

Architekturdokumentation

<Mängel-Manager>

erstellt von:

Luca Kündig

Sandro Ritz

Mike Monticoli

Mike Iten

Cihan Demir

<Gruppe B>

Template Revision: 6.0 DE (Release Candidate)

19. März 2012

We acknowledge that this document uses material from the arc 42 architecture template, <http://www.arc42.de>. Created by Dr. Peter Hruschka & Dr. Gernot Starke.
For additional contributors see arc42.de/about/contributors.html



INHALTSVERZEICHNIS

1. EINFÜHRUNG UND ZIELE	5
1.1 AUFGABENSTELLUNG	6
1.2 QUALITÄTSZIELE.....	9
1.3 STAKEHOLDER.....	10
2. RANDBEDINGUNGEN	10
2.1 TECHNISCHE RANDBEDINGUNGEN	10
2.2 ORGANISATORISCHE RANDBEDINGUNGEN.....	11
2.3 KONVENTIONEN	12
3. KONTEXTABGRENZUNG.....	12
3.1 FACHLICHER KONTEXT	12
4. LÖSUNGSSTRATEGIE	13
5. BAUSTEINSICHT	14
5.1 MÄNGEL-MANAGER EBENE 0	14
5.1.1 Model.....	14
5.1.2 Client Intern (Black Box-Beschreibung).....	15
5.1.3 Client Extern (Black Box-Beschreibung).....	15
5.1.4 Webservice (Black Box-Beschreibung)	15
5.1.5 RMI (Black Box-Beschreibung)	15
5.1.6 Business (BlackBox-Beschreibung)	15
5.1.7 Persister (BlackBox-Beschreibung)	15
5.2 MÄNGEL-MANAGER EBENE 1	15
5.2.1 Client Intern LVL1 (Whitebox-Beschreibung).....	15
5.2.2 Client Extern LVL1 (Whitebox-Beschreibung).....	16
5.2.3 Webservice LVL1 (Whitebox-Beschreibung)	16
5.2.4 RMI LVL1 (Whitebox-Beschreibung)	16
5.2.5 Business LVL1 (Whitebox-Beschreibung)	16
5.2.6 Persister LVL1 (Whitebox-Beschreibung)	16
5.3 MÄNGEL-MANAGER EBENE 2	16
5.3.1 Client Intern & Extern LVL2 (Whitebox-Beschreibung).....	16
6. LAUFZEITSICHT	17
7. VERTEILUNGSSICHT	18
8. KONZEPTE	19
8.1 TYPISCHE MUSTER STRUKTUREN.....	19
8.2 TYPISCHE ABLÄUFE.....	20
8.3 PERSISTENZ.....	20
8.4 BENUTZUNGSOBERFLÄCHE	21
8.5 ERGONOMIE.....	23
8.6 ABLAUFSTEUERUNG	23
8.7 TRANSAKTIONSBEHANDLUNG	23
8.8 SESSIONBEHANDLUNG	23
8.9 SICHERHEIT	23
8.10 KOMMUNIKATION UND INTEGRATION MIT ANDEREN IT-SYSTEMEN	24
8.11 VERTEILUNG	24
8.12 PLAUSIBILISIERUNG UND VALIDIERUNG	24
8.13 AUSNAHME-/FEHLERBEHANDLUNG	24
8.14 MANAGEMENT DES SYSTEMS & ADMINISTRIERBARKEIT	24
8.15 LOGGING, PROTOKOLLIERUNG, TRACING	25
8.16 GESCHÄFTSREGELN	25

8.17	KONFIGURIERBARKEIT.....	25
8.18	PARALLELISIERUNG UND THREADING.....	25
8.19	INTERNATIONALISIERUNG.....	25
8.20	MIGRATION.....	25
8.21	TESTBARKEIT.....	25
8.22	SKALIERUNG, CLUSTERING.....	26
8.23	HOCHVERFÜGBARKEIT.....	26
9.	ENTWURFSENTSCHEIDUNGEN	27
9.1	ENTWURFSENTSCHEIDUNG 1	27
9.1.1	<i>Fragestellung.....</i>	27
9.1.2	<i>Rahmenbedingungen.....</i>	27
9.1.3	<i>Annahmen.....</i>	27
9.1.4	<i>Betrachtete Alternativen.....</i>	27
9.1.5	<i>Entscheidung.....</i>	27
9.2	ENTWURFSENTSCHEIDUNG 2	27
9.2.1	<i>Fragestellung.....</i>	27
9.2.2	<i>Rahmenbedingungen.....</i>	27
9.2.3	<i>Annahmen.....</i>	27
9.2.4	<i>Betrachtete Alternativen.....</i>	27
9.2.5	<i>Entscheidung.....</i>	28
9.3	ENTWURFSENTSCHEIDUNG 3	28
9.3.1	<i>Fragestellung.....</i>	28
9.3.2	<i>Rahmenbedingungen.....</i>	28
9.3.3	<i>Annahmen.....</i>	28
9.3.4	<i>Betrachtete Alternativen.....</i>	28
9.3.5	<i>Entscheidung.....</i>	28
10.	QUALITÄTSSZENARIEN	29
10.1	QUALITÄTSBAUM.....	29
10.2	BEWERTUNGSSZENARIEN.....	30
11.	RISIKEN	30
12.	GLOSSAR	32
13.	WEITERE ARC42 DOKUMENTATIONEN.....	33
14.	REQUIREMENTS DOKUMENTATION	33
15.	USECASE DOKUMENTATION.....	34
15.1	USECASE001	34
15.1.1	<i>UseCase001 Beschreibung</i>	34
15.1.2	<i>UseCase001 Visualisierung.....</i>	35
15.1.3	<i>UseCase001 Aktivitätsdiagramm.....</i>	36
15.2	USECASE002	40
15.2.1	<i>UseCase002 Beschreibung</i>	40
15.2.2	<i>UseCase002 Visualisierung.....</i>	41
15.2.3	<i>UseCase002 Aktivitätsdiagramm.....</i>	42
15.3	USECASE003	45
15.3.1	<i>UseCase003 Beschreibung</i>	45
15.3.2	<i>UseCase003 Visualisierung.....</i>	46
15.3.3	<i>UseCase003 Aktivitätsdiagramm.....</i>	47
15.4	USECASE004	48
15.4.1	<i>UseCase004 Beschreibung</i>	48
15.4.2	<i>UseCase004 Visualisierung.....</i>	49
15.4.3	<i>UseCase004 Aktivitätsdiagramm.....</i>	50
15.5	USECASE005	54
15.5.1	<i>UseCase005 Beschreibung</i>	54
15.5.2	<i>UseCase005 Visualisierung.....</i>	55

15.5.3	UseCase005 Aktivitätsdiagramm	56
15.6	USECASE006	58
15.6.1	UseCase006 Beschreibung	58
15.6.2	UseCase006 Visualisierung	59
15.6.3	UseCase006 Aktivitätsdiagramm	60
15.7	USECASE007	61
15.7.1	UseCase007 Beschreibung	61
15.7.2	UseCase007 Visualisierung	62
15.7.3	UseCase007 Aktivitätsdiagramm	63
15.8	USECASE008	64
15.8.1	UseCase008 Beschreibung	64
15.8.2	UseCase008 Visualisierung	65
15.8.3	UseCase008 Aktivitätsdiagramm	66
16.	KLASSENDIAGRAMME	67
16.1	KLASSENDIAGRAMM MODELS	67
16.2	KLASSENDIAGRAMM PERSISTER	68
16.3	KLASSENDIAGRAMM BUSINESS	69
16.4	KLASSENDIAGRAMM RMI	70
16.5	KLASSENDIAGRAMM CLIENT-INTERN (RMI)	71
16.6	KLASSENDIAGRAMM WEBSERVICE	72
16.7	KLASSENDIAGRAMM EXTERNER CLIENT (WEBCIENT)	73
17.	DEYPLOMENT-INFOS	74
17.1	POSTGRES UND DATENBANK:	74
17.2	STARTEN RMI-SERVER:	74
17.3	RMI-CLIENT STARTEN:	75
17.4	WEBSERVICE STARTEN:	75
17.5	WEB CLIENT STARTEN:	75
18.	TDD UND JUNIT	75
18.1	TDD	75
18.2	TEST-FILES	75
18.3	JUNIT	76
19.	FUNKTIONALE TEST'S	76
20.	DB - DOKUMENTATION	78
21.	BEITRÄGE PRO PROJEKTMITGLIED	79
21.1	LUCA KÜNDIG	79
21.2	SANDRO RITZ	82
21.3	MIKE ITEN	86
21.4	MIKE MONTICOLI	88
21.5	CIHAN DEMIR	91
22.	WEITERE DOKUMENTATIONEN	95
22.1	INDIVIDUELLE MANAGEMENT SUMMARYS	95
22.1.1	Luca Kündig	95
22.1.2	Sandro Ritz	96
22.1.3	Mike Iten	97
22.1.4	Mike Monticoli	98
22.1.5	Cihan Demir	99
22.2	AUSSERGEWÖHNLICHES	100
22.2.1	Weitere Diagramme und Dokumente	100
22.2.2	Projektleiter verlässt das Team	100
23.	SOURCE-CODE VON SELBER BESCHRIEBENEM CODE	101

1. Einführung und Ziele

Die „W&W Architekten GmbH“ besteht seit über zehn Jahren und beschäftigt bereits mehr als 30 Mitarbeiter. Die Generalunternehmung (im weiteren Text GU) „W&W Architekten GmbH“ ist beim Bau von Ein- und Mehrfamilienhäusern tätig. Es werden Bauprojekte realisiert, wobei die GU die Zentralfigur ist. Die GU wird ausschliesslich vom Bauherrn informiert und diese wiederum verständigt weitere Firmen (Subunternehmen) mit den noch zu erledigenden Arbeiten (z.B. Elektriker, Maler, Küchenbauer, Sanitär, Gartenbauer etc.).

IT Unterstützung

Momentan haben die GU Mitarbeiter, welche direkt mit der Bautätigkeit zu tun haben, eine eher spärliche IT Umgebung. Die Bauleiter beschwerten sich, dass die Übersicht mit der Zunahme von Arbeitsvolumen nachlässt. Das Risiko, dass essentielle Informationen in Vergessenheit geraten steigt kontinuierlich. Aufgrund dieses Risikos muss nach einer Lösung gesucht werden, welche als Unterstützung der Bauleiter dienen soll um die anfallenden Kontrollen effizient und zuverlässig durchzuführen.

Ablauf

Wenn ein Bauleiter auf einer Baustelle eine Kontrolle durchführt, kontrolliert er, ob die erledigten Arbeiten korrekt ausgeführt wurden oder nicht. Sollte er dabei auf Fehler bei den ausgeführten Arbeiten stossen, werden diese als Mängel erfasst. Bei jedem Mangel wird erfasst,

- um was für einen Mangel es sich handelt (eine kurze Beschreibung),
- welches Subunternehmen für die Arbeit und somit für Behebung des Mangels zuständig ist,
- wann der Mangel festgestellt wurde (Datum) und
- bis wann der Mangel behoben werden sollte (Datum).

Die Angabe, bis wann der Mangel behoben werden muss, ist sehr wichtig, da andere Arbeiten unter Umständen erst nach der Behebung des Mangels ausgeführt werden können.

Die Erfassung von Mängel wird auf dem Papier gemacht und am Abend, wenn der Bauleiter ins Büro zurückkehrt, in eine Excel-Arbeitsmappe eingetragen (pro Projekt wird eine Excel-Arbeitsmappe angelegt). Anschliessend wird ein PDF generiert und um den Aufwand im Rahmen zu halten, an alle betroffene Subunternehmen gesendet. Die Rede ist von einem Mängelrapport (im weiteren Text MR). Das kommt bei einigen Subunternehmen nicht gut an. Sie halten es für unangebracht, dass andere Subunternehmen den Einblick in Ihre bemängelten Arbeiten bekommen. Die Bauleiter weigern sich aber weiterhin, für jedes betroffene Subunternehmen ein separates MR zu machen. Sie begründen es mit dem unverhältnismässigen Aufwand den sie betreiben müssten.



Wenn ein Subunternehmen (im weiteren Text SU) den MR bekommt, muss für die bemängelte Arbeit zuständige Person in jenem SU darüber informiert werden. Idealerweise per Telefon (in dringenden Fällen) oder am nächsten Morgen, wenn alle Mitarbeiter des SU wieder in der Werkstatt sind. Daraufhin kann der zuständige Mitarbeiter des SU reagieren:

- Es kann sein, dass es schlicht und einfach ein Fehler ist, der behoben wird.
- Es kann auch sein, dass es für diesen Fehler nachvollziehbare Gründe gibt und auf die Behebung des vermeintlichen Fehlers evtl. verzichtet werden kann.
- Es kann auch sein, dass der vermeintliche Fehler gar kein Fehler ist und die bemängelte Arbeit aus bestimmten Gründen explizit so ausgeführt wurde.

Wenn dies geklärt ist, wird der Bauleiter vom GU entsprechend informiert und je nach dem, was der Grund für den Mangel ist, kann sein, dass es eine weitere Klärungsrunde geben muss usw. Das SU sendet an den Bauleiter ein Email, in dem alles was nötig ist, angegeben wird. Beispielsweise könnte dies wie folgt aussehen:

- Der Mangel Nr. X wird wie von Ihnen gewünscht bis zum angegebenen Datum behoben.
- Der Mangel Nr. Y kann nicht so behoben werden, wie Sie es gewünscht haben. Dazu braucht es eine Abklärung vor Ort. Bitte geben Sie uns an, wann Sie wieder vor Ort sein werden.
- Der Mangel Z ist gar kein Mangel. Die Ausführung wurde so explizit von Ihrem Chef-Architekt (z.B. Hr. Max Mustermann) gewünscht. Bitte klären Sie es mit Herrn Mustermann ab.

Wenn ein Mangel definitiv behoben ist, wird der Bauleiter benachrichtigt, damit er es noch abschliessend kontrollieren kann. Nach der abschliessenden Kontrolle wird der Mangel von Bauleiter als erledigt markiert, solange die Behebung zufriedenstellend war. Sollte das nicht der Fall sein, wendet sich der Bauleiter erneut an das zuständige SU usw. Dieser Vorgang kann bei Bedarf mehrfach wiederholt werden. Am Schluss müssen alle Mängel behoben und in der entsprechenden Excel-Arbeitsmappe als solche markiert sein.

Die Kommunikation in diesem Kontext ist etwas umständlich und bis ein Fall geklärt wird, kann unter Umständen lange dauern, da mehrere "Runden" benötigt werden. Um das Problem zu entschärfen, soll eine Applikation entwickelt werden, welche das Management von Baumängeln in einer einfacheren Art und Weise ermöglichen soll.

1.1 Aufgabenstellung

Nachfolgend werden die Anforderungen an die Applikation so gut wie zurzeit möglich beschrieben. Mit weiteren Ergänzungen muss gerechnet werden, Verbesserungsvorschläge sind willkommen.



Verwaltung von Stammdaten

Die zu erstellende Applikation muss mindestens folgendes ermöglichen (Erweiterungen sind erlaubt):

- a) die Verwaltung aller nötigen Daten von Subunternehmen, welche für die "W&W Architekten GmbH" im Auftragsverhältnis Arbeiten ausführen.
- b) die Verwaltung aller nötigen Daten von Bauherren, für welche ein Bau ausgeführt bzw. ein Projekt realisiert wird. Dabei ist zu berücksichtigen, dass in Rolle des Bauherrn sowohl eine Person als auch eine Firma auftreten kann.
- c) die Verwaltung aller Bauprojekte, welche von der GU realisiert werden. Für jedes Bauprojekt muss es schnell ersichtlich sein:
 - um was für ein Projekt es sich handelt (Einfamilienhaus, Mehrfamilienhaus, Wohnung, Garage etc.),
 - um was für eine Arbeit es sich handelt (Neu- oder Umbau, Renovation, Teil-Renovation),
 - in welcher Zeitperiode das Projekt realisiert werden soll (Start / Ende),
 - an welcher Adresse sich das neugebaute / umgebaute Objekt befindet,
 - wer der Bauherr ist,
 - welcher Bauleiter für das Projekt bzw. Bauobjekt seitens GU zu welcher Zeit zuständig ist,
 - welche Subunternehmen in dem Projekt involviert sind und
 - wer zu welcher Zeit die Ansprechperson für das jeweilige Subunternehmen in den einzelnen Projekten ist.

Die obige Aufzählung ist nicht abschliessend und kann bzw. soll im Bedarfsfall ergänzt werden.

Verwaltung von Mängeldaten

Die zu erstellende Applikation muss dem Bauleiter ermöglichen, die während einer Kontrolle bzw. einer Besichtigung der Baustelle gefundenen Mängel vor Ort zu erfassen und die erfassten Daten direkt auf dem Server der GU zu speichern. Das heisst, dass die Daten zum Server übertragen und da von einer dafür zuständigen Komponente in Empfang genommen und weiter verarbeitet werden.



Benutzerschnittstelle (User Interface)

Die Applikation muss drei Benutzerschnittstellen zur Verfügung stellen:

- 1) Die Benutzerschnittstelle für BackOffice (BO-UI) wird im Intranet von administrativen Mitarbeitern benutzt und stellt einem Benutzer alle Funktionalitäten zur Verfügung, welche für die Verwaltung von Daten (Stammdaten und Mängeldaten) benötigt werden.
- 2) Die Schnittstelle für Bauleiter (BL-UI) wird im Extranet von Bauleiter benutzt. Sie soll einem Bauleiter den Zugriff auf alle für ihn relevanten Daten zu einem beliebigen Zeitpunkt ermöglichen. Des Weiteren soll sie auch die Erfassung von während einer Kontrolle gefundenen Mängel vor Ort ermöglichen. Alle Bauleiter werden mit Notebooks oder Tablets ausgerüstet. Ausserdem wird dafür gesorgt, dass alle Bauleiter immer online sind und auf die IT Infrastruktur der GU (Website der GU, Server etc.) jederzeit zugreifen können.
- 3) Die Subunternehmen-Schnittstelle (SU-UI) soll den Zugriff auf die für ein Subunternehmen relevanten Daten ermöglichen. Dazu gehören die eigenen Stammdaten und die für das Subunternehmen bestimmten Mängeldaten.

Zugang und Datenablage

Der Zugriff auf die Applikation und Daten muss in allen Fällen durch ein entsprechendes Anmelde-Verfahren geschützt werden. Die Applikation wird einerseits von der GU Mitarbeiter (BackOffice und Bauleiter) und andererseits von den Mitarbeiter von Subunternehmen benutzt. Dabei muss auch sichergestellt werden, dass die unterschiedlichen Benutzer nur auf jene Daten zugreifen können, für die sie berechtigt sind. Die Benutzerschnittstelle für Subunternehmen (SU-UI) wird, sobald verfügbar, im Downloadbereich der GU allen Subunternehmen zum Downloaden zur Verfügung gestellt.

Die Verwaltung der anfallenden Daten muss mit einem zuverlässigen DBMS (z.B. PostgreSQL, MySQL, Oracle-DB, SQL-Server etc.) realisiert werden. Dabei muss sichergestellt werden, dass der verwendete DBMS problemlos und ohne zusätzliche Aufwände ausgetauscht werden kann (Bindung an ein bestimmtes DBMS ist nicht erlaubt).

Diverses

Nachfolgend werden noch weitere Anforderungen aufgeführt:

- Für ein Projekt (Objekt) ist seitens GU i.d.R. ein Bauleiter zuständig. Es kann aber sein, dass ein Projekt vorübergehend von einem anderen Bauleiter betreut wird (z.B. Ferienabwesenheit). Es muss aber auch möglich sein, dass ein Projekt von einem an-



deren Bauleiter vollständig übernommen wird (z.B. wenn ein Bauleiter die GU verlässt oder andere Projekte übernimmt). Sie müssen beim Bauleiter also eine Art "Buch führen", wann welcher Bauleiter welche Mängel erfasst hat und zuständig war.

- Ein Subunternehmen kann in mehreren Projekten Aufträge übernehmen. Für jedes Projekt, in dem das Subunternehmen mitarbeitet, muss eine Ansprechperson seitens Subunternehmen bestimmt werden. Je nach Grösse des Subunternehmens und Umfang der auszuführenden Arbeiten kann eine Ansprechperson für alle Projekte oder auch unterschiedliche Ansprechpersonen für unterschiedliche Projekte bestimmt werden.
- Ein Bauleiter der GU darf nur jene Daten verwalten können, die sich auf ihm zugeordnete Projekte beziehen, das heisst, von ihm geleitet werden. Die Einsicht in Projekte, die von andern Bauleiter geleitet werden, darf ein Bauleiter nicht haben.
- Die Zuweisung, welcher Bauleiter welches Projekt leitet, wird durch die Administration des GU vorgenommen (Sachbearbeiter / Administrator).
- Seitens Subunternehmen ist es auch so, dass eine Ansprechperson nur jene Projektdaten sehen darf, für die sie zuständig ist. Die Zuweisung einer Ansprechperson einem bestimmten Projekt wird durch den Administrator (Sachbearbeiter) des Subunternehmens vorgenommen.

1.2 Qualitätsziele

Qualitätsmerkmal	Ziel
Änderbarkeit	Es muss sichergestellt werden, dass der verwendete DBMS (Datenbankmanagementsystem) problemlos und ohne zusätzliche Aufwände ausgetauscht werden kann.
Benutzbarkeit	Es ist einfach gestaltet und einfach zu bedienen.
Effizienz	Das Programm soll zu jeder Zeit, ungeachtet des Projektumfangs, schnell laufen.
Sicherheit	Die Projektdaten sollen abgesichert auf einem Server gespeichert sein, welcher gut geschützt ist.
Zuverlässigkeit	Der Mängel-Manager muss auf jede mögliche Fehlerquelle korrekt reagieren.
Betreibbarkeit	Der Mängel-Manager soll kostengünstig und ohne viel Aufwand zu betreiben sein.



