaturverzeichnis

[1] Zoë Hoare. Randomisation: what, why and how?
Signigicance 2010; 7: 136-138.

[2] Schrimpf D, Plotnicki L, Pilz LR.: Web-based open source application for the randomization process in clinical trials: RANDI2.
Int J Clin Pharmacol Ther. 2010; 48: 465-467.

[3] Schichtenmodell zu RANDI2.
http://www.randi2.org 15. Dezember 2011

[4] Scott Chacon (Hrsg.): Pro Git.
Apress Verlag 2009

[5] Wikipedia.org: Git.
http://de.wikipedia.org/w/index.php?title=Git&oldid=98578130 19. Januar 2012

[6] Wikipedia.org: GitHub.
http://de.wikipedia.org/w/index.php?title=GitHub&oldid=96984550 11. Dezember 2011

[7] Kai Uwe Bachmann (Hrsg.): Maven 2 Eine Einführung.
Addison-Wesley Verlag 2009

[8] Markus Hiermer (Hrsg.): Autodesk Revit Architecture 2012 Grundlagen.
Verlagsgruppe Hüthig Jehle Rehm GmbH 1. Auflage 2011

[9] Marcus Deininger, Georg Faust und Thomas Kessel (Hrsg.): Java leicht gemacht.
Oldenbourg Wissenschaftverlag GmbH 2009

[10] Cornelia Heinisch, Frank Müller-Hofmann und Joachim Goll (Hrsg.): Java als erste Programmiersprache.
Vieweg + Teuber Verlag 2000, 6., überarbeitete Auflage 2011

9 Abbildungsverzeichnis

10 Eidesstattliche Erklärung

Ich erkläre hiermit an Eides Statt, dass ich die vorliegende Arbeit selbstständig und ohne Benutzung anderer, als der angegebenen Hilfsmittel angefertigt habe; die aus fremden Quellen (einschließlich elektronischer Quellen) direkt oder indirekt übernommenen Gedanken sind als solche kenntlich gemacht.

Die Arbeit wurde bisher weder im Inland noch im Ausland in gleicher Form oder ähnlicher Form einer anderen Prüfungsbehörde vorgelegt und ist auch noch nicht veröffentlicht.

Heilbronn, 06. Februar 2012

11 Anhang

11.1 Beschreibung der Aktivitäten

<table>
<tr><td>Aktivitäten Name</td><td colspan="3">Tomcat Installieren</td></tr>
<tr><td>Aktivitäten Beschreibung</td><td colspan="3">Der JEE Applikation Server Tomcat wird heruntergeladen und installiert.</td></tr>
<tr><td>Vorbedingungen</td><td colspan="3">- Sprache wurde ausgewählt
- System ist mit dem Internet verbunden</td></tr>
<tr><td rowspan="9">Abfolge der Ereignisse</td><td></td><td>Anwender</td><td>System</td></tr>
<tr><td>1</td><td>Klicke Button: Download</td><td></td></tr>
<tr><td>2</td><td></td><td>Öffne Tomcat Webseite</td></tr>
<tr><td>3</td><td>Lade die Tomcat Datei herunter</td><td></td></tr>
<tr><td>4</td><td>Entpacke die Tomcat Datei</td><td></td></tr>
<tr><td>5</td><td>Klicke Button: Öffnen</td><td></td></tr>
<tr><td>6</td><td></td><td>Öffne File Chooser Dialog</td></tr>
<tr><td>7</td><td>Wähle Tomcat Home Verzeichnis aus</td><td></td></tr>
<tr><td>8</td><td></td><td>Speicher den Pfad</td></tr>
<tr><td>Nachbedingungen</td><td colspan="3">- Tomcat wurde heruntergeladen
- Der Tomcat Home Pfad wurde gespeichert
- Der Status des Schrittes wird durch das System auf erfolgreich gesetzt</td></tr>
</table>