1.3 Stakeholder

Beteiligte	Rolle	Beschreibung
Generalunternehmen	BackOffice	Das BackOffice des GU ist sozusagen die Zentralstelle wenn es um den Mängel-Manager geht dementsprechend verfügt Sie auch über alle Funktionalitäten des Mängel-Managers. Sie kann Stammdaten und Mängeldaten hinzufügen, bearbeiten oder auch löschen. Mit Stammdaten sind aktuelle Projekte, Login Daten (Generalunternehmen Bauleiter, Subunternehmen BackOffice) gemeint.
	Bauleiter	Der Bauleiter ist für Projekte zuständig. Er führt Kontrollen durch und erfasst allfällige Mängel mit einem Notebook oder Tablet. Die Bauleiter können zu jeder Zeit und an jedem Ort die gefundenen Mängel per Mängel-Manager erfassen.
Subunternehmen	BackOffice	Das BackOffice des SU kann seine eigenen Stammdaten ändern und die Mängel die der betreffen SU zugewiesen sind einsehen und bearbeiten.
	Ansprechperson	Wenn bei einer Kontrolle ein Mangel von einem Bauleiter erfasst wird, wird es dem betreffenden SU zugeteilt. Das SU bestimmt eine Ansprechperson welche für diesen Mangel zuständig ist. Die Ansprechperson kann also demnach nur diejenigen Mängel sehen und bearbeiten die ihm zugeteilt worden sind.

2. Randbedingungen

2.1 Technische Randbedingungen

Hardware-Vorgaben	
Angemessene Hardwareausstattung	Der Mängel-Manager muss auf einem, zum Release Zeitpunkt, marktüblichen Notebook und Desktop-PC betrieben werden können.
Arbeitsspeicher	> 1 GB
Netzwerk	10/100 Mbit
Software-Vorgaben	
Siehe Punkt 1.1 „Aufgabenstellung“ für Software-Vorgaben	
Vorgaben des Systembetriebs	
Kompatibel mit Windows 8 & 8.1	Die Software muss voll funktionsfähig mit Windows 8 und 8.1 sein.
Datenbanksysteme	Der Mängel-Manager benutzt das Datenbankmanagementsystem (DBMS) PostgreSQL, sollte aber ohne grossen Aufwand auf andere DBMS konfiguriert werden.

Verfügbarkeit der Server	Die Verfügbarkeit sollte mindesten bei der heutigen allgemein normalen Rate von 99% liegen.
Programmiervorgaben	
Programmierung in Java	Die Lösung sollte in Java implementiert werden. Es wird mit der aktuellsten Java Version entwickelt. (Java SE 8)
Architektur	Es wird Vorausgesetzt, dass der Mängel-Manager verschiedene Schichten wie, Modell, Persister, Business, RMI und Webservice aufweist.

2.2 Organisatorische Randbedingungen

Organisation und Struktur	
Organisationsstruktur beim Arbeitgeber	Die Projektaufgabe wurde anhand eines PDF-Dokuments, welche alle nötigen Informationen zur Implementierung und Dokumentationen beinhaltet, gestellt. Die Ansprechpersonen sind klar definiert und werden sich während der Projektdauer voraussichtlich nicht ändern.
Eigenentwicklung	Das Projekt wird zu 100% vom unten genannten Team entwickelt unter der Zuhilfenahme von einigen populären Java Libraries.
Entwicklung als Produkt	Der Mängel-Manager wird im Auftrag der Firma W&W Architekten GmbH erstellt.
Ressourcen (Budget, Zeit, Personal)	
Zeitplan	<i>Siehe Punkt 23. „Weitere Dokumentationen“</i>
Release	Der Abgabetermin des Mängel-Managers ist auf Mittwoch den 20.5.2015, 12.00 Uhr datiert.
Team	Zum Team gehören 6 Studenten der Hochschule Luzern – Departement Wirtschaft, Studiengang Wirtschaftsinformatik, 2. Semester Vollzeit: Luca Kündig (Projektleiter) Sandro Ritz (Projektleiter Stv.) Mike Iten Mike Monticoli Cihan Demir
Präsentation	Der Entwicklergruppe stehen am 21.05.2015, 20 Minuten für die Präsentation des Mängel-Managers zur Verfügung wobei die Punkte: Art und Weise der Lösung, Umgang mit Problemen, Funktionsweise des Deployments, Tests, Lessons Learned erläutert werden.
Organisatorische Standards	
Vorgehensmodell	Die Modellierung der Software sollte mit Hilfe von „use case diagram“ in der Modellierungssprache „Unified Modeling Language“ (UML) durchgeführt. Zusätzlich wird ein „Entity-Relationship-Modell“ (ERM) der zukünftigen Datenbank erstellt.

Entwicklungswerkzeuge	<p>Eine Versionsverwaltungssoftware wird aufgrund der Parallelen Arbeiten von verschiedenen Projektmitarbeitern vorausgesetzt. Für die Erstellung der use case diagramme wurde uns der Enterprise Architect, von den Auftraggebern, empfohlen.</p> <p>Weiterhin wird auch empfohlen, dass man als Datenbankmanagementsystem die PostgreSQL Anwendung verwenden sollte.</p>
GUI Mockup	Bevor das GUI implementiert wird muss ein Mockup den Auftraggebern vorgestellt werden.
Konfigurations- und Versionsverwaltung	Das ganze Projekt wird als „public repository“ von „GitHub“ gehostet womit direkt auch „Git“ für die Versionsverwaltung zum Einsatz kommt.
Testwerkzeuge und Prozesse	Für jede Schicht müssen für mindestens zwei Klassen nach dem TDD-Verfahren (test driven development) mit JUnit schon im Voraus die entsprechenden Testklassen erstellt werden.
Juristische Faktoren	
Datenschutz	Das System muss nach Belieben konfiguriert werden, im Normalfall haben die Unternehmen nur auf Daten Zugriff die für Ihren Geschäftsfall auch relevant ist.

2.3 Konventionen

Konventionen	
Architekturdokumentation	Dokumentierung des ganzen Softwareprozesses unter Benützung des arc42-Template-v60.
Implementierungsrichtlinien	Sich an die Grundlegenden Java Coding Conventions von Oracle einhalten.

3. Kontextabgrenzung

Schnittstelle	Daten	Technologie
Java Application	Quellcode	Via Webservice
PostgreSQL	Objekte	RMI

3.1 Fachlicher Kontext



4. Lösungsstrategie

Der Mängel-Manager wurde aufgrund der Punkte Plattformunabhängigkeit, Sicherheit und Flexibilität in der Programmiersprache Java geschrieben und weist verschiedene Schichten wie Business, Model, Persister, Webservice, RMI auf. Für weitere Informationen siehe Punkt 7. „Verteilungssicht“.

Für die Grafische Benutzeroberfläche (GUI) wurde das aktuelle JavaFX 8 UI-Toolkit ausgewählt. Das Design wurde mithilfe der Layout-Visualisierungs-Software „Scene Builder“ welche von Oracle als Open Source Software daherkommt, verwirklicht. Auf Swing wurde bewusst verzichtet, da das neue JavaFX 8 moderner daherkommt, simpel zu bedienen ist und sich rasant weiterentwickelt.

Die Softwarelösung wird eine relationale Datenbank voraussetzen, in unserem Fall wird es das PostgreSQL sein. Das DBMS sollte leicht austauschbar sein weswegen wir auf JDBC (Java Database Connectivity) setzen um einen Austausch des Datenbankmanagementsystems schnell und einfach vornehmen zu können.

Es wird versucht vier verschiedene Sichten, also je eine Sicht für einen Stakeholder, zu realisieren. *Siehe Punkt 5. „Bausteinsicht“ für weitere detaillierte Informationen zu den verschiedenen Views.*



5. Bausteinsicht

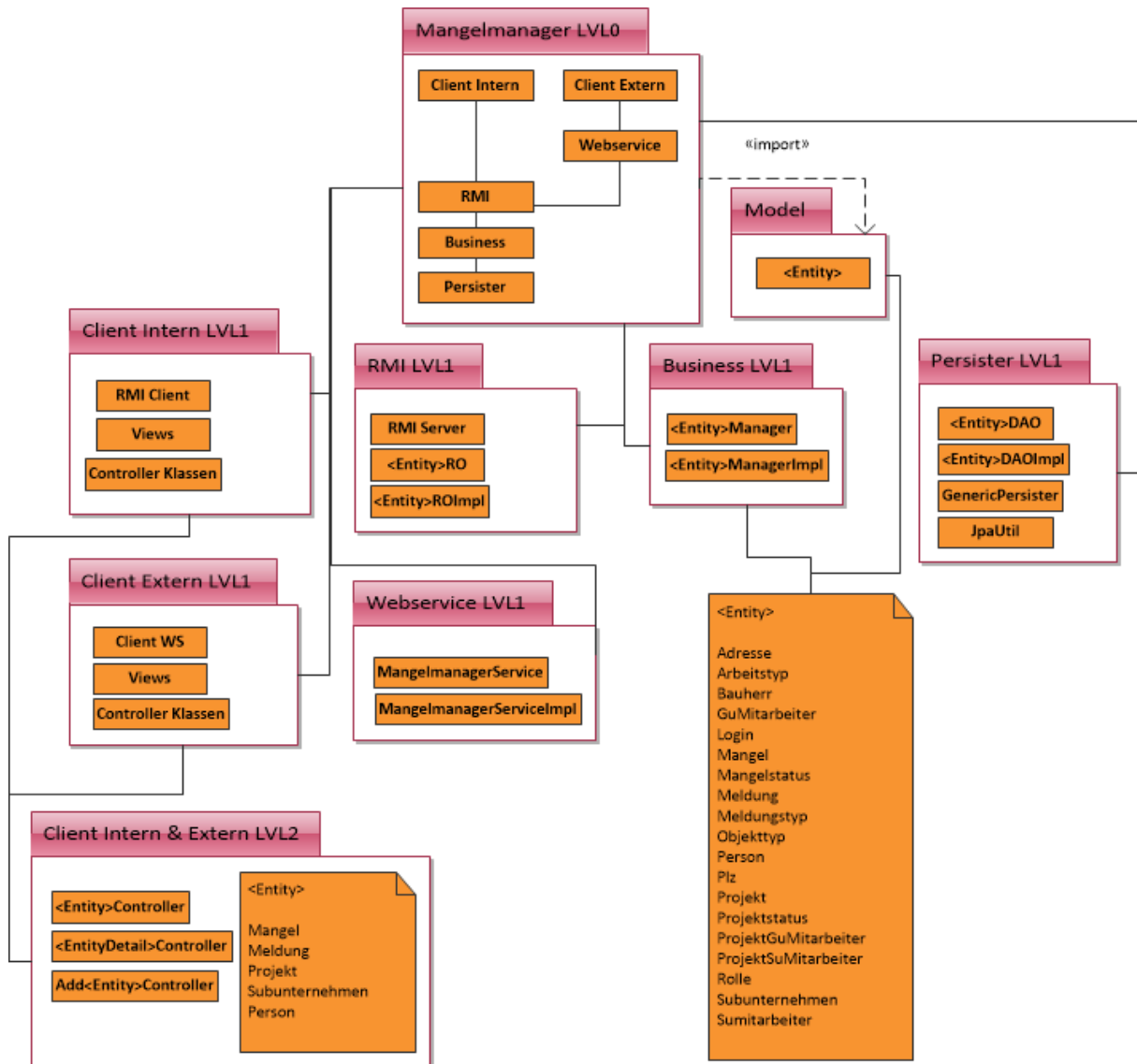


Abbildung 1

5.1 Mängel-Manager Ebene 0

Die Abbildung „Bausteinsicht 1“ zeigt die Hauptbausteine unseres Systems und deren Abhängigkeiten.

5.1.1 Model

Die Models dienen als Bauplan für alle Objekte die persistiert werden müssen.

5.1.2 Client Intern (Black Box-Beschreibung)

Der interne Client bietet den Mitarbeiter im General- und Subunternehmen Einsicht in Projekte, Personal, Mängel, Meldungen und vielem mehr. Je nach Rolle sollen unterschiedliche Zugriffe möglich sein.

Offene Punkte: Rollenbasiertes Zugriffssystem ist noch nicht implementiert.

5.1.3 Client Extern (Black Box-Beschreibung)

Der externe Client bietet den Bauleitern vom Generalunternehmen Mängel und Meldung ausserhalb des Betriebes einzusehen und zu erstellen.

Offene Punkte: Rollenbasiertes Zugriffssystem ist noch nicht implementiert.

5.1.4 Webservice (Black Box-Beschreibung)

Der Webservice stellt eine Schnittstelle für den externen Client zur Verfügung.

5.1.5 RMI (Black Box-Beschreibung)

Die RMI Schnittstelle wird für die Kommunikation zwischen internem Client und dem Unternehmen internem Netzwerk benötigt. Der externe Client greift mittels Webservice auf die RMI Schnittstelle zu.

5.1.6 Business (BlackBox-Beschreibung)

In der Business Schnittstelle befindet sich die komplette Business Logik der Software. Von hier aus wird mit der Persistier Schicht kommuniziert um Daten anzufordern oder auch zu senden.

5.1.7 Persister (BlackBox-Beschreibung)

Der Persister wartet auf Anfragen vom Business Layer. Diese bearbeitet er und greift auf die Daten in der Datenbank zu. Das gewünschte Ergebnis liefert er der Business Schicht wieder zurück.

5.2 Mängel-Manager Ebene 1

5.2.1 Client Intern LVL1 (Whitebox-Beschreibung)

Der Client Intern beinhaltet die ganzen Views im FXML Format, den RMI Client für die Kommunikation mit dem RMI Server und schliesslich alle Controller Klassen. Die Controllerklassen verarbeiten alle Event, welche vom GUI erzeugt werden. In jedem Controller befindet sich eine Referenz auf den RMIClient.



5.2.2 Client Extern LVL1 (Whitebox-Beschreibung)

Dito „Client Intern 1“

5.2.3 Webservice LVL1 (Whitebox-Beschreibung)

Die Webservice Schnittstelle beinhaltet das Interface MangelManagerService und die Klasse MangelmanagerServiceImpl. Diese nehmen Anfragen vom externen Client entgegen und leiten diese der RMI Schnittstelle weiter.

5.2.4 RMI LVL1 (Whitebox-Beschreibung)

Die RMI Schnittstelle beinhaltet einen RMIServer, welcher die Kommunikation mit den Clients aufnimmt. Für alle Funktionalitäten sind Interface und ihre Implementation zuständig. Sie tätigen eine Anfrage auf das Business Layer.

5.2.5 Business LVL1 (Whitebox-Beschreibung)

Für jede Entität existieren ein Interface und seine Implementation. Diese nehmen Anfragen von der RMI Schnittstelle an und leiten diese dem Persister weiter.

5.2.6 Persister LVL1 (Whitebox-Beschreibung)

Für jede Entität existieren ein Interface und seine Implementation. Diese nehmen Anfragen von der Business Schnittstelle an und arbeiten diese ab. Sie holen und persistieren Daten auf Datenbank Ebene. Für generische Abfragen wird die GenericPersister Klasse verwendet. Die JpaUtil Klasse wird benötigt um ein EntityManager zu erzeugen.

5.3 Mängel-Manager Ebene 2

5.3.1 Client Intern & Extern LVL2 (Whitebox-Beschreibung)

Die Client besitzen für die Entitäten Mangel, Meldung, Projekt, Person, Subunternehmen je einen AddController, einen DetailController und einen HauptController.

Der AddController wird benötigt um mittels GUI eine neue Entität zu erzeugen.

Der HauptController ist für die äussere Ansicht und der DetailController für die Innere Ansicht zuständig.



6. Laufzeitsicht

Das untenstehende Sequenzdiagramm visualisiert das Erfassen eines Mangels und das anschließende Anzeigen via internem GUI.

Der User erfasst den Mangel im GUI welches das Objekt via Controller an die RMI-Schnittstelle weiterleitet. Das Objekt wird dann durch die Business-Schicht an den Persister geleitet.

Nach dem Speichervorgang wird auf die Mangel-Hauptansicht weitergeleitet. Via Doppelklick auf den Mangel wird dann die Detailansicht gestartet.

Die Anfrage wird wie oben durch die Schichten geleitet. Im Diagramm ist dann zu sehen, wie die Anfrage das gefundene Objekt wieder durch alle Schichten nach oben bis zum Controller leitet. Der Controller holt sich dann die Entsprechenden Informationen aus dem Objekt und zeigt sie im GUI an.

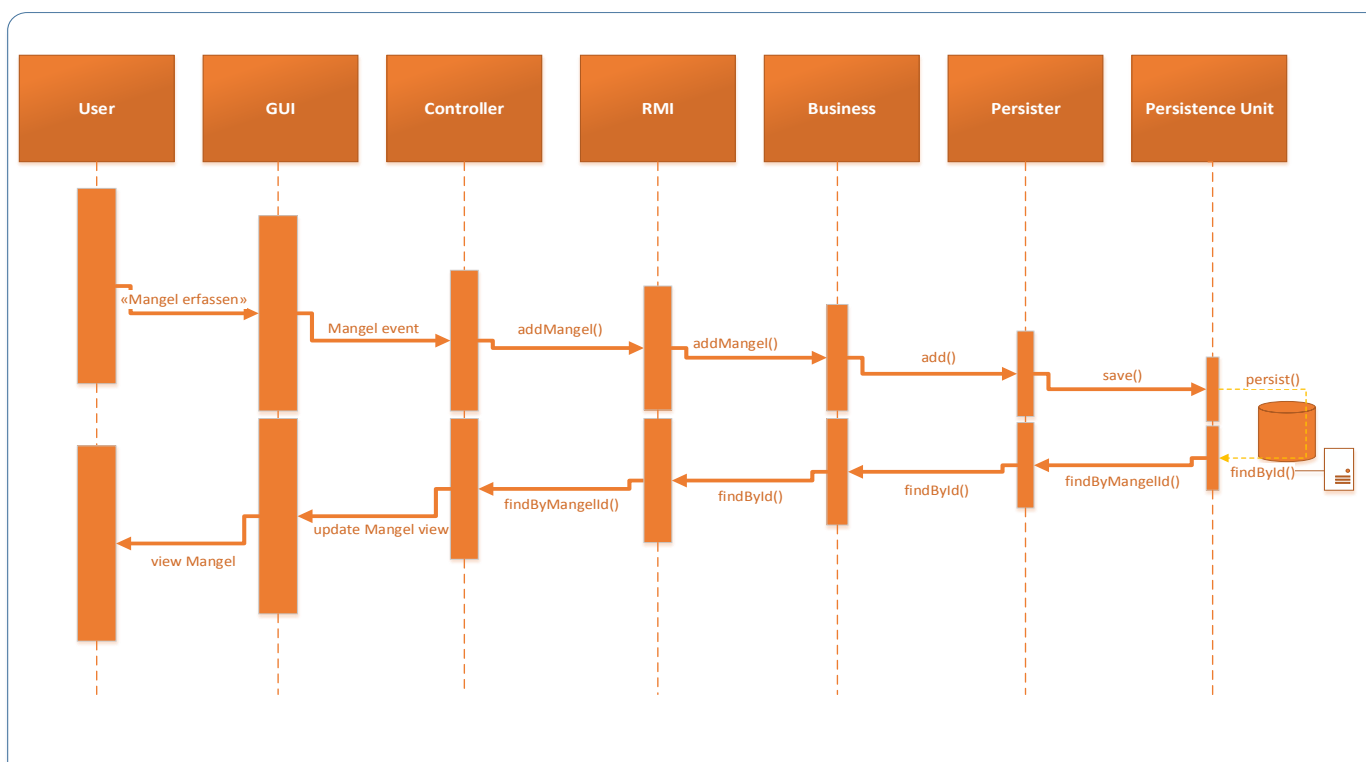


Abbildung 2

7. Verteilungssicht

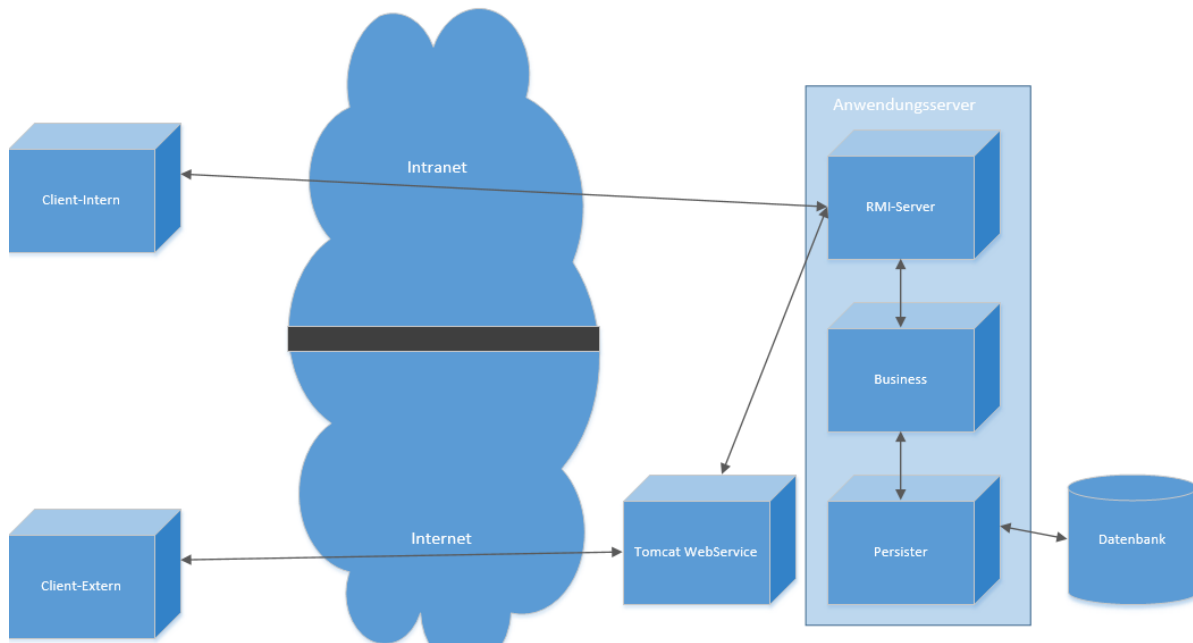


Abbildung 3

Die Software MangelManager verfügt über eine sogenannte 3-Tier Architektur. Client, Businesslogik und Daten sind getrennt.