<table>
<tr><td>Aktivitäten Name</td><td colspan="5">Auswahl DBMS</td></tr>
<tr><td>Aktivitäten Beschreibung</td><td colspan="5">Der Benutzer wählt aus, ob er bereits ein MySQL oder PostgreSQL DBMS installiert hat. Wenn ja, gibt er die Zugangsdaten ein</td></tr>
<tr><td>Vorbedingungen</td><td colspan="5"></td></tr>
<tr><td rowspan="7">Abfolge der Ereignisse</td><td></td><td colspan="2">Anwender</td><td colspan="2">System</td></tr>
<tr><td>1</td><td colspan="2">Wähle zwischen:
- Ja, MySQL
- Ja, PostgreSQL
- Nein</td><td colspan="2"></td></tr>
<tr><td>2</td><td colspan="2"></td><td colspan="2">Speichere die Information</td></tr>
<tr><td>3</td><td>Auswahl Ja</td><td>Auswahl Nein</td><td>Auswahl Ja</td><td>Auswahl Nein</td></tr>
<tr><td>4</td><td></td><td></td><td>Gebe Eingabemaske frei</td><td>Sperre Eingabemaske</td></tr>
<tr><td>5</td><td>Gebe die Zugangsdaten zum SQL-Server ein</td><td></td><td></td><td></td></tr>
<tr><td>6</td><td></td><td></td><td>Fahre mit Schritt RANDI2 installieren fort.</td><td>Fahre fort mit XAMMP installieren</td></tr>
<tr><td>Nachbedingungen</td><td colspan="5">- Der Status des Schrittes wird durch das System auf erfolgreich gesetzt</td></tr>
</table>

<table>
<tr><td>Aktivitäten Name</td><td colspan="3">SQL-Server Installieren</td></tr>
<tr><td>Aktivitäten Beschreibung</td><td colspan="3">XAMPP wird heruntergeladen, installiert und gestartet.</td></tr>
<tr><td>Vorbedingungen</td><td colspan="3">- System ist mit dem Internet verbunden
- Keine andere Datenbank ist installiert</td></tr>
<tr><td rowspan="7">Abfolge der Ereignisse</td><td></td><td>Anwender</td><td>System</td></tr>
<tr><td>1</td><td>Klicke Button: Download</td><td></td></tr>
<tr><td>2</td><td></td><td>Öffne Apache Friends Webseite</td></tr>
<tr><td>3</td><td>Lade die XAMPP Datei herunter</td><td></td></tr>
<tr><td>4</td><td>Installiere XAMMP</td><td></td></tr>
<tr><td>5</td><td>Starte das Control Panel von XAMPP</td><td></td></tr>
<tr><td>6</td><td>Klicke Button: MySQL</td><td></td></tr>
<tr><td>Nachbedingungen</td><td colspan="3">- MySQL Datenbank wurde gestartet
- Der Status des Schrittes wird durch das System auf erfolgreich gesetzt</td></tr>
</table>

<table>
<tr><td>Aktivitäten Name</td><td colspan="3">Datenbank konfigurieren</td></tr>
<tr><td>Aktivitäten Beschreibung</td><td colspan="3">Die MySQL oder PostgreSQL Datenbank wird erstellt und ein Benutzer wird angelegt.</td></tr>
<tr><td>Vorbedingungen</td><td colspan="3">- Eine MySQL oder PostgreSQL Datenbank läuft auf dem System.
- Ein User mit folgenden Login Daten ist vorhanden (Bei XAMPP Standard):
Benutzername: root
Passwort:
- Der Benutzer hat ausgewählt, ob er eine MySQL oder PostgreSQL Datenbank verwendet</td></tr>
<tr><td rowspan="7">Abfolge der Eregnisse</td><td></td><td>Anwender</td><td>System</td></tr>
<tr><td>1</td><td>Mache folgende Eingaben:
- Datenbankserver
- Datenbankname
- Benutzername
- Passwort
- Wiederholte Eingabe des Passwortes</td><td></td></tr>
<tr><td>2</td><td>Klicke Button: Speichern</td><td></td></tr>
<tr><td>3</td><td></td><td>Speicher die Daten</td></tr>
<tr><td>4</td><td></td><td>Lege eine Datenbank mit dem gewählten Namen an</td></tr>
<tr><td>5</td><td></td><td>Lege einen Benutzer mit Benutzernamen und Passwort an</td></tr>
<tr><td>6</td><td></td><td>Gebe dem Benutzer alle Rechte für die Datenbank</td></tr>
<tr><td>Nachbedingungen</td><td colspan="3">- Die Datenbank und der Benutzer mit allen Rechten wurden angelegt.
- Der Status des Schrittes wird durch das System auf erfolgreich gesetzt</td></tr>
</table>

<table>
<tr><td>Aktivitäten Name</td><td colspan="3">RANDI2 einbinden</td></tr>
<tr><td>Aktivitäten Beschreibung</td><td colspan="3">RANDI2 wird heruntergeladen und ins Tomcat/webapps Verzeichnis kopiert</td></tr>
<tr><td>Vorbedingungen</td><td colspan="3">- System muss mit dem Internet verbunden sein
- Tomcat muss auf dem System installiert sein
- Der Pfad zum Tomcat Home Verzeichnis muss gesetzt sein</td></tr>
<tr><td rowspan="10">Abfolge der Ereignisse</td><td></td><td>Anwender</td><td>Installer</td></tr>
<tr><td>1</td><td>Klicke Button: Download</td><td></td></tr>
<tr><td>2</td><td></td><td>Öffne Webseite randi2.org</td></tr>
<tr><td>3</td><td>Lade RANDI2 herunter</td><td></td></tr>
<tr><td>4</td><td>Klicke Button: Öffnen</td><td></td></tr>
<tr><td>5</td><td></td><td>Öffne File Chooser Dialog</td></tr>
<tr><td>6</td><td>Wähle den Pfad zu RANDI2.war</td><td></td></tr>
<tr><td>7</td><td></td><td>Überprüfe den Pfad</td></tr>
<tr><td>8</td><td></td><td>Speicher den Pfad</td></tr>
<tr><td>9</td><td></td><td>Kopiere RANDI2.war nach Tomcat/webapps</td></tr>
<tr><td>Nachbedingungen</td><td colspan="3">- RANDI2 wurde heruntergeladen und ins Tomcat Verzeichnis kopiert
- Der Status des Schrittes wird durch das System auf erfolgreich gesetzt</td></tr>
</table>

Aktivitäten Name	Tomcat starten		
UsÖffne Tomcat Start Datei für Mace Case Beschreibung	Der Tomcat Server wird gestartet		
Vorbedingungen	- Tomcat muss installiert sein - Der Tomcat Pfad muss gesetzt sein - RANDI2 muss installiert sein		
Abfolge der Ereignisse		Anwender	System
	1	Klicke Button: Mac Start oder Win Start	
	2		Starte die Datei Tomcat/bin/startup.sh bzw. Tomcat/bin/startup.bat
Nachbedingungen	- Tomcat wurde gestartet - Die RANDI2.war Datei wurde von Tomcat entpackt		

<table>
<tr><td>Aktivitäten Name</td><td colspan="3">Datenbank initialisieren</td></tr>
<tr><td>Aktivitäten Beschreibung</td><td colspan="3">Die Tabellen in der Datenbank werden angelegt</td></tr>
<tr><td>Vorbedingungen</td><td colspan="3">- Datenbank muss gestartet sein
- Benutzer muss MySQL oder PostgreSQL ausgewählt haben
- Datenbank muss angelegt worden sein
- Benutzer muss angelegt worden sein
- Benutzer muss Rechte erhalten haben</td></tr>
<tr><td rowspan="6">Abfolge der Ereignisse</td><td></td><td>Anwender</td><td>System</td></tr>
<tr><td>1</td><td>Klicke Button: Öffnen</td><td></td></tr>
<tr><td>2</td><td></td><td>Öffne File Chooser</td></tr>
<tr><td>3</td><td>Wähle die passende init.sql Datei</td><td></td></tr>
<tr><td>4</td><td></td><td>Verbinde mit Server</td></tr>
<tr><td>5</td><td></td><td>Führe das SQL Skript aus</td></tr>
<tr><td>Nachbedingungen</td><td colspan="3">- Tabellen in der Datenbank wurden erstellt
- Der Status des Schrittes wird durch das System auf erfolgreich gesetzt</td></tr>
</table>