Der MangelManager kann mit zwei Clients aufgerufen werden. Einerseits über den Intranet Client (Client-Intern.jar) Dieser kommuniziert im Internet direkt mit dem RMI Server. Die RMI-Schnittstelle nimmt sämtliche Anfragen entgegen und leitet sie über die Business und Persister Schicht an die Datenbank weiter.

Der WebClient (Client-Extern.jar) Ermöglicht eine Verbindung aus dem Internet. Der Client kommuniziert mit dem Apache Tomcat-Server dieser stellt den MangelManagerService zur Verfügung welcher dann wieder auf die RMI-Schnittstelle zugreift.

So ist gewährleistet, dass Daten nur via RMI an den Persister gelangt und auch nur der Persister Daten Manipulieren kann.

8. Konzepte

8.1 Typische Muster Strukturen

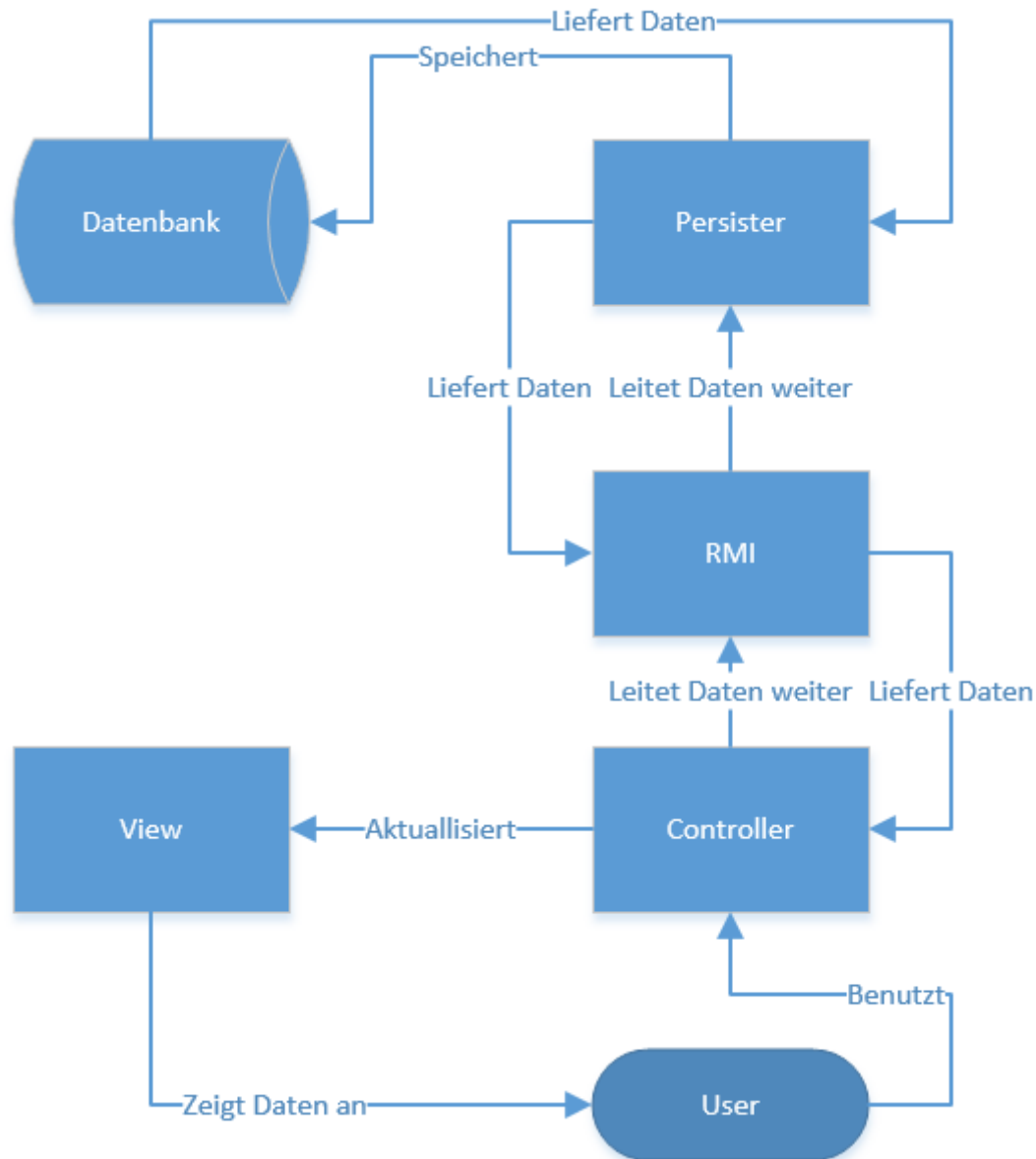


Abbildung 4

Szenario 1:

Der User Öffnet eine View. Der Controller holt nun via RMI-Schnittstelle und dem Persister die Daten aus der Datenbank. Die gesuchten Daten werden nun wieder durch alle Schichten an den Controller weitergeleitet. Dieser wertet nun die erhaltenen Daten aus, sortiert, formatiert sie und zeigt sie in der View an.

Szenario 2:

Der User erstellt ein neues Objekt. Der Controller erfasst die Daten des Objekts und leitet sie via RMI an den Persister weiter. Dieser speichert dann die Daten in die Datenbank. Beim Aktualisieren der View holt der Controller wie bei Szenario 1 die Daten aus der Datenbank.

8.2 Typische Abläufe

Siehe diverse Aktivitätsdiagramme Kapitel 15. „UseCase Dokumentation“

8.3 Persistenz

Um die Persistenz sicherzustellen müssen verschiedene Bedingungen wie erfolgreicher Login, stabile Konnektivität und reibungslose Kommunikation zwischen Client bis zu der letzten Schicht der Architektur (Datenbank) des Mängel-Managers erfüllt sein.

Der Client muss z.B. bis zur Erstellung eines Mangels mit der Datenbank verbunden sein, um diese Informationen auch erfolgreich persistieren zu können. Alles, was nicht mit der Datenbank synchronisiert worden ist, aufgrund Unvollständigkeit befindet sich im Cache-Speicher des Clients und ist somit flüchtig, was bei einem Systemabsturz verloren wäre.

Die Unmengen an Daten, welche die Datenbank beherbergt, werden bei der Persistenz schlussendlich, für alle erfolgreiche Verbindungen, durch den Massenspeicher der Server zur Verfügung gestellt.

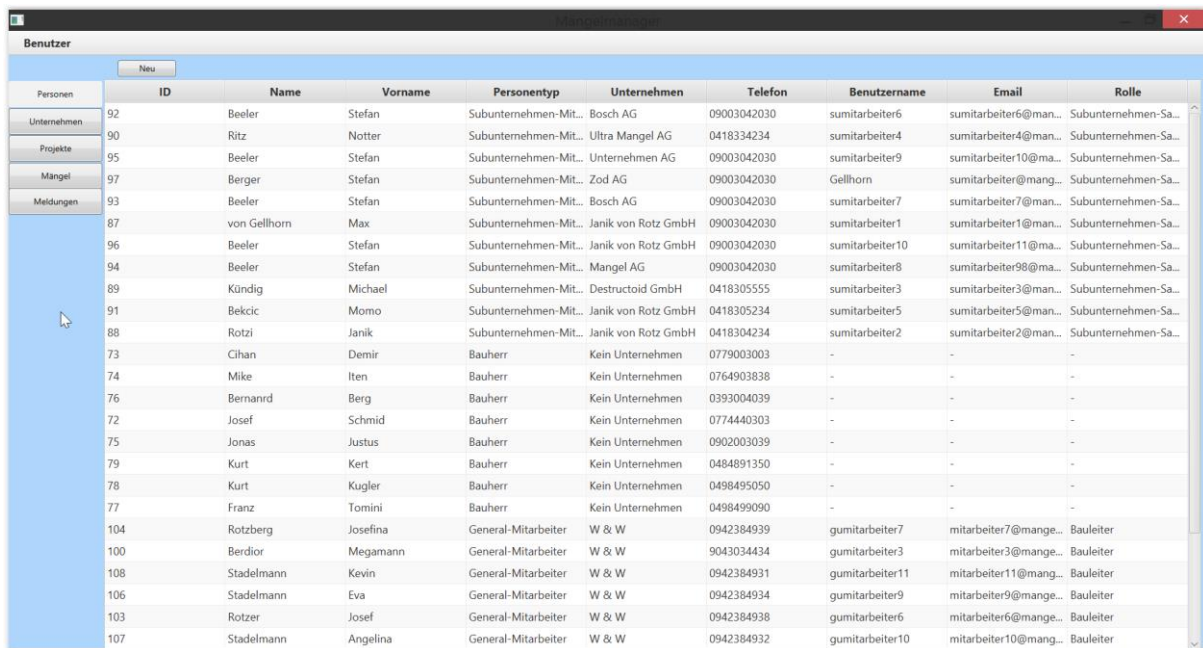


8.4 Benutzungsoberfläche



Abbildung 5

Beide Clients sowohl extern als auch intern verfügen über ein Login-Fenster. Beim internen Client kann zusätzlich die IP und der Port des RMI-Servers angegeben werden.

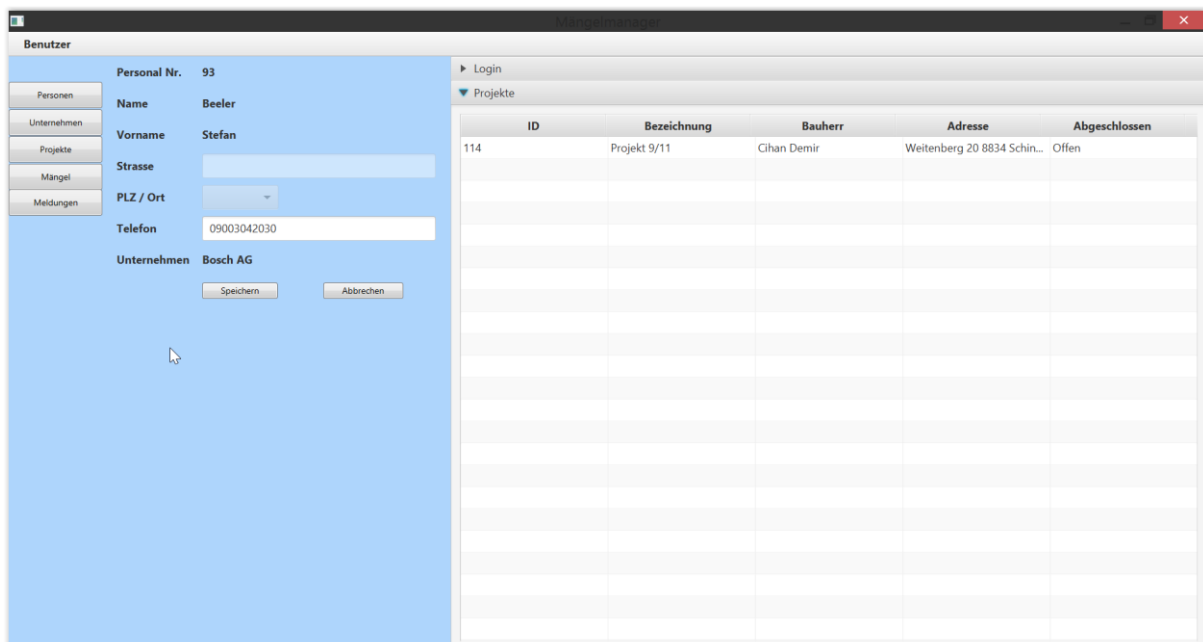


ID	Name	Vorname	Personentyp	Unternehmen	Telefon	Benutzername	Email	Rolle
92	Beeler	Stefan	Subunternehmen-Mit...	Bosch AG	09003042030	sumitarbeiter6	sumitarbeiter6@man...	Subunternehmen-Sa...
90	Ritz	Nottter	Subunternehmen-Mit...	Ultra Mangel AG	0418334234	sumitarbeiter4	sumitarbeiter4@man...	Subunternehmen-Sa...
95	Beeler	Stefan	Subunternehmen-Mit...	Unternehmen AG	09003042030	sumitarbeiter9	sumitarbeiter10@ma...	Subunternehmen-Sa...
97	Berger	Stefan	Subunternehmen-Mit...	Zod AG	09003042030	Gellhorn	sumitarbeiter@mang...	Subunternehmen-Sa...
93	Beeler	Stefan	Subunternehmen-Mit...	Bosch AG	09003042030	sumitarbeiter7	sumitarbeiter7@man...	Subunternehmen-Sa...
87	von Gellhorn	Max	Subunternehmen-Mit...	Janik von Rotz GmbH	09003042030	sumitarbeiter1	sumitarbeiter1@man...	Subunternehmen-Sa...
96	Beeler	Stefan	Subunternehmen-Mit...	Janik von Rotz GmbH	09003042030	sumitarbeiter10	sumitarbeiter11@ma...	Subunternehmen-Sa...
94	Beeler	Stefan	Subunternehmen-Mit...	Mangel AG	09003042030	sumitarbeiter8	sumitarbeiter9@ma...	Subunternehmen-Sa...
89	Kündig	Michael	Subunternehmen-Mit...	Destructoid GmbH	0418305555	sumitarbeiter3	sumitarbeiter3@man...	Subunternehmen-Sa...
91	Bekic	Momo	Subunternehmen-Mit...	Janik von Rotz GmbH	0418305234	sumitarbeiter5	sumitarbeiter5@man...	Subunternehmen-Sa...
88	Rotzi	Janik	Subunternehmen-Mit...	Janik von Rotz GmbH	0418304234	sumitarbeiter2	sumitarbeiter2@man...	Subunternehmen-Sa...
73	Cihan	Demir	Bauherr	Kein Unternehmen	0779003003	-	-	-
74	Mike	Iten	Bauherr	Kein Unternehmen	0764903838	-	-	-
76	Bernanrd	Berg	Bauherr	Kein Unternehmen	0393004039	-	-	-
72	Josef	Schmid	Bauherr	Kein Unternehmen	077440303	-	-	-
75	Jonas	Justus	Bauherr	Kein Unternehmen	0902003039	-	-	-
79	Kurt	Kert	Bauherr	Kein Unternehmen	0484891350	-	-	-
78	Kurt	Kugler	Bauherr	Kein Unternehmen	0498495050	-	-	-
77	Franz	Tomini	Bauherr	Kein Unternehmen	0498499090	-	-	-
104	Rotzberg	Josefina	General-Mitarbeiter	W & W	0942384939	gumitarbeiter7	mitarbeiter7@mange...	Bauleiter
100	Berdior	Megamann	General-Mitarbeiter	W & W	9043034434	gumitarbeiter3	mitarbeiter3@mange...	Bauleiter
108	Stadelmann	Kevin	General-Mitarbeiter	W & W	0942384931	gumitarbeiter11	mitarbeiter11@mang...	Bauleiter
106	Stadelmann	Eva	General-Mitarbeiter	W & W	0942384934	gumitarbeiter9	mitarbeiter9@mange...	Bauleiter
103	Rotzer	Josef	General-Mitarbeiter	W & W	0942384938	gumitarbeiter6	mitarbeiter6@mange...	Bauleiter
107	Stadelmann	Angelina	General-Mitarbeiter	W & W	0942384932	gumitarbeiter10	mitarbeiter10@mang...	Bauleiter

Abbildung 6

Die Software verfügt jeweils über Tabellarische Ansicht aller Objekte. Hier werden Objektspezifische Informationen dargestellt.

Via Doppelklick kann zur Detailansicht eines Objekts gewechselt werden.



ID	Bezeichnung	Bauherr	Adresse	Abgeschlossen
114	Projekt 9/11	Cihan Demir	Weitenberg 20 8834 Schin...	Offen

Abbildung 7

In der Detailansicht finden sich erneut Informationen zum Objekt, die Teilweise auch angepasst und gespeichert werden können. Zusätzlich werden Tabellen verknüpfter Objekte dargestellt.

8.5 Ergonomie

Die Benutzeroberfläche wurde möglichst simpel gestaltet. Sie soll eine unkomplizierte Bedienung durch die User ermöglichen.

Die Verschiedenen Ansichten wurden optisch gleich gestaltet, damit ein einheitliches Bild der Applikation entsteht.

8.6 Ablaufsteuerung

Siehe Kapitel 8.3 „Typische Abläufe“

8.7 Transaktionsbehandlung

Die Transaktionsbehandlung funktioniert über den Entity-Manager. Bevor die Daten in die Datenbank geschrieben werden, werden sämtliche Aufgaben vom Entity-Manager ausgeführt. Werden Fehler gefunden, werden die Daten weder in die Datenbank geschrieben noch gelesen.

Transaktionen gemäss dem ACID Prinzip:

- **Atomicity:** Die Daten werden ganz oder gar nicht übertragen, sind nicht alle Daten resp. Felder ausgefüllt, werden die Daten nicht gespeichert.
- **Consistency:** Nach Ausführung der Transaktion sind die Daten fest vorhanden. Das heisst, sie sind auf der Datenbank persistent abgespeichert.
- **Isolation:** Bei gleichzeitigen Ausführungen von Transaktionen beeinflussen diese sich gegenseitig nicht.
- **Durability:** Die Daten sind dauerhaft auf der Datenbank vorhanden, können also nicht verloren gehen.

8.8 Sessionbehandlung

Nicht notwendig.

8.9 Sicherheit

Um sich überhaupt mit der Datenbank verbinden zu können wird neben der Client Software ein Login vorausgesetzt, welcher zuerst vom BackOffice erstellt werden muss. Wenn eine erfolgreiche Authentifikation stattgefunden hat wird als aller erstes geprüft welche Rechte der Benutzer hat und dementsprechend werden nur Items angezeigt die für diesen Benutzer relevant sind.



Momentan gibt es zwei Benutzergruppen, das BackOffice GU und das BackOffice SU mit folgenden Rechten:

- **Die BackOffice GU:**
Kann neue BackOffice GU wie auch BackOffice SU Benutzer erstellen. Kann auf alle Informationen zugreifen, Stammdaten wie auch Mängeldaten hinzufügen bearbeiten oder löschen. Im Grunde genommen ist der BackOffice GU Benutzer eine Art Administrator.
- **Die BackOffice SU:**
Kann nur neue BackOffice SU erstellen (dient anstelle der Ansprechpartner SU).
Kann nur auf Informationen zugreifen welche für die BackOffice SU bestimmt sind.
Kann nur die eigenen Stammdaten bearbeiten. Kann nur die Mängel bearbeiten die Ihr zugeteilt sind.

Es war auch eine Benutzergruppe für den Bauleiter und die Ansprechperson geplant, jedoch mussten wir den Funktionsumfang, aufgrund einer Verringerung der Projektmitglieder, kürzen um den Zeitplan auch einhalten zu können.

8.10 Kommunikation und Integration mit anderen IT-Systemen

Die Kommunikation mit anderen IT-Systemen wird durch den Webservice realisiert.

8.11 Verteilung

Siehe Kapitel 7. „Verteilungssicht“

8.12 Plausibilisierung und Validierung

Aufgrund der fehlenden Man-Power (Absenz von Max von Gellhorn), blieb uns leider nicht genug Zeit eine Plausibilisierung und eine Validierung zu implementieren.

8.13 Ausnahme-/Fehlerbehandlung

Aufgrund der fehlenden Man-Power (Absenz von Max von Gellhorn), blieb uns leider nicht genug Zeit eine Ausnahme-/Fehlerbehandlung zu implementieren.

8.14 Management des Systems & Administrierbarkeit

Der Mängel-Manager wird auf einem IT-System zentral in der Generalunternehmung der „W&W Architekten GmbH“ betrieben. Die Stakeholder werden bezüglich spezieller Eingriffs- oder Konfigurationsmöglichkeiten bei Bedarf, eingeschult.



8.15 Logging, Protokollierung, Tracing

Die Anwendungsmeldungen werden hauptsächlich mithilfe des „log4j“ Frameworks, welcher von der Apache Software Foundation bereitgestellt wird, geloggt.

8.16 Geschäftsregeln

Da dieses Projekt und die Firma W & W fiktiv sind bestehen keine Geschäftsregeln.

8.17 Konfigurierbarkeit

RMI-Server:

Die RMI-Serverkomponente ist via Propertyfile konfigurierbar. In diesem File kann sowohl die IP als auch der Port unter dem der Server laufen soll eingestellt werden. Dies muss vor dem Start passieren.