<table>
<tr><td>Aktivitäten Name</td><td colspan="3">JDBC Treiber installieren</td></tr>
<tr><td>Aktivitäten Beschreibung</td><td colspan="3">JDBC Treiber werden in Tomcat installiert</td></tr>
<tr><td>Vorbedingungen</td><td colspan="3">- Tomcat muss installiert sein
- Der Tomcat Pfad muss gesetzt sein
- Das System muss mit dem Internet verbunden sein
- MySQL oder PostgreSQL muss ausgewählt worden sein</td></tr>
<tr><td rowspan="9">Abfolge der Ereignisse</td><td></td><td>Anwender</td><td>System</td></tr>
<tr><td>1</td><td>Klicke Button: Download</td><td></td></tr>
<tr><td>2</td><td></td><td>Öffne MySQL oder PostgreSQl Seite</td></tr>
<tr><td>3</td><td>Lade passenden JDBC Treiber herunter</td><td></td></tr>
<tr><td>4</td><td>Klicke Button: Öffnen</td><td></td></tr>
<tr><td>5</td><td></td><td>Öffne File Chooser</td></tr>
<tr><td>6</td><td>Wähle die .jar Datei aus</td><td></td></tr>
<tr><td>7</td><td></td><td>Überprüfe Datei auf richtige Endung</td></tr>
<tr><td>8</td><td></td><td>Kopiere die Datei nach Tomcat/lib</td></tr>
<tr><td>Nachbedingungen</td><td colspan="3">- Der JDBC Treiber wurde in Tomcat installiert
- Der Status des Schrittes wird durch das System auf erfolgreich gesetzt</td></tr>
</table>

<table>
<tr><td>Aktivitäten Name</td><td colspan="3">Mailserver konfigurieren</td></tr>
<tr><td>Aktivitäten Beschreibung</td><td colspan="3">Java Mail API wird in Tomcat installiert</td></tr>
<tr><td>Vorbedingungen</td><td colspan="3">- Tomcat muss installiert sein
- Der Tomcat Pfad muss gesetzt sein
- Das System muss mit dem Internet verbunden sein</td></tr>
<tr><td rowspan="9">Abfolge der Ereignisse</td><td></td><td>Anwender</td><td>System</td></tr>
<tr><td>1</td><td>Klicke Button: Download</td><td></td></tr>
<tr><td>2</td><td></td><td>Öffne die Webseite von oracle.com</td></tr>
<tr><td>3</td><td>Lade passende Java Mail API herunter</td><td></td></tr>
<tr><td>4</td><td>Klicke Button: Öffnen</td><td></td></tr>
<tr><td>5</td><td></td><td>Öffne File Chooser</td></tr>
<tr><td>6</td><td>Wähle die .jar Datei aus</td><td></td></tr>
<tr><td>7</td><td></td><td>Überprüfe Datei auf richtige Endung</td></tr>
<tr><td>8</td><td></td><td>Kopiere die Datei nach Tomcat/lib</td></tr>
<tr><td>Nachbedingungen</td><td colspan="3">- Die JAVA Mail API wurde erfolgreich in Tomcat installiert
- Der Status des Schrittes wird durch das System auf erfolgreich gesetzt</td></tr>
</table>

<table>
<tr><td>Aktivitäten Name</td><td colspan="3">Java Activation Framework installieren</td></tr>
<tr><td>Aktivitäten Beschreibung</td><td colspan="3">Java Activation Framework wird in Tomcat installiert</td></tr>
<tr><td>Vorbedingungen</td><td colspan="3">- Tomcat muss installiert sein
- Der Tomcat Pfad muss gesetzt sein
- Das System muss mit dem Internet verbunden sein</td></tr>
<tr><td rowspan="9">Abfolge der Ereignisse</td><td></td><td>Anwender</td><td>System</td></tr>
<tr><td>1</td><td>Klicke Button: Download</td><td></td></tr>
<tr><td>2</td><td></td><td>Öffne die Webseite von oracle.com</td></tr>
<tr><td>3</td><td>Lade das passende Java Activation Framework herunter</td><td></td></tr>
<tr><td>4</td><td>Klicke Button: Öffnen</td><td></td></tr>
<tr><td>5</td><td></td><td>Öffne File Chooser</td></tr>
<tr><td>6</td><td>Wähle die .jar Datei aus</td><td></td></tr>
<tr><td>7</td><td></td><td>Überprüfe Datei auf richtige Endung</td></tr>
<tr><td>8</td><td></td><td>Kopiere die Datei nach Tomcat/lib</td></tr>
<tr><td>Nachbedingungen</td><td colspan="3">- Das Java Activation Framework wurde erfolgreich in Tomcat installiert
- Der Status des Schrittes wird durch das System auf erfolgreich gesetzt</td></tr>
</table>

<table>
<tr><td>Aktivitäten Name</td><td colspan="3">Context.xml editieren</td></tr>
<tr><td>Aktivitäten Beschreibung</td><td colspan="3">Die Context.xml aus dem Tomcat/conf Verzeichnis wird angepasst. Es werden die JNDI und die Mail Server Daten eingegeben</td></tr>
<tr><td>Vorbedingungen</td><td colspan="3">- Datenbank Daten müssen eingegeben worden sein
- Tomcat muss installiert sein
- Der Tomcat Pfad muss gesetzt sein</td></tr>
<tr><td rowspan="7">Abfolge der Ereignisse</td><td></td><td>Anwender</td><td>System</td></tr>
<tr><td>1</td><td>Geben die fehlenden Daten zum Mail Server ein (Mailserver, Benutzername, Passwort und Wiederholung des Passwortes)</td><td></td></tr>
<tr><td>2</td><td>Klicke Button: Speichern</td><td></td></tr>
<tr><td>3</td><td></td><td>Überprüfe die Daten auf Korrektheit</td></tr>
<tr><td>4</td><td></td><td>Öffne die Tomcat/conf/ Context.xml</td></tr>
<tr><td>5</td><td></td><td>Füge die neuen Daten ein</td></tr>
<tr><td>6</td><td></td><td>Überschreibe die Datei Tomcat/lib/Context.xml</td></tr>
<tr><td>Nachbedingungen</td><td colspan="3">- Die Context.xml im Tomcat Verzeichnis wurde mit den neuen Daten überschrieben.
- Der Status des Schrittes wird durch das System auf erfolgreich gesetzt</td></tr>
</table>

Aktivitäten Name	Administrator konfigurieren		
Aktivitäten Beschreibung	Der Benutzer Administrator wird in der Datenbank mit den Konfigurationen des Anwenders aktualisiert.		
Vorbedingungen	- Datenbank muss gestartet sein - Es muss zwischen MySQL und PostgreSQL entschieden worden sein		
Abfolge der Ereignisse		Anwender	System
	1	Gebe die Daten akademischer Grad, Vorname, Nachname, Benutzername und Geschlecht ein	
	2	Klicke Button: Speichern	
	3		Überprüfe die Daten auf Korrektheit
	4		Speicher die Daten in ein Objekt der Klasse Administrator
	5	Klicke Button: Weiter	
	6	Gebe die Daten E-Mail, Faxnummer, Mobilfunknummer, Telefonnummer, Passwort und wiederholtes Passwort ein.	
	7	Klicke Button: Speichern	
	8		Überprüfe die Daten auf Korrektheit
	9		Speicher die Daten in der Datenbank in den Tabellen Person und Login
Nachbedingungen	- Der Benutzer wurde in der Datenbank aktualisiert - Der Status dieses und des letzten Schrittes wird durch das System auf erfolgreich gesetzt		

Aktivitäten Name	Zentrums und Ansprechpartner konfigurieren		
Aktivitäten Beschreibung	Das Zentrum und der Ansprechpartner werden aktualisiert		
Vorbedingungen	- Datenbank muss gestartet sein - Es muss zwischen MySQL und PostgreSQL entschieden worden sein		
Abfolge der Ereignisse		Anwender	System
	1	Gebe die Daten Name, Land, Stadt, Postleitzahl und Straße ein	
	2	Klicke Button: Speichern	
	3		Überprüfe die Daten auf Korrektheit
	4		Speicher die Daten in ein Objekt der Klasse Center
	5	Klicke Button: Weiter	
	4	Gebe die Daten akademischer Grad, Vorname, Nachname, E-Mail und Geschlecht ein	
	5	Klicke Button: Speichern	
	6		Überprüfe die Daten auf Korrektheit
	7		Speicher die Daten in einem Objekt der Klasse Ansprechpartner
	8	Gebe die Daten Telefonnummer, Mobilfunknummer, Faxnummer, Passwort und wiederholtes Passwort ein.	
	9	Klicke Button: Speichern	