RMI-Client:

Der RMI Client kann via GUI so konfiguriert werden. Auch hier können IP und Port des laufenden RMI-Servers angegeben werden.

WebService:

Der Webservice kann ebenfalls via Propertyfile konfiguriert werden. Hier besteht ebenfalls die Möglichkeit die IP und den Port an den RMI-Server anzupassen.

8.18 Parallelisierung und Threading

Wird automatisch durch RMI erledigt.

8.19 Internationalisierung

Die „W&W Architekten GmbH“ ist ein in der Schweiz tätiges Unternehmen. Es wird grundsätzlich mit schweizer Standard Einstellungen gearbeitet, weswegen wir auf Anpassungen betreffend länderspezifischer Merkmale verzichten können.

8.20 Migration

Da die „W&W Architekten GmbH“ kein Altsystem benutzt hat gibt es wenig zu migrieren. Es müssen Stammdaten und eventuell offene, schon bekannte Mängel erfasst werden.

Die die bereits bekannten, offenen Mängel schon im Excel Format bereit stehen, können sie einfach mithilfe des GUI per Copy & Paste übertragen werden.

8.21 Testbarkeit

Siehe Kapitel 18. „Funktionale Test's“



8.22 Skalierung, Clustering

Die Datenbank kann problemlos skaliert werden. Ebenso kann die Rechenleistung des Servers problemlos angepasst werden. Unsere Applikation funktioniert für eine Person oder 100 genau gleich.

8.23 Hochverfügbarkeit

Eine Verfügbarkeit von 99% ist heutzutage ein Standard bei qualitativ hochwertigen EDV-Geräten und wird somit vorausgesetzt. Eine Hochverfügbarkeit von 99.99% würde der Verfügbarkeitsklasse 4 entsprechen welche von uns auch angestrebt wird und mithilfe des USV-Systems auf ziemlich realistisch scheint. Die Verfügbarkeitsklasse 4 erlaubt es pro Monat 4:23 Minuten und pro Jahr 52:36 Minuten offline zu sein.



9. Entwurfsentscheidungen

9.1 Entwurfsentscheidung 1

9.1.1 Fragestellung

Womit erstellen wir unser GUI?

9.1.2 Rahmenbedingungen

Die Wahl wurde mehrheitlich uns überlassen. Da wir das mit Java Eclipse programmierten lag es nahe den Scenebuilder zu verwenden und mit JavaFX zu arbeiten.

9.1.3 Annahmen

Die Person, welche sich um das GUI kümmert hat einige Vorkenntnisse zu JavaFX aus dem 1. Modul Java im Studium.

9.1.4 Betrachtete Alternativen

Swing: Wurde allerdings verworfen aufgrund fehlendem Vorwissen.

9.1.5 Entscheidung

Wir haben uns für JavaFX entschieden, da wir dort schon Vorkenntnisse haben und wissen wir man mit dem Scenebuilder umgeht.

9.2 Entwurfsentscheidung 2

9.2.1 Fragestellung

Wie gestalten wir das GUI übersichtlich?

9.2.2 Rahmenbedingungen

Das GUI sollte sich nicht überladen anfühlen und trotzdem viele Informationen vorweisen.

9.2.3 Annahmen

Mittels JavaFX.Accordion können wir Informationen verschachteln.

9.2.4 Betrachtete Alternativen

Durch TableViews können die Informationen kompakter angezeigt werden.



9.2.5 Entscheidung

Wir entschieden uns für eine Kombination. Wir benutzen Accordions um die angezeigte Information zu begrenzen und innerhalb dieser TableViews, um die Informationen kompakter und übersichtlich darzustellen.

9.3 Entwurfsentscheidung 3

9.3.1 Fragestellung

Wie soll das GUI aussehen?

9.3.2 Rahmenbedingungen

Unser Ziel war es ein intuitives GUI zu designen. Unser Hauptkriterium war daher, dass man sich schnell einarbeiten kann.

9.3.3 Annahmen

Durch klare Kennzeichnung und gute Namengebung findet man sich schnell zurecht. Eine Navigationsleiste und Verbindungen zwischen den einzelnen Entitäten ermöglicht schnelles Arbeiten

9.3.4 Betrachtete Alternativen

Es gab für uns keine wirkliche Alternative.

9.3.5 Entscheidung

Wir erstellen eine Navigationsleiste wo die wichtigsten Fenster direkt angewählt werden können. Innerhalb der Entitäten ist schnelles springen zwischen z.B. Projekt und Mängel möglich. Desweiteren sind alle relevanten Informationen jederzeit ersichtlich.



10. Qualitätsszenarien

10.1 Qualitätsbaum

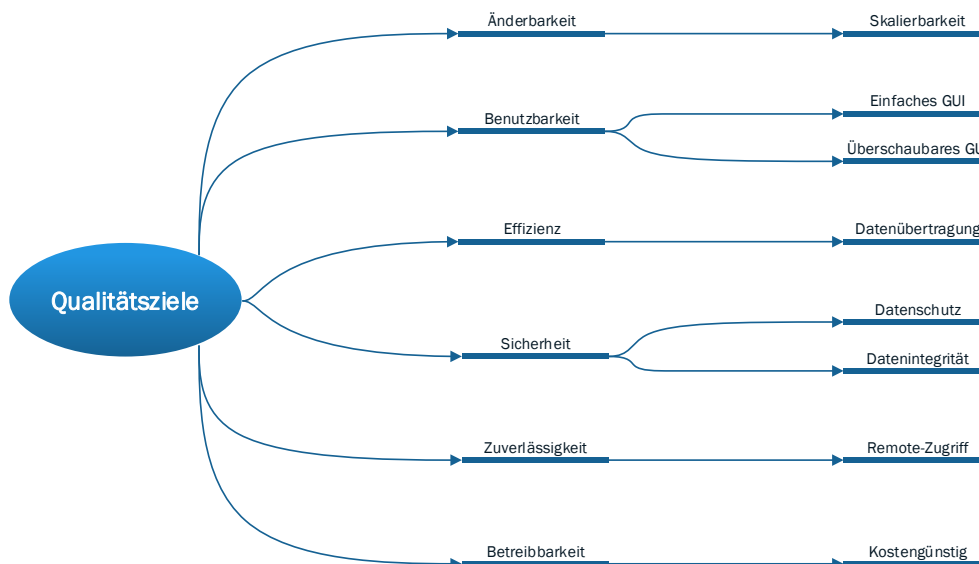


Abbildung 8

Skalierbarkeit

Der Mängelmanager sollte einfach erweiterbar sein mit zusätzlicher Hardware, falls der Gebrauch des Managers expandiert.

Einfaches GUI

Es soll ein simples gestaltetes GUI sein, welches für Jedermann einfach zu bedienen ist, selbst für Mitarbeiter, die diesen Manager noch nie zuvor verwendet haben. Diese sollten den Manager mit einem Blick bereits einfach verwenden können.

Überschaubares GUI

Wenn man den Manager begutachtet und damit arbeitet, soll er nicht nur einfach zu bedienen sein, sondern auch dem Auge "schmeicheln". Es soll schön anzusehen sein demnach auch alles direkt auf den ersten Blick zu finden sein.

Datenübertragung

Alle Daten die hinzugefügt werden, sind es Mängel, Meldungen, Projekte oder eine neue Ansprechperson für ein Subunternehmen, sollen direkt für die jeweils autorisierten Mitarbeiter zur Verfügung stehen. So sollen falls neue Kriterien für das Filtern hinzugefügt wurden, sofort verwendet werden können.

Datenschutz

Um auf die jeweils berechtigten Daten zugreifen zu können, muss man sich über den Login bekennen. Falls die korrekten Login-Informationen angegeben wurden, kann der Benutzer auf die ihm berechtigten Daten zugreifen.

Datenintegrität

Es wurde darauf geachtet, dass keine Redundanzen und somit unnötiger Ballast vorhanden ist. So wurden auch für die einzelnen Felder die korrekten Attribute hinzugefügt wie z.B. dass Felder, die ausschliesslich aus Zahlen bestehen sollen, keine Buchstaben verwenden können.

Remote-zugriff

Der Manager soll z.B. auch über das Tablet des Bauherrn aufzurufen sein. Damit der Bauherr bei allfälligen Mängeln vor Ort direkt Mängel oder Meldungen über sein Tablet erfassen, bearbeiten oder als erledigt kennzeichnen kann.

10.2 Bewertungsszenarien

Nutzungsszenario (Qualitätsziel Zuverlässigkeit):

Dem Bauleiter wird vor Ort mitgeteilt, dass einige Ressourcen bezüglich des Baus fehlen. Entweder muss er den Projektplan ändern oder er muss sich beim Subunternehmen melden.

Dazu erstellt er über sein Tablet erstmal einen Mangel um später noch einmal darauf hingewiesen zu werden.

Nutzungsszenario (Qualitätsziel Benutzbarkeit):

Ein Bauleiter muss sich neu mit dem System befassen und startet zum ersten Mal die Menüansicht. Er sieht alles auf einen Blick und wenn er die Menüwahl links ändert, bleibt alles einheitlich bis auf den mittleren Bereich welcher sich dynamisch verändert. Somit bleibt alles übersichtlich und einfach zu bedienen.

11. Risiken

Das Risikomanagement umfasst sämtliche Maßnahmen zur systematischen Erkennung, Analyse, Bewertung, Überwachung und Kontrolle von Risiken. Hier werden die uns wichtigsten Risiken aufgezählt.

Risiko 1:

Datenverlust von Projektdaten

Eintrittswahrscheinlichkeit:

Die Eintrittswahrscheinlichkeit ist klein.

Schadenshöhe:

Je nach Verlust grösser oder kleiner

Massnahmen:



Jeden Tag wird ein Backup der Kundendaten durchgeführt. Datenbank ist hinter einer USV, so ist auch bei Stromausfall die Datenbank weiterhin online.

Risiko 2:

Angriff von Aussen auf den Server

Eintrittswahrscheinlichkeit:

Die Eintrittswahrscheinlichkeit ist klein.

Schadenshöhe:

Grosser Verlust

Massnahmen:

Der Server soll gut geschützt sein und zu jederzeit mit dem aktuellsten Firewall ausgestattet sein.



12. Glossar

Begriff	Beschreibung
Graphical user interface	Auf Deutsch auch grafische Benutzeroberfläche genannt. Dient dazu eine Applikation mittels Symbole oder Steuerelemente bedienbar zu machen um Aktionen einfach mit der Maus und der Tastatur ausführen zu können.
Datenbankmanagementsystem (DBMS)	Ein Datenbankmanagementsystem ist eine Software, welche an erster Stelle, eine Datenbank zur Verfügung stellt, die verschiedene Nutzern-Anfragen oder Anfragen von anderen Programmen entgegennehmen und bearbeiten kann.
Entity-Relationship-Modell (ERM)	Auf Deutsch etwa Gegenstand-Beziehung-Modell, welches primär für den Entwurf der Datenbank dient, bevor sie per SQL (Structured Query Language) implementiert wird.
Mockup	Im Deutschen auch unter Maquette bekannt. Kann als Instrument bezeichnet werden um Entwürfe oder Vorformen einer Website oder eines App's zur Konzeption zu erstellen.
Middleware	Eine Middleware ist eine sogenannte Zwischenanwendung die zwischen zwei verschiedenen Schichten eines Programms vermittelt was somit auch die Komplexität und Infrastruktur gegenüber der Aussenwelt verborgen hält.
Coding conventions	Im Deutschen auch unter dem Begriff Programmierstil bekannt, welcher in der Programmierung vorgibt nach welchen Regeln der Quellcode erstellt werden muss.
Use cases	Das ist ein Instrument welches für die Erstellung von verschiedenen Szenarien dient, die einem Stakeholder (Akteur) möglich sind, zur Erreichung eines bestimmten technischen Zieles.
Unified Modeling Language (UML)	Die Unified Modeling Language ist eine grafische Modellierungssprache die grundsätzlich zur Visualisierung von Diagrammen benutzt wird.
Framework	Unter einem Framework versteht man im Bereich der Software-Engineering ein Rahmenwerk, welches Komponenten und Libraries zur Verfügung stellt womit sie dem Programmierer beim Erstellen einer Applikation unter die Arme greift.
Java Database Connectivity (JDBC)	Die Java Database Connectivity ist eine universelle Java Datenbankschnittstelle, die eine Schnittstelle zu Datenbanken verschiedener Hersteller bietet.

Die Beschreibungen der Begriffe oben sind kurzgefasste, vereinfachte Formen der Beschreibungen die jeweils unter <http://www.itwissen.info> oder <http://de.wikipedia.org> zu finden sind.



13. Weitere Arc42 Dokumentationen

14. Requirements Dokumentation

Nr.	Anforderung
1	Datenverwaltung Back-Office für Subunternehmen
2	Datenverwaltung des Unternehmens
3	Datenverwaltung für alle Bauprojekte des Unternehmens
4	Projektobjekt muss ersichtlich sein (Mehrfamilienhaus, Wohnung usw.)
5	Projekttyp klar ersichtlich (Neubau, Renovation...)
6	Projekte haben Start und Enddatum
7	Adresse des Projekts ist ersichtlich
8	Bauherr des Projekts ist ersichtlich
9	Arbeiten/Mängel aller Bauleiter eines Projekts sind zeitlich aufgeführt
10	Alle involvierten Subunternehmen sind ersichtlich
11	Die Ansprechpersonen der Subunternehmen sind zeitlich aufgeführt
12	Mängel können vor Ort durch den Bauleiter erfasst werden
13	Mängeldaten werden direkt auf dem Server der GU gespeichert
14	Benutzerschnittstelle für Administration
15	Benutzerschnittstelle für Bauleiter
16	Benutzerschnittstelle für Subunternehmen
17	Sicheres Login auf allen Benutzerschnittstellen
18	Datenzugriff für die jeweiligen Benutzer einschränken
19	Subunternehmenssoftware wird Online zum Download zur Verfügung gestellt
20	Zuverlässiges Datenbankmanagementsystem wird zum Verwalten der Daten verwendet
21	Software darf nicht Datenbank abhängig sein
22	Projekte können durch mehrere Bauleiter betreut werden
23	Hauptbauleiter des Projekts ist ersichtlich und kann ausgetauscht werden
24	Ein Subunternehmen kann in mehreren Projekten tätig sein
25	Projekte können durch mehrere, müssen aber durch mindestens eine Ansprechperson des SU betreut werden.
26	Bauleiter dürfen nur eigenen Projekte einsehen und bearbeiten
27	Bauleiter eines Projekts werden durch die Administration des GU festgelegt
28	Ansprechpersonen eines SU dürfen nur eigene Projekte einsehen und bearbeiten
29	Stammdaten müssen durch Administrative Mitarbeiter Initial erfasst werden
30	Sobald Mängel erfasst wurden sind diese auch für das SU ersichtlich
31	Mängel können durch den SU Mitarbeiter quittiert werden
32	Mängel können mehrere Meldungen beinhalten, die durch GU oder SU erfasst werden können
33	Meldungen von Mängeln stehen dem Bauleiter sofort zur Verfügung
34	Mängelbehebung muss durch SU explizit angegeben werden.
35	GU Bauleiter kontrolliert Mängel und bestätigt falls Behebung in Ordnung
36	Sollten die Mängel nicht behoben sein kann der Bauleiter die Mängel erneut dem SU schicken
37	Meldungen sind nach dem Speichern unveränderbar
38	Alle UIs müssen die Mängeldaten nach diversen Kriterien filtern können
39	Mängeldaten müssen in Listenform angezeigt und ausgedruckt werden können.
40	Benutzerverwaltung durch GU-BO
41	Adminuser für GU-BO wird initial erstellt
42	Adminuser für SU-BO wird bei Erfassung von SU im GU-BO erstellt

15. UseCase Dokumentation

15.1 UseCase001

15.1.1 UseCase001 Beschreibung

UseCase001	SU-Backoffice Datenverwaltung
Autor	Sandro Ritz
Ziel	Der BackOffice-Mitarbeiter der Subunternehmen kann Ansprechpersonen in der Datenbank sowohl erfassen, löschen, ändern, lesen und einem Projekt zuweisen.
Kategorie	Primär, ist essenziell für die Funktionalität des Mängelmanagers
Vorbereitungen	Ein SU-Ansprechperson ist noch nicht im System erfasst
Nachbedingungen Erfolg	Die SU-Ansprechperson ist mit ID, Namen, Vornamen, Rolle, Direkt-Telefon und Login Daten im Mängelmanager erfasst
Nachbedingungen Fehlschlag	Die SU-Ansprechperson ist nicht oder fehlerhaft (Mit falschen oder unvollständigen Informationen) im Mängelmanager erfasst
Akteure	SU-Admin, SU-Ansprechperson
Szenario 01	
Auslösendes Ereignis	Neuer SU-Mitarbeiter tritt in ein Unternehmen ein.
Beschreibung	<ol style="list-style-type: none">BackOffice Mitarbeiter überprüft ob der Mitarbeiter bereits im System istBackOffice Mitarbeiter erfasst den Mitarbeiter mit ID, Namen, Vornamen, Rolle, Direkt-Telefon und Login Daten im MangelmanagerMitarbeiter klickt auf Save und speichert so den neuen Mitarbeiter und BenutzerDie Benutzerdaten werden in der Datenbank gespeichert
Szenario 02	
Auslösendes Ereignis	SU-Mitarbeiter verlässt das Unternehmen
Beschreibung	<ol style="list-style-type: none">BackOffice Mitarbeiter überprüft ob der SU-Mitarbeiter im System vorhanden ist.BackOffice Mitarbeiter deaktiviert SU-Mitarbeiter inkl. Login. Der SU-Mitarbeiter wird nicht aus der Datenbank gelöscht
Szenario 03	
Auslösendes Ereignis	SU-Mitarbeiter Information ändern. z.B. Telefonnummer wird geändert.
Beschreibung	<ol style="list-style-type: none">BackOffice Mitarbeiter überprüft ob der SU-Mitarbeiter im System vorhanden ist.SU-Mitarbeiter Datensatz wird geöffnet.Daten werden geändert.Datensatz wird mittels Klick auf „Speichern“ abgespeichert und in der Datenbank aktualisiert.
Szenario 04	
Auslösendes Ereignis	Einem Projekt wird eine Ansprechperson zugewiesen.