	10		Überprüfe die Daten auf Korrektheit
	11		Aktualisiere die Tabellen TrialSite und Person in der Datenbank
Nachbedingungen	- Der Ansprechpartner wurde in der Datenbank aktualisiert - Die Angaben zum Zentrum wurden in der Datenbank aktualisiert - Der Status dieses und der Status, der letzten beiden Schritte, wird durch das System auf erfolgreich gesetzt		

<table>
<tr><td>Aktivitäten Name</td><td colspan="5">Selbstauthentifizierung am Zentrum</td></tr>
<tr><td>Aktivitäten Beschreibung</td><td colspan="5">- Es wird festgelegt, ob sich Benutzer selbst am Zentrum authentifizieren können
- Wenn ja, wird ein Passwort festgelegt</td></tr>
<tr><td>Vorbedingungen</td><td colspan="5">- Datenbank muss gestartet sein
- Es muss zwischen MySQL und PostgreSQL entschieden worden sein</td></tr>
<tr><td rowspan="9">Abfolge der Ereignisse</td><td></td><td colspan="2">Anwender</td><td colspan="2">System</td></tr>
<tr><td>1</td><td colspan="2">Dürfen sich neue Benutzer selbst registrieren?</td><td colspan="2"></td></tr>
<tr><td>2</td><td>Ja</td><td>Nein</td><td>Ja</td><td>Nein</td></tr>
<tr><td>3</td><td></td><td></td><td>Zeige Passwortfeld der an</td><td></td></tr>
<tr><td>4</td><td>Gebe Passwort ein</td><td></td><td></td><td></td></tr>
<tr><td>5</td><td>Wiederhole Eingabe</td><td></td><td></td><td></td></tr>
<tr><td>6</td><td colspan="2"></td><td>Überprüfe Eingaben</td><td></td></tr>
<tr><td>7</td><td colspan="2"></td><td>Speicher Daten in der Datenbank</td><td></td></tr>
<tr><td>8</td><td colspan="2"></td><td>Setzte Variable in randi2.properties</td><td>Setzte Variable in randi2.properties</td></tr>
<tr><td>Nachbedingungen</td><td colspan="5">- Es wurde festgelegt, ob sich Benutzer selbst authentifizieren dürfen.
- Der Status des Schrittes wird durch das System auf erfolgreich gesetzt</td></tr>
</table>

<table>
<tr><td>Aktivitäten Name</td><td colspan="3">Logo und Webseite konfigurieren</td></tr>
<tr><td>Aktivitäten Beschreibung</td><td colspan="3">Es werden weitere Konfigurationen durchgeführt:
- Anpassen des Logo
- Eintragen der Webseite des Unternehmens</td></tr>
<tr><td>Vorbedingungen</td><td colspan="3">- Tomcat muss installiert sein
- Der Tomcat Pfad muss gesetzt sein</td></tr>
<tr><td rowspan="11">Abfolge der Ereignisse</td><td></td><td>Anwender</td><td>System</td></tr>
<tr><td>1</td><td>Klicke Button: Öffnen</td><td></td></tr>
<tr><td>2</td><td></td><td>Öffne FileChooser</td></tr>
<tr><td>3</td><td>Wähle ein Logo aus</td><td></td></tr>
<tr><td>4</td><td>Gebe die Daten zur Webseite ein</td><td></td></tr>
<tr><td>5</td><td>Klicke Button: Speichern</td><td></td></tr>
<tr><td>6</td><td></td><td>Überprüfe Eingaben</td></tr>
<tr><td>7</td><td></td><td>Kopiere das Logo nach Tomcat/webapps/RANDI2/ resources/ hostingInstLogo.png</td></tr>
<tr><td>8</td><td></td><td>Öffne die Datei Tomcat/ webapps/RANDI2/ RANDI2.properties</td></tr>
<tr><td>9</td><td></td><td>Passe die Daten zur Webseite an</td></tr>
<tr><td>10</td><td></td><td>Überschreibe die Datei Tomcat/webapps/RANDI2/ RANDI2.properties</td></tr>
<tr><td>Nachbedingungen</td><td colspan="3">- Das Logo und die Webseite werden in Tomcat angepasst.
- Der Status des Schrittes wird durch das System auf erfolgreich gesetzt</td></tr>
</table>

Aktivitäten Name	Informationsserver konfigurieren		
Aktivitäten Beschreibung	Es werden Informationen zum Server in der Tomcat Installation von RANDI2 angepasst		
Vorbedingungen	- Tomcat muss installiert sein - Der Tomcat Pfad muss gesetzt sein - RANDI2 muss installiert sein		
Abfolge der Ereignisse		Anwender	System
	1	Gebe die Daten E-Mail Absender, Webseite und Gruppe an	
	2	Klicke Button: Speichern	
	3		Überprüfe die Daten auf Korrektheit
	4		Öffne die Datei webapps/ RANDI2/WEB-INF/classes/ META-INF/ configuration.properties
	5		Passe die Datei durch die eingegebenen Werte an.
	6		Überschreibe die Datei webapps/RANDI2/WEB-INF/classes/META-INF/ configuration.properties
Nachbedingungen	- Die Datei configuration.properties wurde erfolgreich angepasst - Der Status des Schrittes wird durch das System auf erfolgreich gesetzt		

Aktivitäten Name	Disclaimer editieren		
Aktivitäten Beschreibung	Ein deutscher und ein englischer Disclaimer werden eingegeben		
Vorbedingungen	- Tomcat muss installiert sein - Der Tomcat Pfad muss gesetzt sein - RANDI2 muss installiert sein		
Abfolge der Ereignisse		Anwender	System
	1	Gebe einen deutschen und einen englischen Disclaimer ein	
	2	Klicke Button: Speichern	
	3		Überprüfe die Eingabe
	4		Öffne die Datei webapps/ RANDI2/WEB-INF/classes/ de/randi2/jsf/i18n/ labels_de_DE.properties
	5		Ersetze den Text des alten Disclaimers mit dem des neuen
	6		Überschreibe die Datei webapps/RANDI2/WEB-INF/classes/de/randi2/jsf/i18n/ labels_de_DE.properties
	7		Öffne die Datei webapps/ RANDI2/WEB-INF/classes/ de/randi2/jsf/i18n/ labels_en_US.properties
	8		Ersetze den Text des alten Disclaimers mit dem des neuen
	9		Überschreibe die Datei webapps/RANDI2/WEB-INF/classes/de/randi2/jsf/i18n/ labels_en_US.properties

Nachbedingungen	- Der deutsche und englische Disclaimer wurde in der jeweiligen Sprachdatei gesetzt. - Der Status des Schrittes wird durch das System auf erfolgreich gesetzt

Aktivitäten Name	Informationstext editieren		
Aktivitäten Beschreibung	Ein deutscher und ein englischer Informationstext werden eingegeben		
Vorbedingungen	- Tomcat muss installiert sein - Der Tomcat Pfad muss gesetzt sein - RANDI2 muss installiert sein		
Abfolge der Ereignisse		Anwender	System
	1	Gebe einen deutschen und einen englischen Informationstext ein	
	2	Klicke Button: Speichern	
	3		Öffne die Datei webapps/ RANDI2/WEB-INF/classes/ de/randi2/jsf/i18n/ labels_de_DE.properties
	4		Ersetze den Text des alten deutschen Informationstextes mit dem des neuen deutschen Informationstextes
	5		Überschreibe die Datei webapps/RANDI2/WEB-INF/classes/de/randi2/jsf/i18n/ labels_de_DE.properties
	6		Öffne die Datei webapps/ RANDI2/WEB-INF/classes/ de/randi2/jsf/i18n/ labels_en_US.properties
	7		Ersetze den Text des alten englischen Informationstextes mit dem des neuen englischen Informationstextes
	8		Überschreibe die Datei webapps/RANDI2/WEB-INF/classes/de/randi2/jsf/i18n/ labels_en_US.properties

Nachbedingungen	- Der deutsche und englische Informationstext wurde in der jeweiligen Sprachdatei gesetzt. - Der Status des Schrittes wird durch das System auf erfolgreich gesetzt