Beschreibung	<ol style="list-style-type: none"> 1. BackOffice Mitarbeiter öffnet in der Projektverwaltung ein ausgewähltes Projekt. 2. Falls keine Ansprechperson zugeteilt ist, kann eine Ansprechperson ausgewählt und gesetzt werden. 3. Falls bereits eine Ansprechperson zugeteilt ist, kann diese geändert werden. Mit einem Klick auf Ansprechperson ändern öffnet sich eine Eingabemaske. 4. Neue Ansprechperson wird ausgewählt. 5. Mit „OK“ bestätigt man die Änderung. 6. Daten werden in der Datenbank aktualisiert.
---------------------	---

15.1.2 UseCase001 Visualisierung

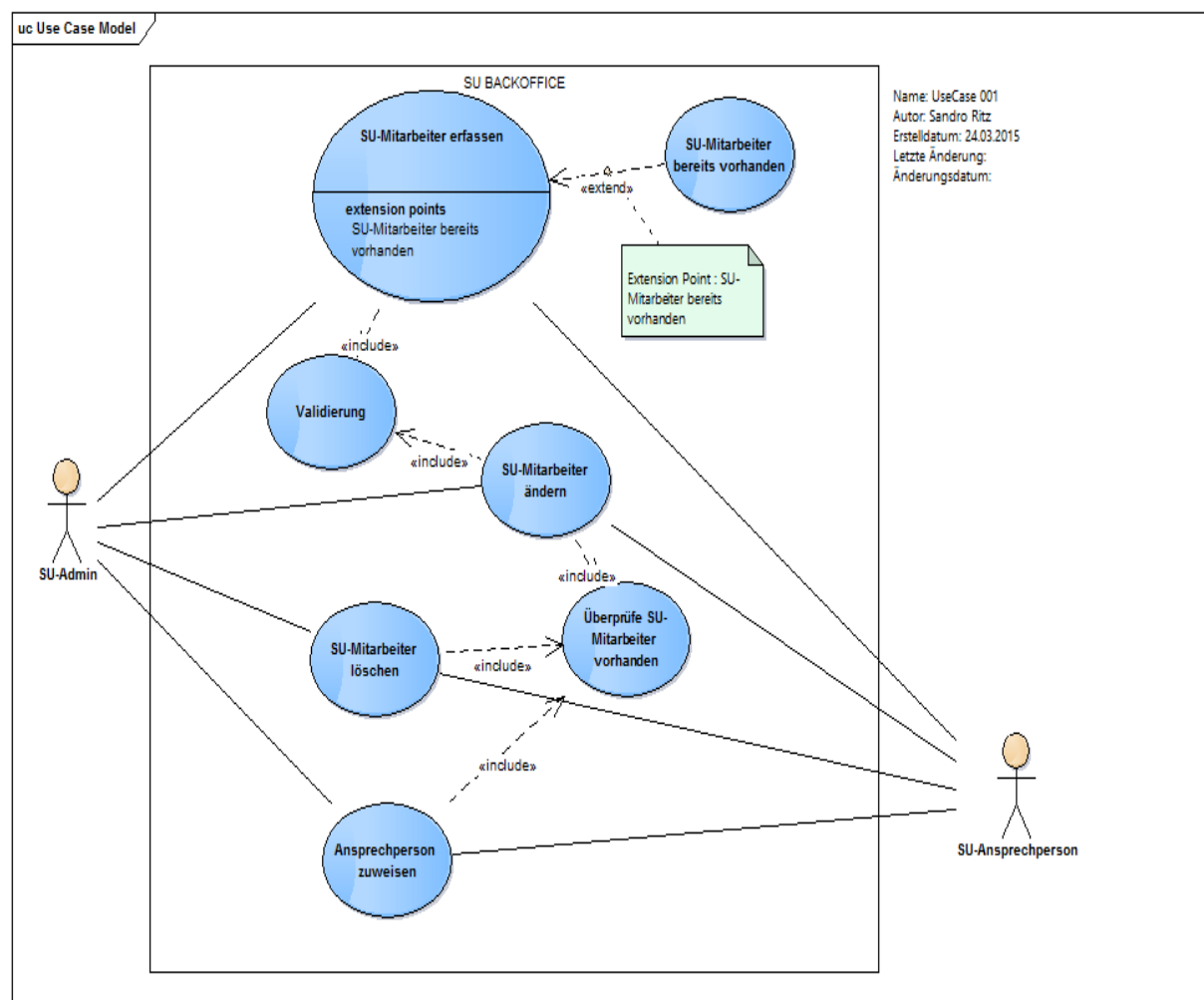


Abbildung 9

15.1.3 UseCase001 Aktivitätsdiagramm

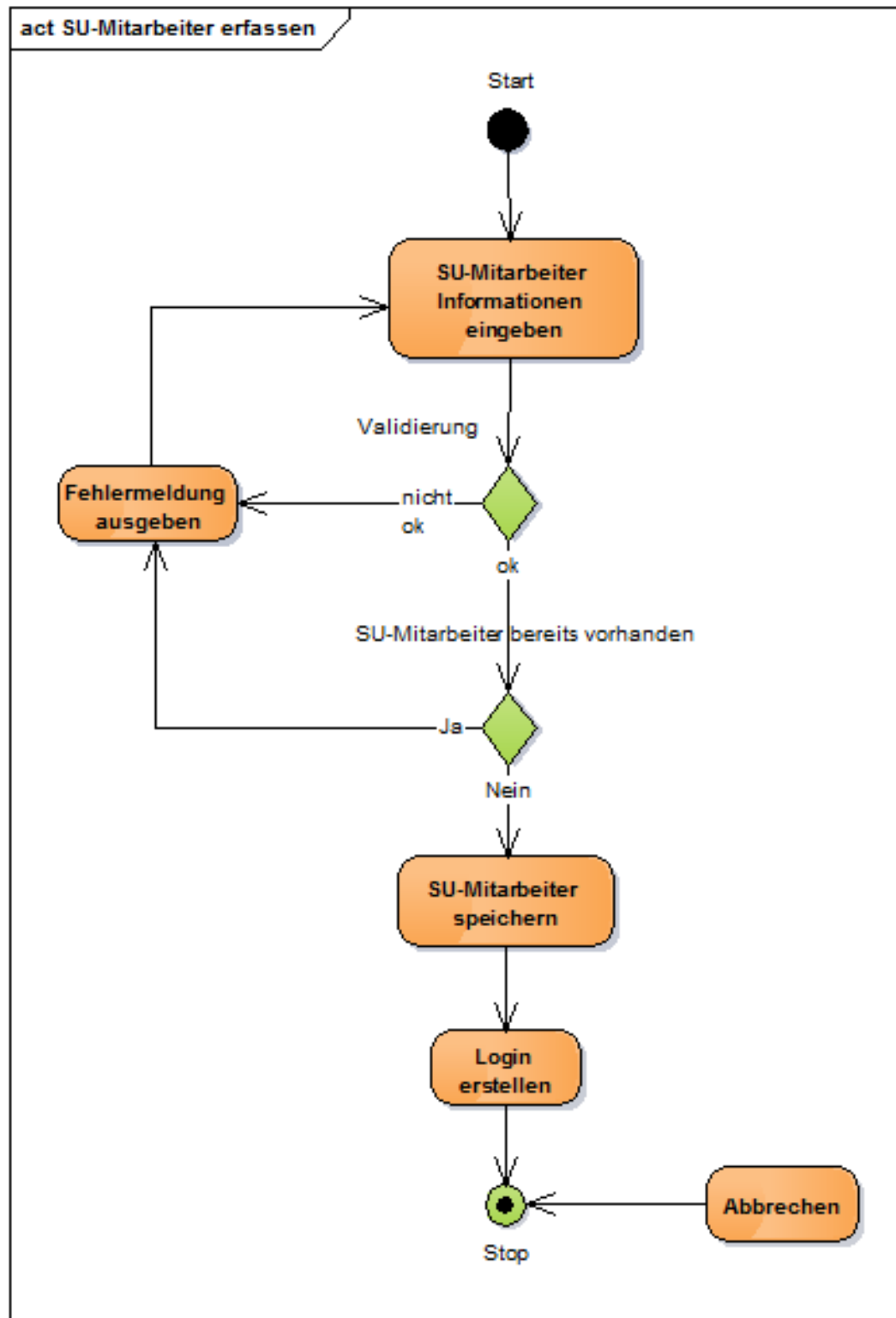


Abbildung 10

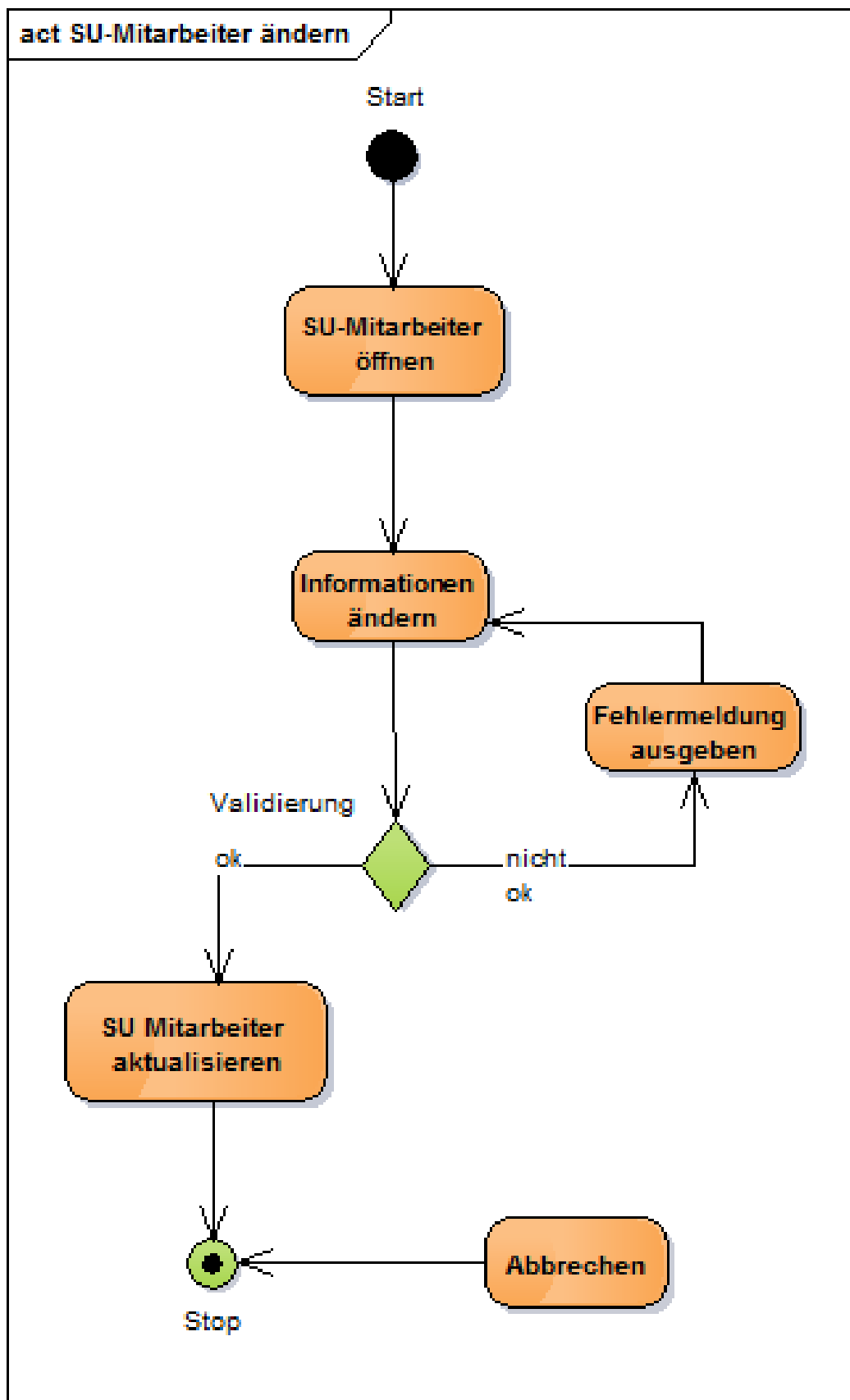


Abbildung 11

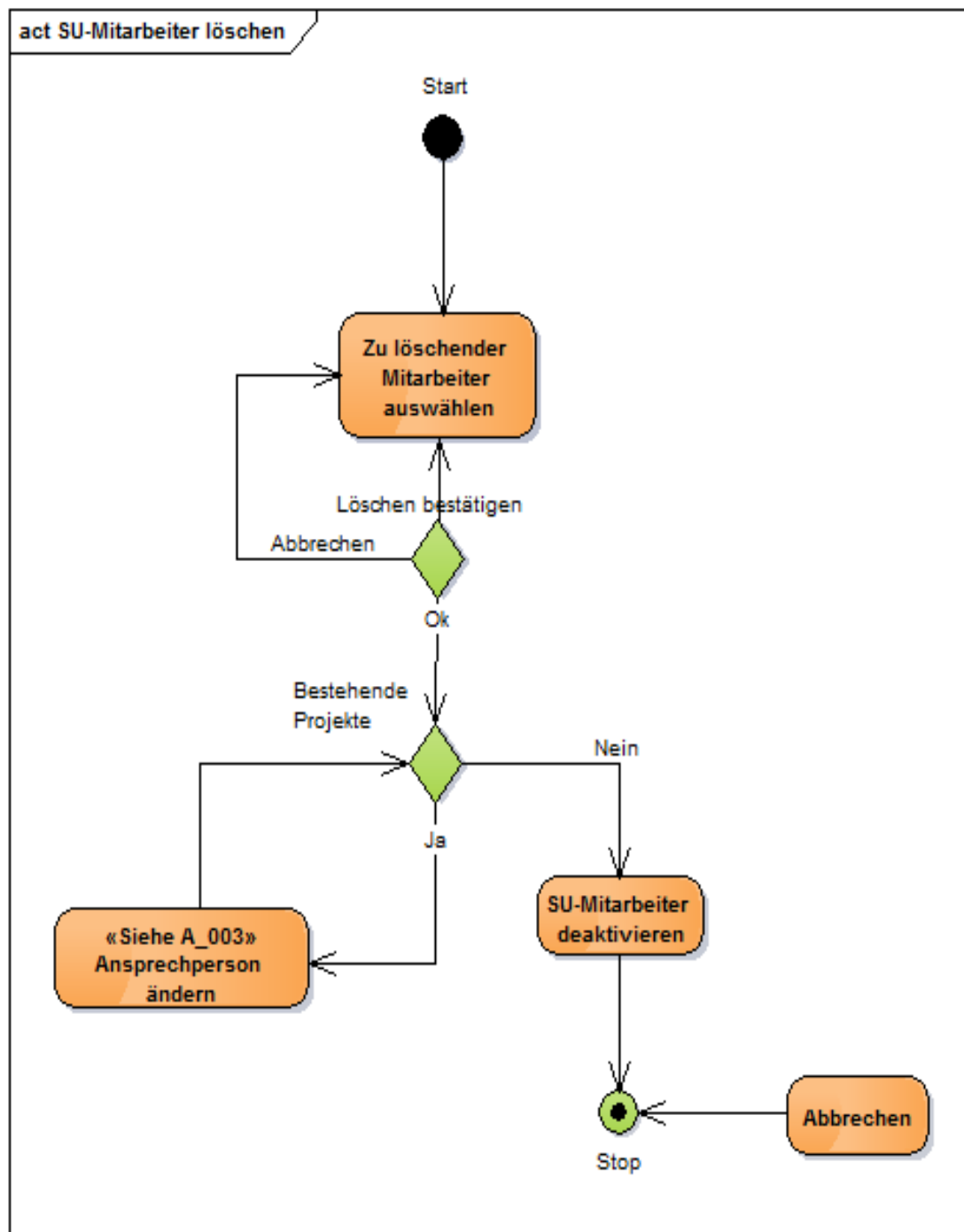


Abbildung 12

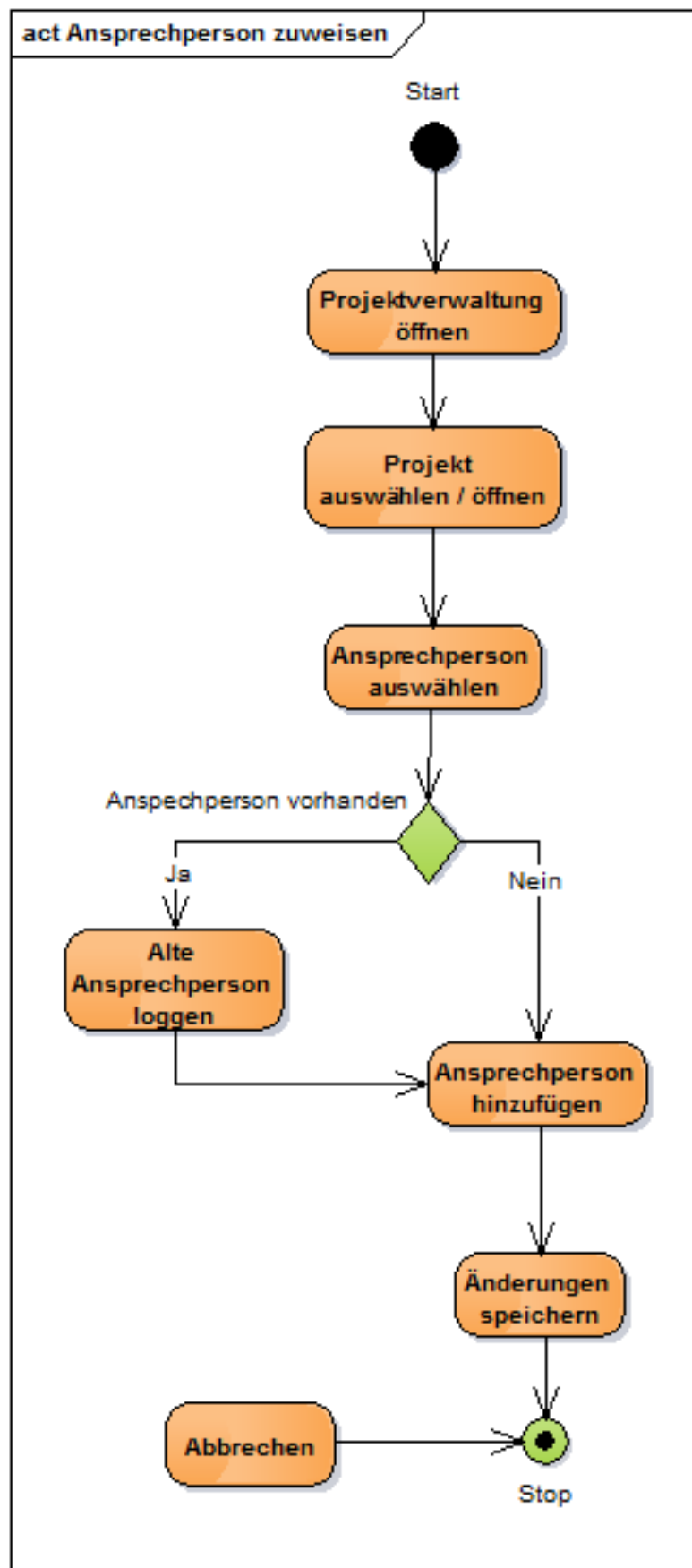


Abbildung 13

15.2 UseCase002

15.2.1 UseCase002 Beschreibung

UseCase002	
SU-Ansprechperson Datenverwaltung	
Autor	Max von Gelhorn
Ziel	Die Ansprechperson des Subunternehmens kann für die Ihm Zugeteilte Projekte die Mängel welche seine Firma betreffen einsehen, bestätigen und Kommentieren.
Kategorie	Primär, Ist essenziell für die Funktionalität des Mängelmanagers
Vorbedingungen	Es sind vom Projektleiter Mängel erfasst worden welche den SU-Anspr. Betreffen. Und welche der SU-Anspr. Noch nicht zur Kenntnis genommen hat.
Nachbedingungen Erfolg	Der SU-Anspr. Hat die Ihm Betreffenden Mängel abgearbeitet. Und in der Datenbank sind die Dazugehörigen Daten dem Entsprechen abgeändert worden.
Nachbedingungen Fehlschlag	Der SU-Anspr. Konnte keine Verbindung zum Server herstellen und konnte deshalb die abgearbeiteten Daten nicht Speichern.
Akteure	GU-Bauleiter, SU-Ansprechperson
Szenario 01 SU-Ansprechperson kann Mangel-Kennntnisnahme bestätigen SU-UI	
Auslösendes Ereignis	Der GU-Bauleiter hat einen neuen Mangel erfasst.
Beschreibung	<ol style="list-style-type: none"> 1. Der SU-Anspr. Loggt sich im Tool ein. 2. Der SU-Anspr. sieht dass es neue Mängel gibt. 3. Wenn er die Meldung gelesen hat, kann er bestätigen dass er sie gelesen hat.
Alternativen	Der SU-Anspr. Liest den Mängel aber bestätigt nicht dass er ihn gelesen hat.
Szenario 02 SU-Ansprechperson kann zugeteilte Mängel/Projekte in SU-UI einsehen	
Auslösendes Ereignis	Der SU-Anspr. Will sehen was für Mängel/Projekte er beheben muss.
Beschreibung	<ol style="list-style-type: none"> 1. Der SU-Anspr. Loggt sich im Tool ein. 2. Der SU-Anspr. sieht eine Liste mit allen Mängel/Projekte welche ihn betreffen.
Alternativen	Der SU-Anspr. Kann die Mängel/Projekte nicht einsehen da er keine Verbindung zum Server hat.
Szenario 03 SU-Ansprechperson kann Meldung zu Mangel verfassen in SU-UI	
Auslösendes Ereignis	Der SU-Anspr. Will eine Meldung zum Mängel schreiben, die der GU-Bauleiter einsehen kann.
Beschreibung	<ol style="list-style-type: none"> 1. Der SU-Anspr. Loggt sich im Tool ein. 2. Der SU-Anspr. Verfässt zu einem Mangel eine Meldung und Speichert diese.
Alternativen	Der SU-Anspr. Kann die Mängel nicht einsehen da er keine Verbindung zum Server hat.

Szenario 04 SU-Ansprechperson kann Mangel als erledigt markieren in SU-UI	
Auslösendes Ereignis	Der SU-Anspr. Hat einen Mangel behoben und möchte diesen als erledigt markieren.
Beschreibung	<ol style="list-style-type: none"> 1. Der SU-Anspr. Loggt sich im Tool ein. 2. Der SU-Anspr. Setzt den Status von einem Mangel auf „Erledigt“
Alternativen	Der SU-Anspr. Kann den Status nicht ändern da er keine Verbindung zum Server hat.

15.2.2 UseCase002 Visualisierung

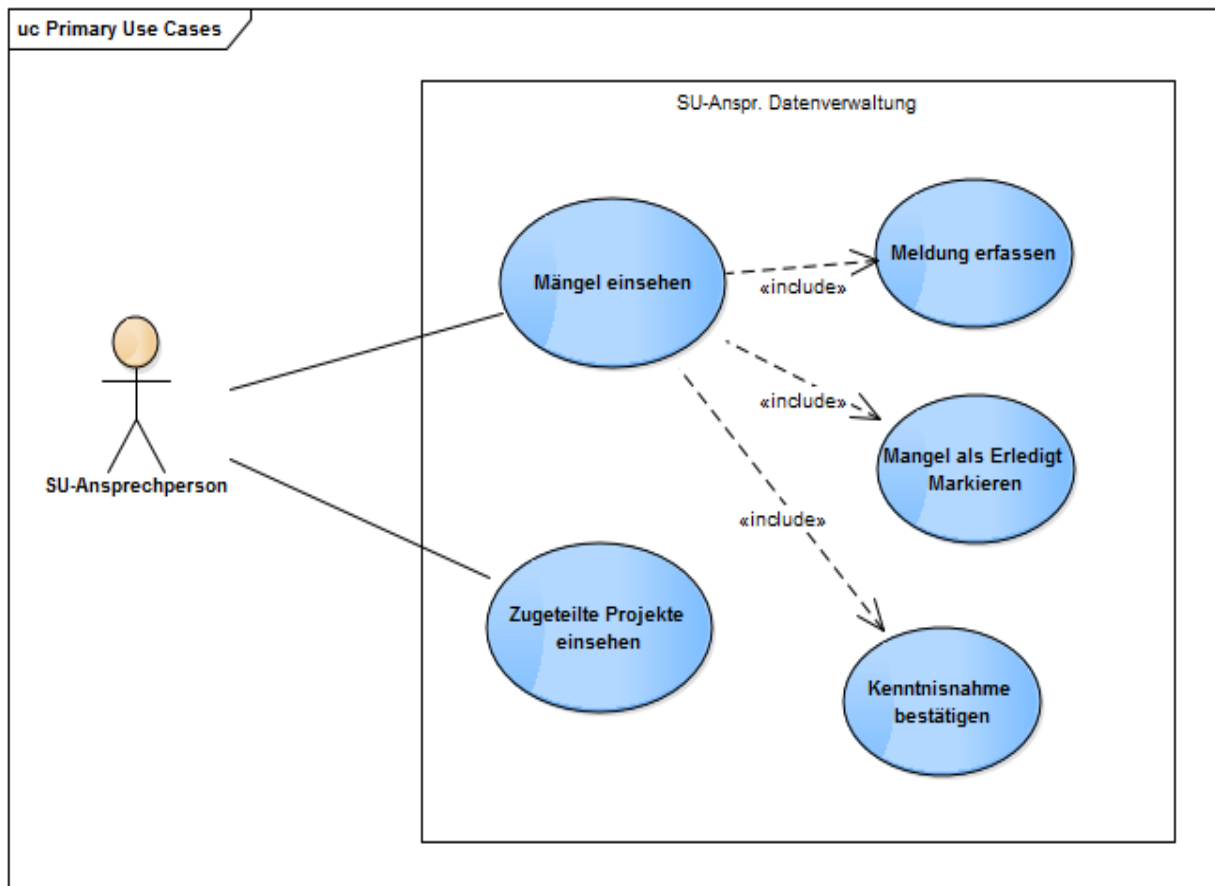


Abbildung 14

15.2.3 UseCase002 Aktivitätsdiagramm

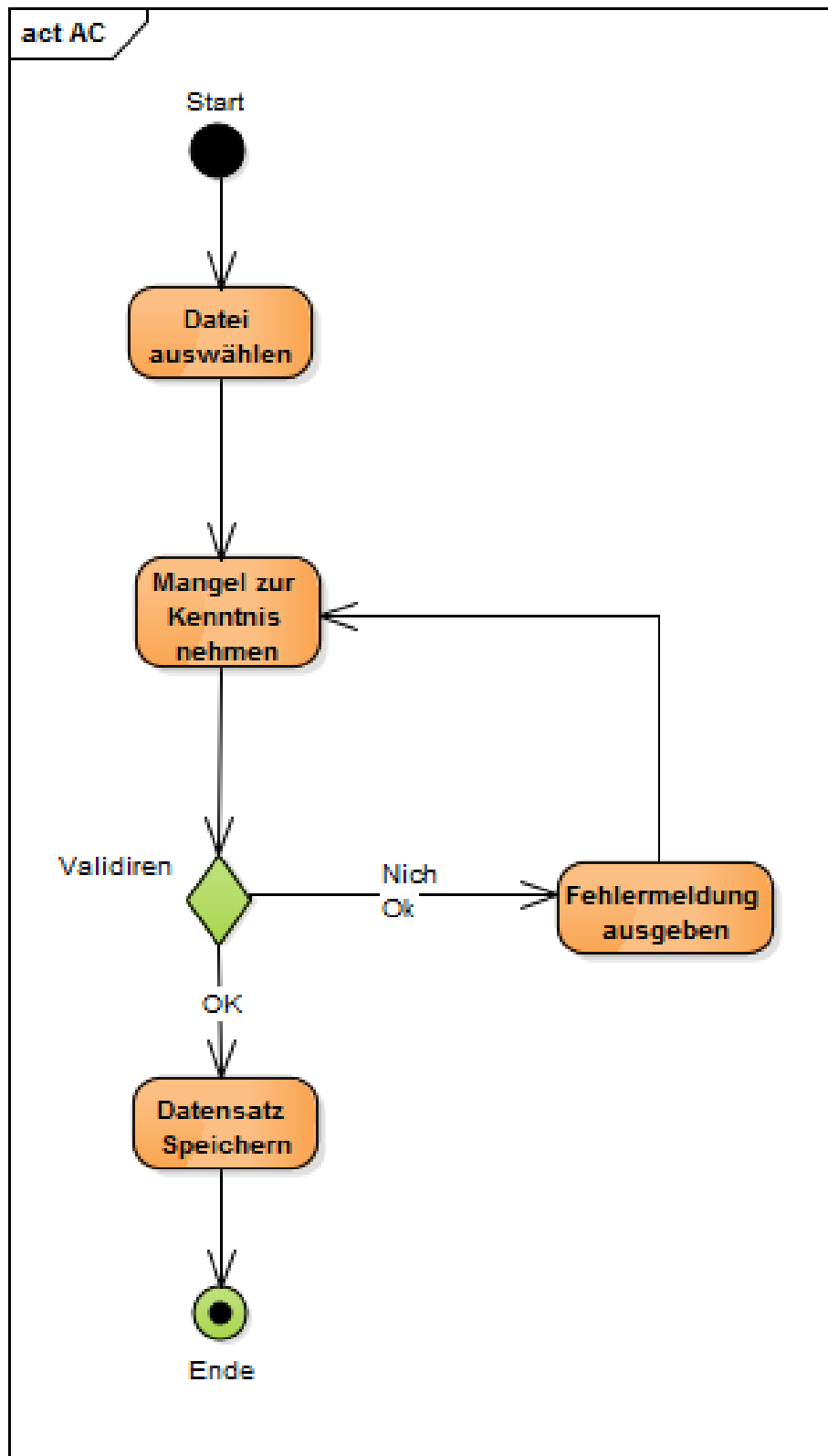


Abbildung 15

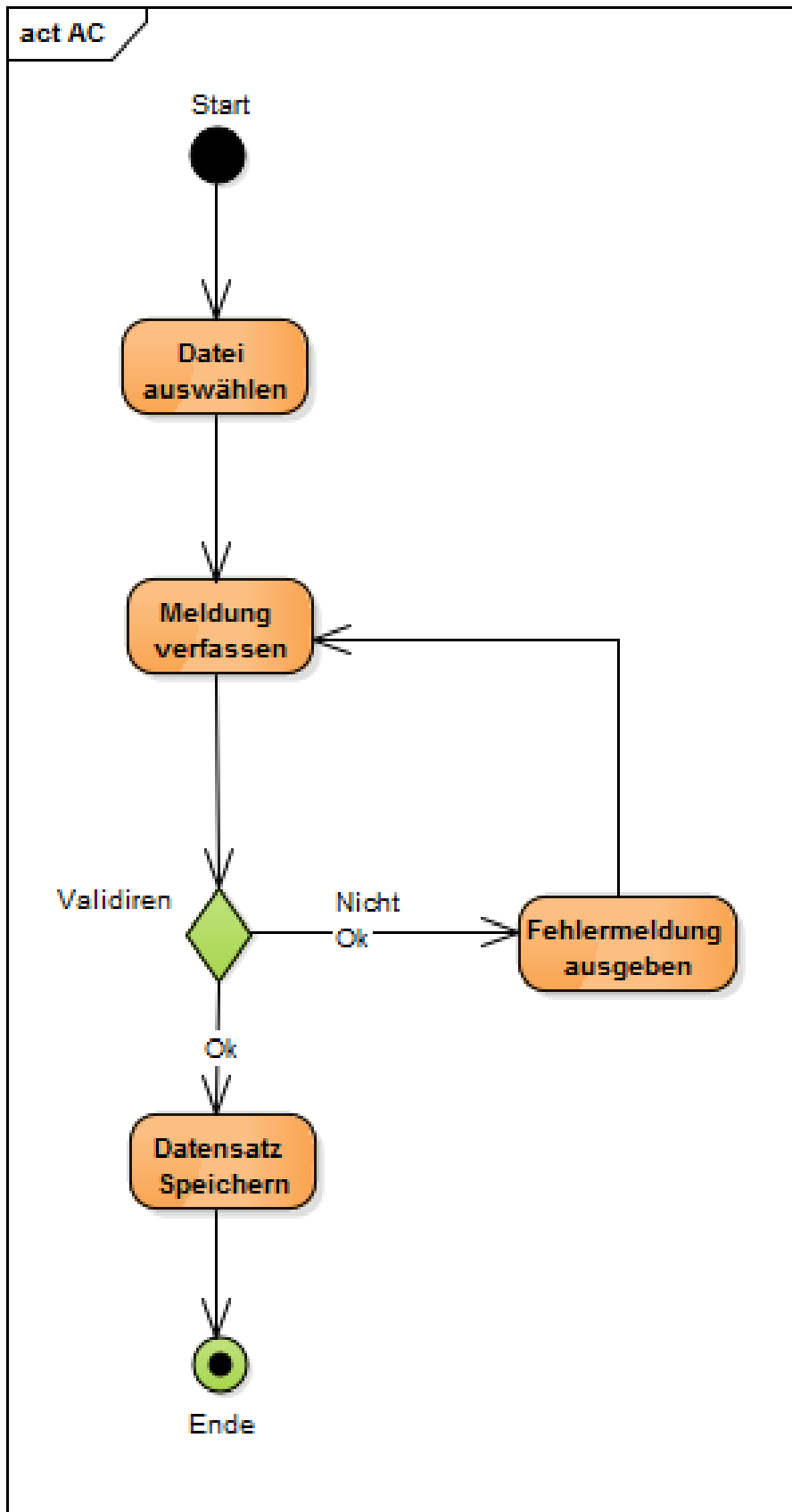


Abbildung 16

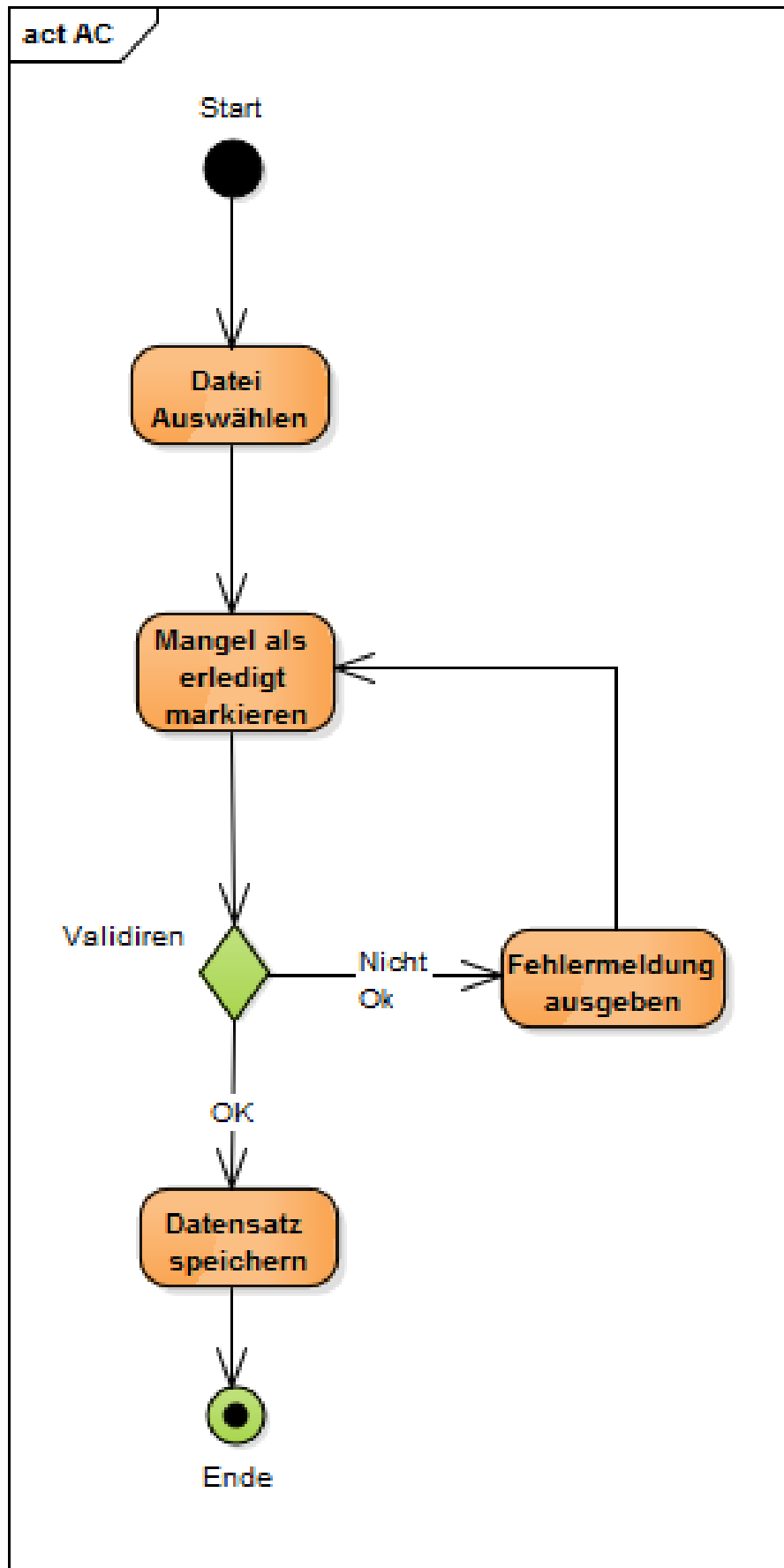


Abbildung 17

15.3 UseCase003

15.3.1 UseCase003 Beschreibung

UseCase003	SU kann Software Online Downloaden
Autor	Luca Kündig
Ziel	Mitarbeiter eines Subunternehmens können die Mängelmanager Software Downloaden
Kategorie	Sekundär, da kein direkten Einfluss auf die Funktionalität
Vorbedingungen	Der Mitarbeiter des Subunternehmens hat die Software noch nicht auf seinem PC
Nachbedingungen Erfolg	Der Mitarbeiter des Subunternehmens kann die Mängelmanager Software auf seinem PC verwenden.
Nachbedingungen Fehlschlag	Der Mitarbeiter des Subunternehmens kann die Mängelmanager Software nicht downloaden, oder der Download ist fehlerhaft.
Akteure	Subunternehmen Ansprechperson, Subunternehmen Backoffice
Auslösendes Ereignis	Neues Subunternehmen wird erfasst und beteiligt sich an einem Projekt
Beschreibung	<ol style="list-style-type: none"> 1. Der Subunternehmen Mitarbeiter verbindet sich auf die Download Seite 2. Der Subunternehmen Mitarbeiter lädt sich die Datei auf seinen PC. 3. Download ist erfolgreich abgeschlossen 4. Der Subunternehmen Mitarbeiter führt die Software aus
Erweiterung	
Alternativen	1a Es gibt keine Download Seite: Die Datei wird via externes Medium (CD, Stick,...) Verbreitet.



15.3.2 UseCase003 Visualisierung

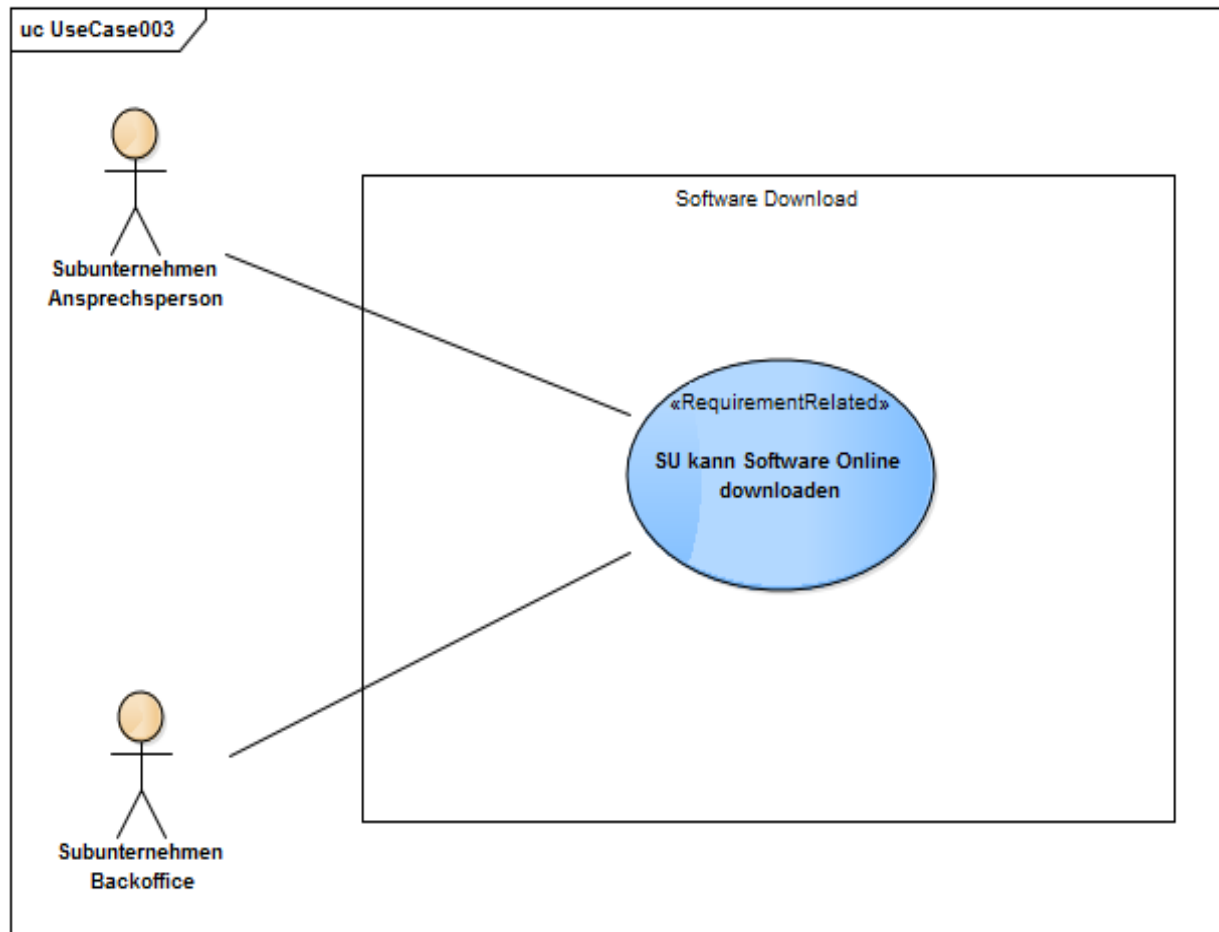


Abbildung 18

15.3.3 UseCase003 Aktivitätsdiagramm

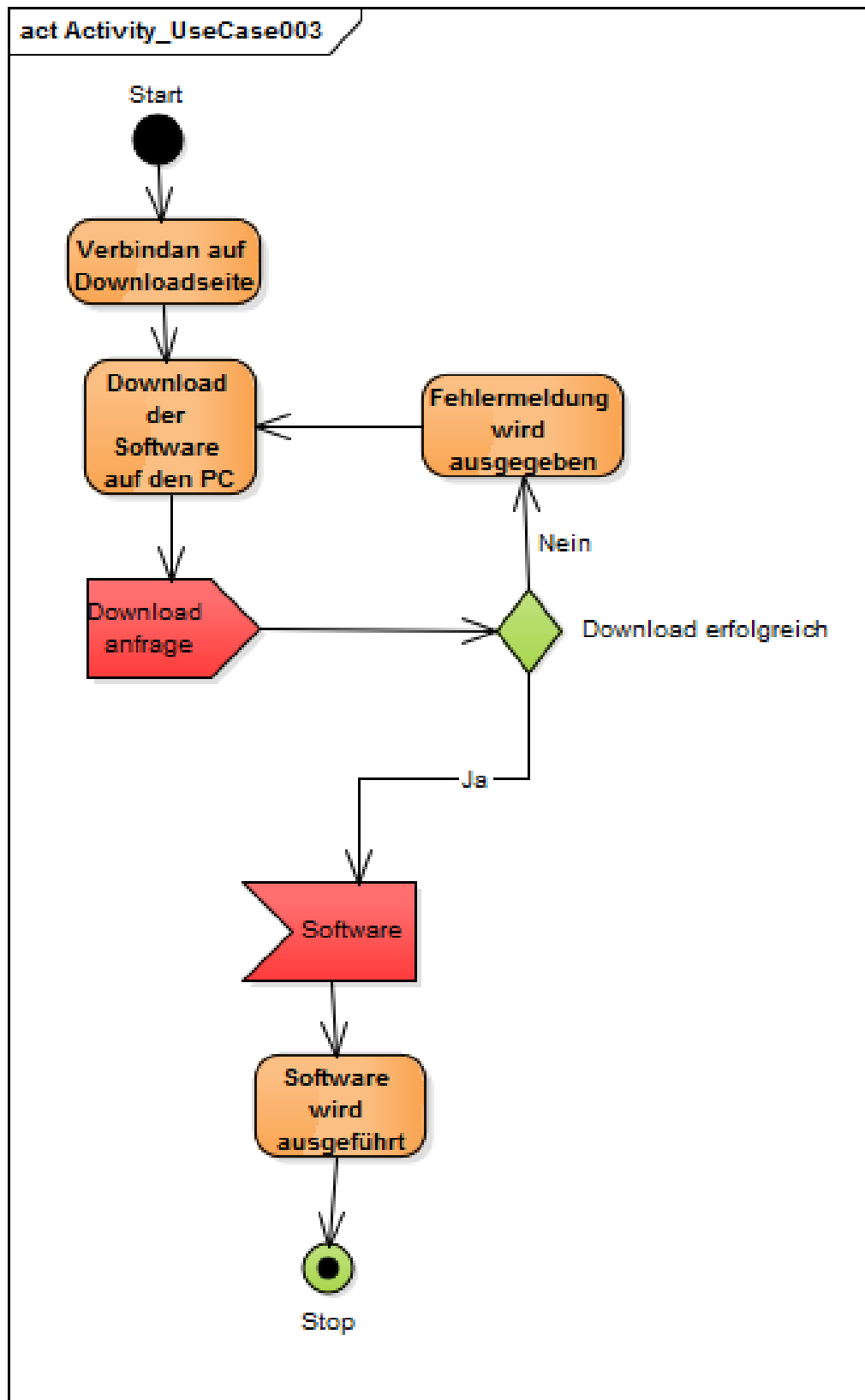


Abbildung 19

15.4 UseCase004

15.4.1 UseCase004 Beschreibung

UseCase004	
GU-BackOffice Datenverwaltung	
Autor	Luca Kündig
Ziel	Der BackOffice-Mitarbeiter der Unternehmung kann Einträge in der Datenbank sowohl erfassen, löschen, ändern und lesen
Kategorie	Primär, Ist essenziell für die Funktionalität des Mängelmanagers
Vorbedingungen	Daten sind nicht vorhanden die erfasst werden müssen und die Daten die geändert, gelöscht oder gelesen werden müssen sind im System vorhanden
Nachbedingungen Erfolg	Daten sind im System erfasst, geändert oder gelöscht worden
Nachbedingungen Fehlschlag	Die Daten sind fehlerhaft erfasst, geändert, nicht gelöscht
Akteure	GU-BackOffice, GU-Bauleiter, SU-BackOffice, Bauherr
Szenario 01	
Auslösendes Ereignis	Neuer Bauleiter tritt in die Unternehmung ein
Beschreibung	<ol style="list-style-type: none"> 1. BackOffice Mitarbeiter überprüft ob der Bauleiter bereits im System ist 2. BackOffice Mitarbeiter öffnet Maske zum Erfassen eines neuen Bauleiters 3. BackOffice Mitarbeiter erfasst den Bauleiter mit ID, Name, Vorname, Adresse, Rolle, Login-Daten im Mängelmanager 4. BackOffice Mitarbeiter klickt auf Save und speichert so den neuen Benutzer 5. Die Benutzerdaten werden in der Datenbank gespeichert
Erweiterung	1a Benutzer existiert bereits: Nichts ist zu tun
Alternativen	2a Der neue Bauleiter erfasst seinen User selber
Szenario 02	
Auslösendes Ereignis	Bauherr ändert seine Adresse
Beschreibung	<ol style="list-style-type: none"> 1. BackOffice Mitarbeiter überprüft ob der Bauherr bereits im System ist 2. BackOffice Mitarbeiter öffnet den Datensatz des SU-BackOffice Mitarbeiter 3. BackOffice Mitarbeiter ersetzt die alte Adresse des Bauherren im Mängelmanager 4. BackOffice Mitarbeiter klickt auf Save und speichert so die neue Adresse 5. Die Adresse des Bauherren wird in der Datenbank gespeichert
Erweiterung	1a Bauherr existiert noch nicht: Bauherr wird via Szenario 01 erfasst
Alternativen	-
Szenario 03	
Auslösendes Ereignis	Bauherr beendet Zusammenarbeit mit GU, das Projekt wird gelöscht

Beschreibung	<ol style="list-style-type: none"> 1. BackOffice Mitarbeiter überprüft ob das Projekt im System vorhanden ist 2. BackOffice Mitarbeiter öffnet den Datensatz des Projekts 3. BackOffice Mitarbeiter klickt auf den delete Button im Mängelmanager 4. BackOffice Mitarbeiter klickt auf bestätigen Button im Mängelmanager 5. Die Projektdaten werden aus der Datenbank gelöscht
Alternativen	Status des Projekt wird auf abgebrochen geändert

15.4.2 UseCase004 Visualisierung

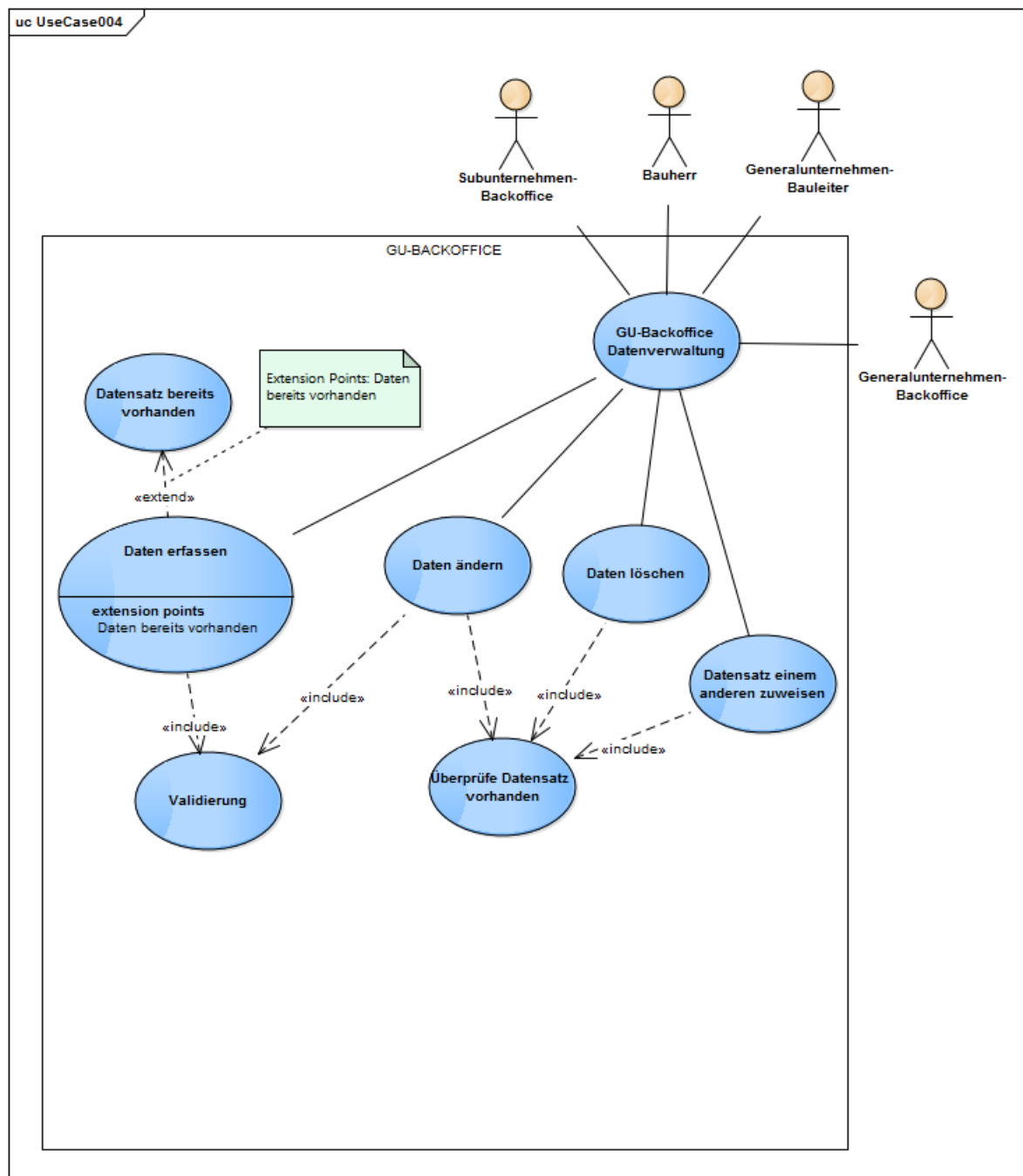


Abbildung 20

15.4.3 UseCase004 Aktivitätsdiagramm

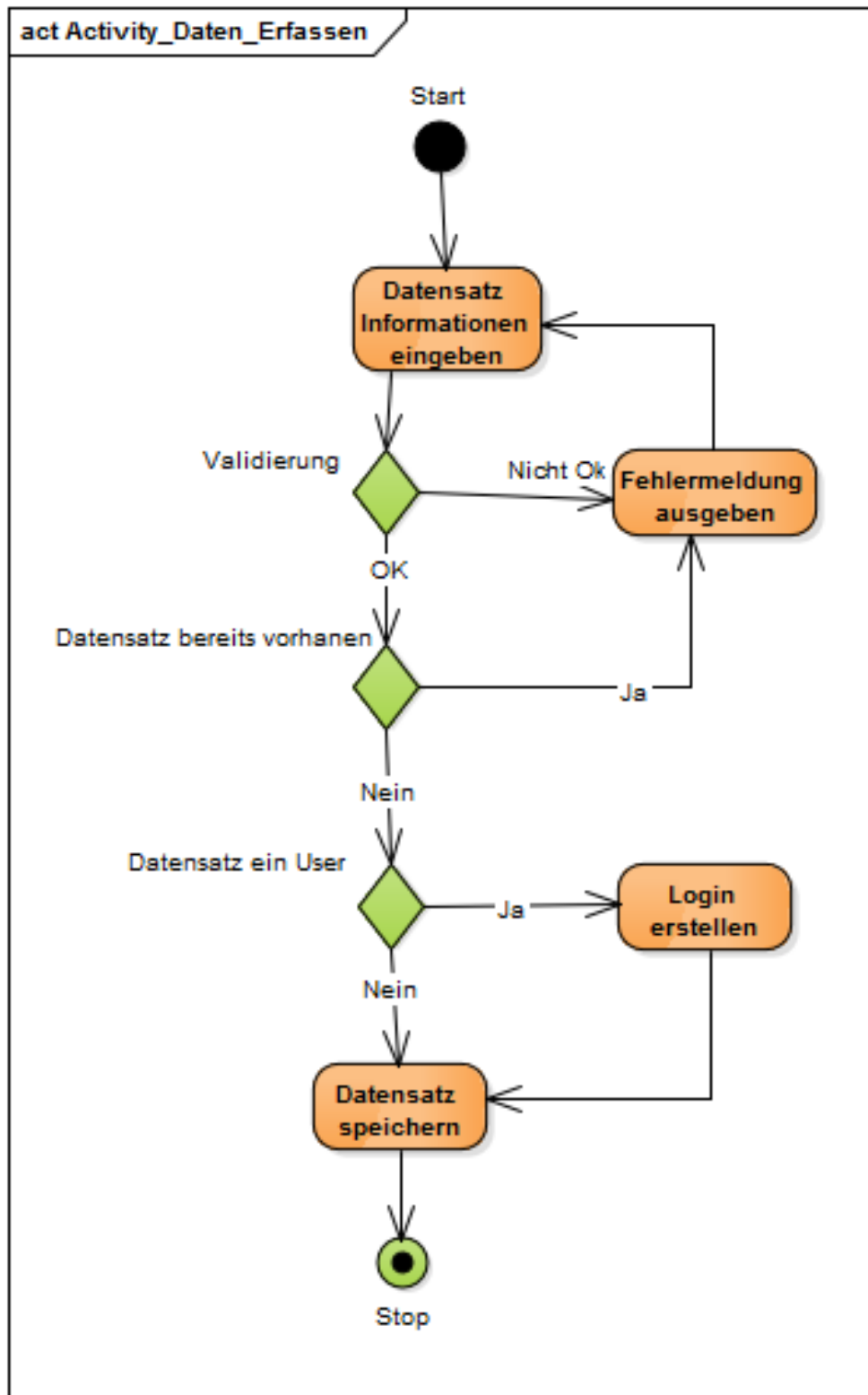


Abbildung 21

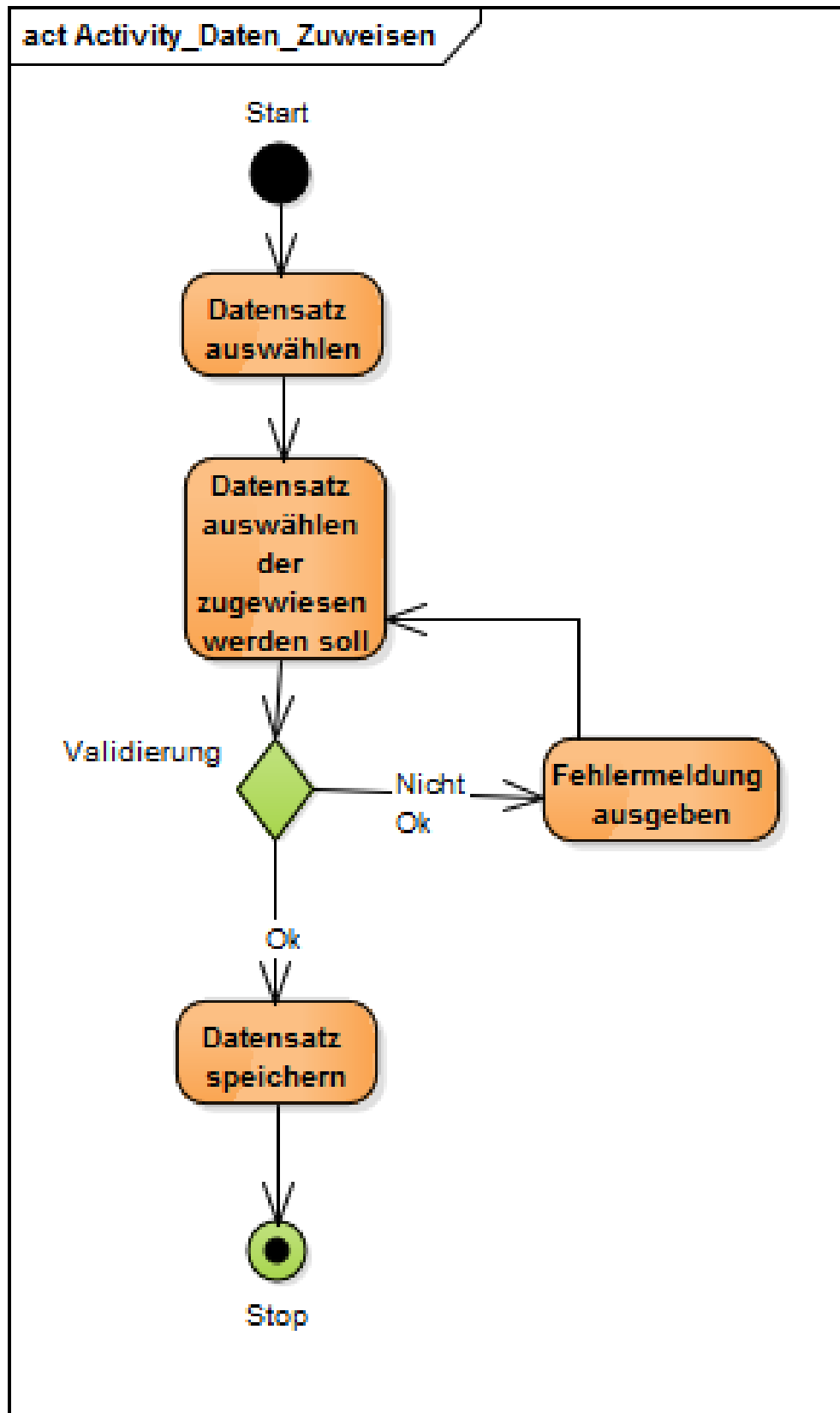


Abbildung 22

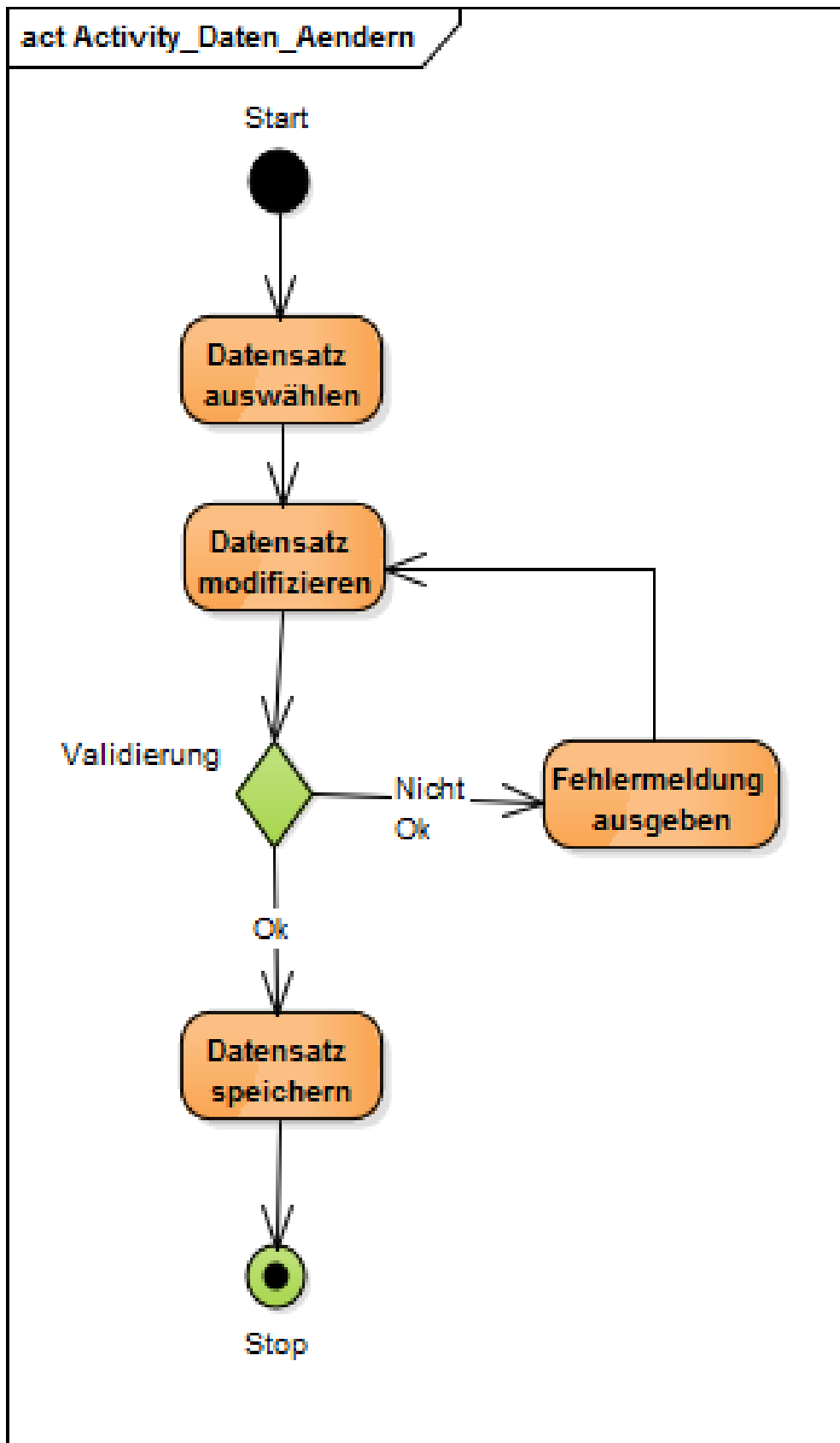


Abbildung 23

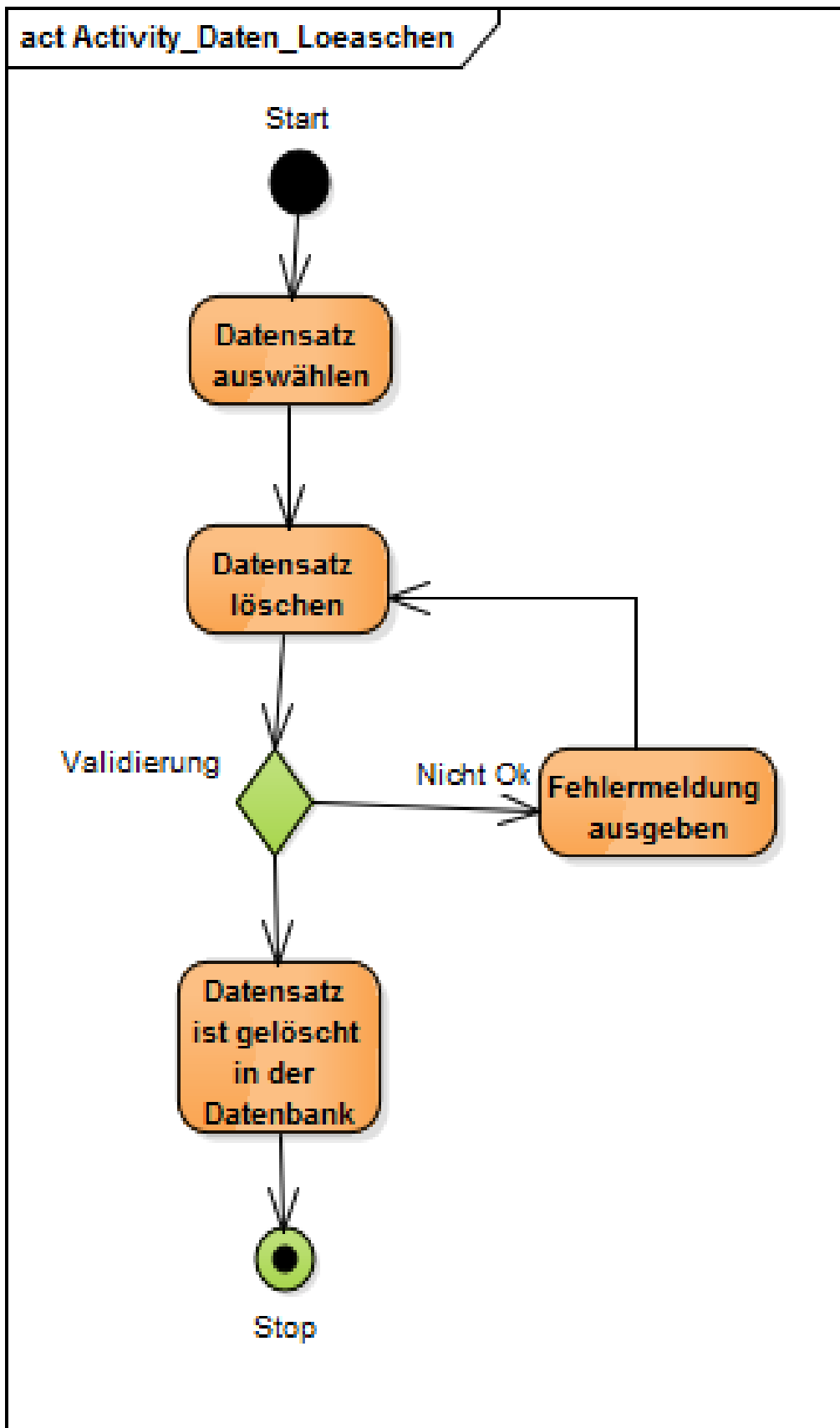


Abbildung 24

15.5 UseCase005

15.5.1 UseCase005 Beschreibung

UseCase005	GU-Bauleiter Datenverwaltung
Autor	Mike Monticoli
Ziel	GU-Bauleiter kann Mängel und Meldungen erfassen. Er kann alle Meldungen und Mängel zu seinem Projekt einsehen. Er kann die Mängelbehebung bestätigen. Er sieht all seine Projekte im GU-UI.
Kategorie	Primär, ist essenziell für die Funktionalität des Mängelmanagers
Vorbedingungen	Ein Mangel / Eine Meldung ist noch nicht erfasst oder muss bearbeitet werden
Nachbedingungen Erfolg	Der GU-Bauleiter konnte die Mängel und Meldungen erfassen beziehungsweise die gewünschte Mängelbehebung bestätigen.
Nachbedingungen Fehlschlag	Die Mängel und Meldungen wurden fehlerhaft erfasst oder die Mängelbehebung konnte nicht bestätigt werden.
Akteure	GU-Bauleiter, SU-Ansprechperson
Szenario 1	
Auslösendes Ereignis	GU-Bauleiter will neuen Mangel erfassen.
Beschreibung	GU-Bauleiter öffnet Projekt in GU-UI. GU-Bauleiter erfasst neuen Mangel im GU-UI. GU-Bauleiter füllt die Daten zum Mangel aus. GU-Bauleiter bestätigt, dass die Daten korrekt erfasst wurden.
Szenario 2	
Auslösendes Ereignis	GU-Bauleiter will eine neue Meldung erfassen.
Beschreibung	GU-Bauleiter erfasst neue Meldung im GU-UI. GU-Bauleiter füllt die Daten zur Meldung aus. GU-Bauleiter bestätigt, dass die Daten korrekt erfasst wurden.
Szenario 3	
Auslösendes Ereignis	GU-Bauleiter will Meldung oder Mangel einsehen
Beschreibung	GU-Bauleiter öffnet das betroffene Projekt im GU-UI. GU-Bauleiter öffnet die gewünschte Meldung beziehungsweise den gewünschten Mangel
Szenario 4	
Auslösendes Ereignis	GU-Bauleiter will die Behebung eines Mangels bestätigen.
Beschreibung	GU-Bauleiter öffnet das betroffene Projekt im GU-UI. GU-Bauleiter öffnet den behobenen Mangel GU-Bauleiter bestätigt die Mangelbehebung
Szenario 5	
Auslösendes Ereignis	GU-Bauleiter will seine Projekt im GU-UI einsehen

Beschreibung	<p>GU-Bauleiter startet das GU-UI.</p> <p>GU-Bauleiter öffnet das gewünschte Projekt.</p> <p>GU-Bauleiter sieht die gewünschten Daten ein.</p>
Erweiterung	<p>Fehlermeldung bei:</p> <p>Mangel ist bereits erfasst.</p> <p>Mangel ist bereits behoben aber noch nicht entfernt.</p> <p>Meldung ist bereits erfasst.</p> <p>Meldung ist bereits gelöscht.</p> <p>Projekt ist nicht vorhanden.</p>
Alternativen	SU erfasst Mängel und kümmert sich um die Behebung dieser.

15.5.2 UseCase005 Visualisierung

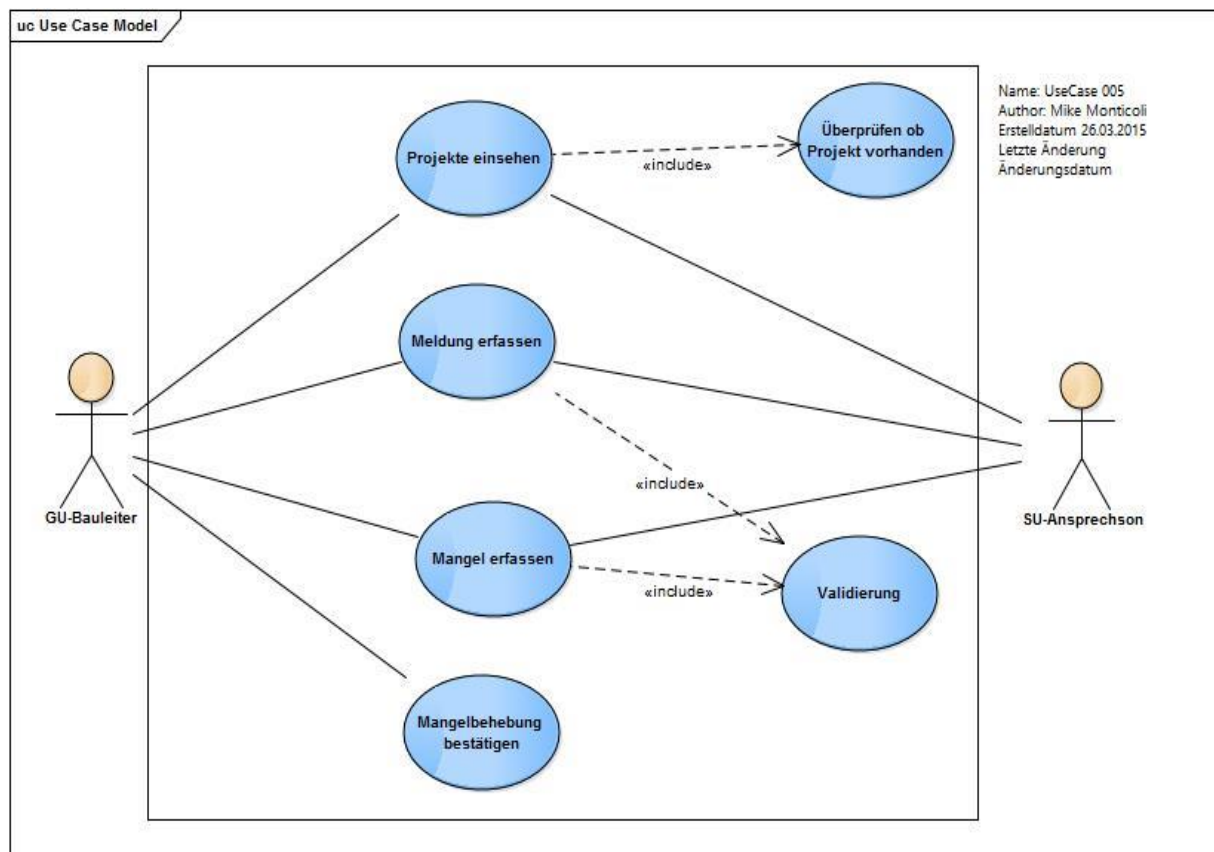


Abbildung 25

15.5.3 UseCase005 Aktivitätsdiagramm

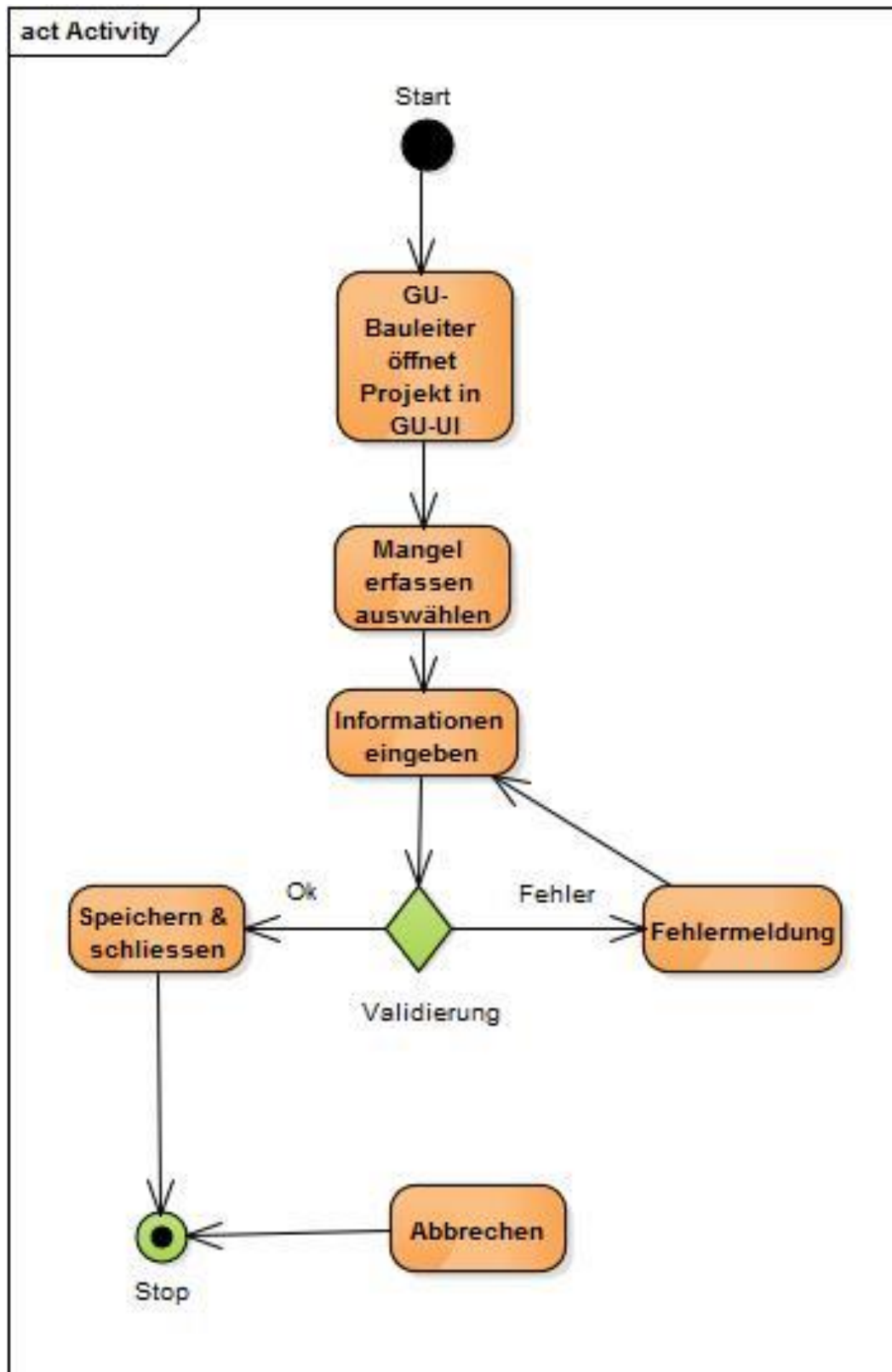


Abbildung 26

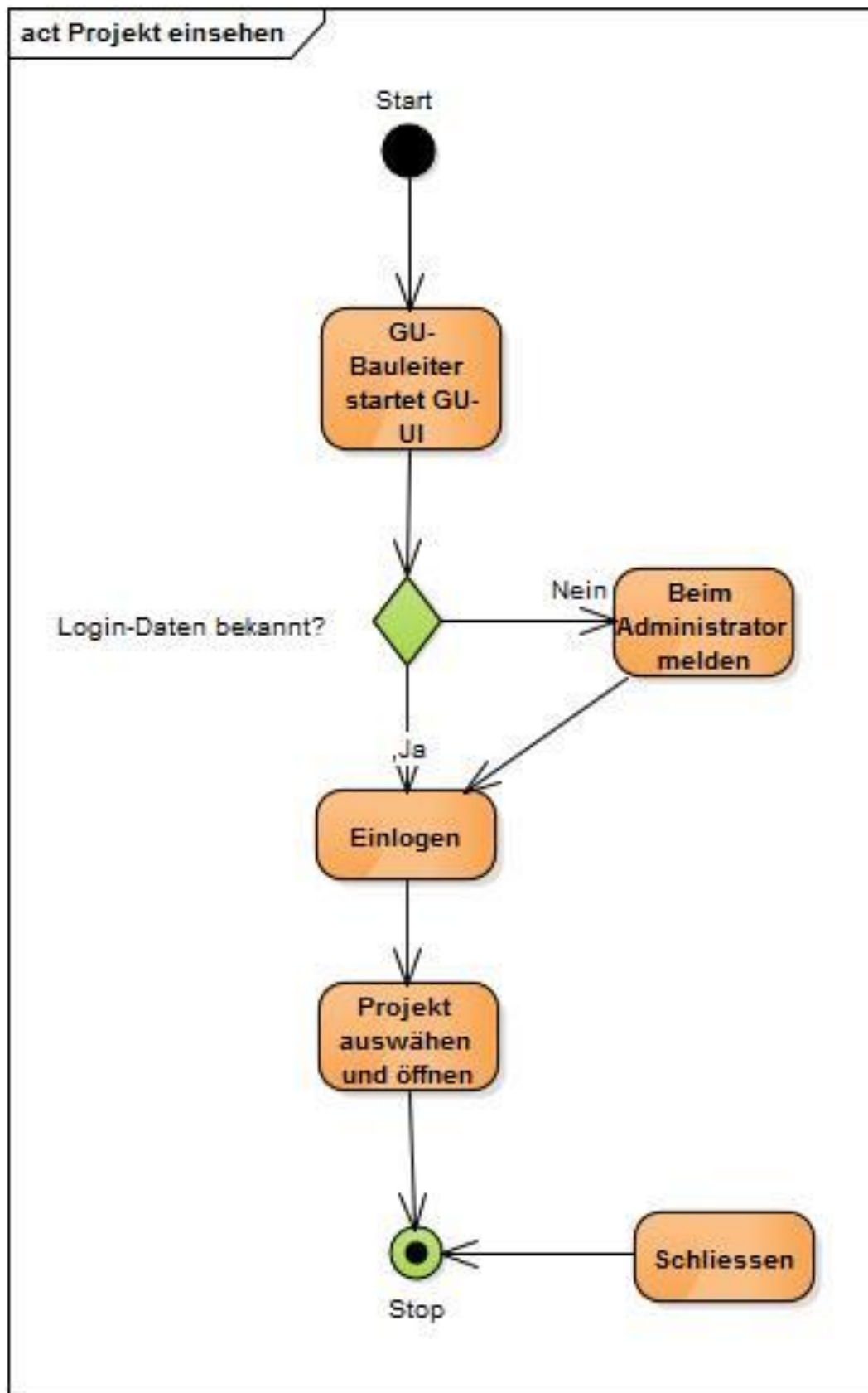


Abbildung 27

15.6 UseCase006

15.6.1 UseCase006 Beschreibung

UseCase006	User logt sich auf UI ein
Autor	Cihan Demir
Ziel	Beliebige User können sich anhand eines graphical user interface's einloggen.
Kategorie	Primär
Vorbedingung	User muss gültige Logindaten (Username, Password) aufweisen. Zusätzlich muss eine aktive Verbindung zwischen UI und Applikationsserver bestehen.
Nachbedingungen Erfolg	Jener User kann anhand den gegebenen Rechten in die Datenbank einsehen und kann die betreffenden Daten nach Bedarf und Möglichkeit verwalten.
Nachbedingungen Fehlschlag	Zugang zur Applikation / Datenbank wurde verweigert.
Akteure	GU-Admin, GU-Bauleiter, GU-Backoffice, GU-User, SU-Admin, SU-Ansprechperson
Auslösendes Ereignis	Beliebiger User muss sich aufgrund eines Geschäftsfalles mit der Applikation / Datenbank verbinden.
Szenario 1	
Beschreibung	<ol style="list-style-type: none"> 1. Der User loggt sich anhand des user interface's ein. 2. Der User erledigt die anstehenden Arbeiten. 3. Der User loggt sich nach erfolgreicher Erledigung der Arbeiten oder infolge Nichtweitergebrauchs, aus.
Szenario 2	
Auslösendes Ereignis	Beliebiger User muss sich aufgrund eines Geschäftsfalles mit der Applikation / Datenbank verbinden.
Beschreibung	<ol style="list-style-type: none"> 1. Der User versucht sich anhand des user interface's einzuloggen. 2. Der User kann sich nicht einloggen, da fehlerhafte Login Daten vorliegen. 3. Der User wählt den Link: „Passwort vergessen“ 4. Der User gibt entweder seine E-Mail-Adresse oder den Benutzernamen ein. 5. Der User wählt den Link in seiner E-Mail um sein Passwort zurückzusetzen. 6. Der User loggt sich anhand des user interface's ein. 7. Der User ändert sein Passwort. 8. Der User erledigt die anstehenden Arbeiten. 9. Der User loggt sich nach erfolgreicher Erledigung der Arbeiten oder infolge Nichtweitergebrauchs, aus.
Erweiterung	Der User wird automatisch nach einer gewissen Zeit angesichts Inaktivität ausgeloggt.



15.6.2 UseCase006 Visualisierung

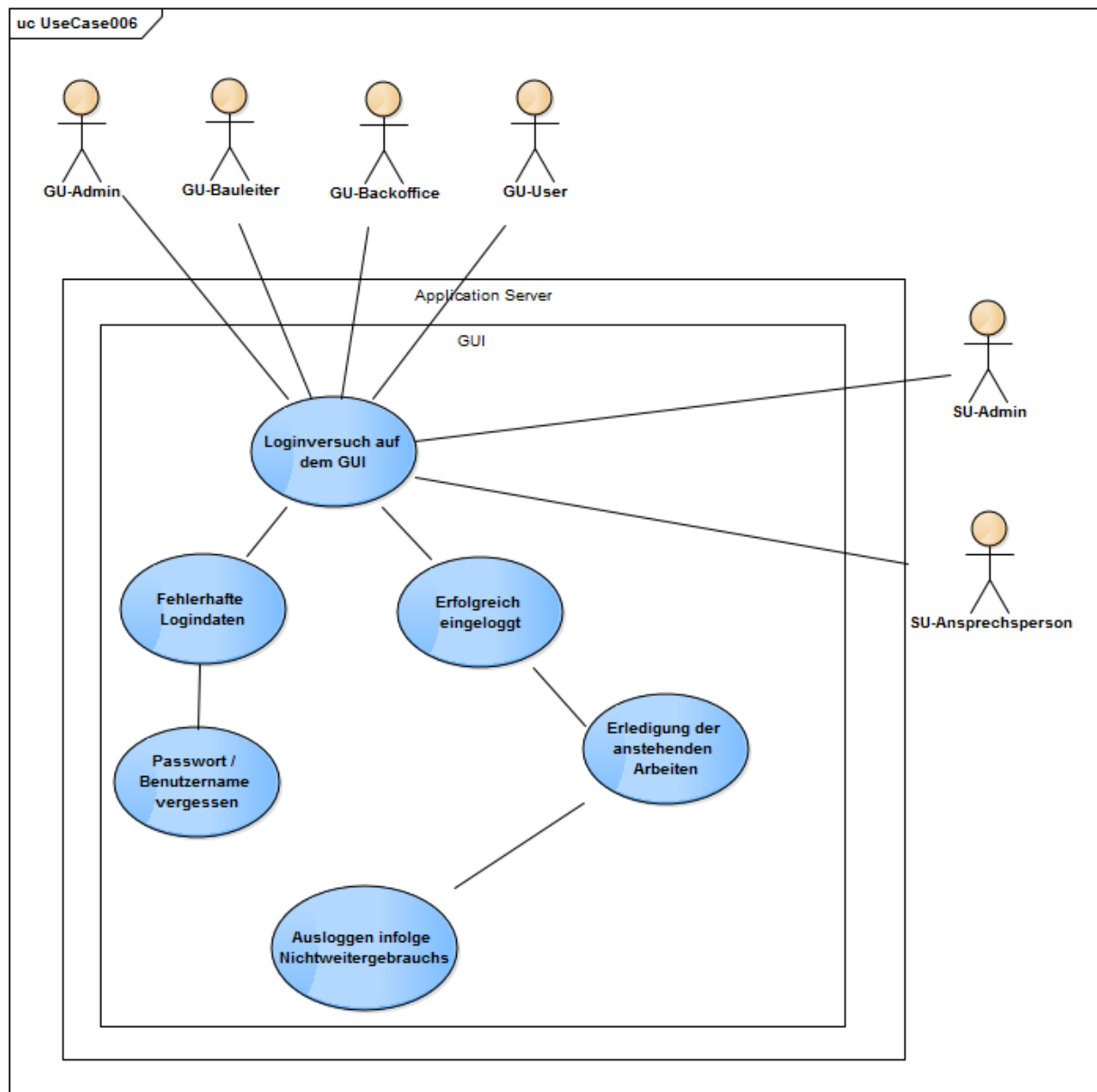


Abbildung 28

15.6.3 UseCase006 Aktivitätsdiagramm

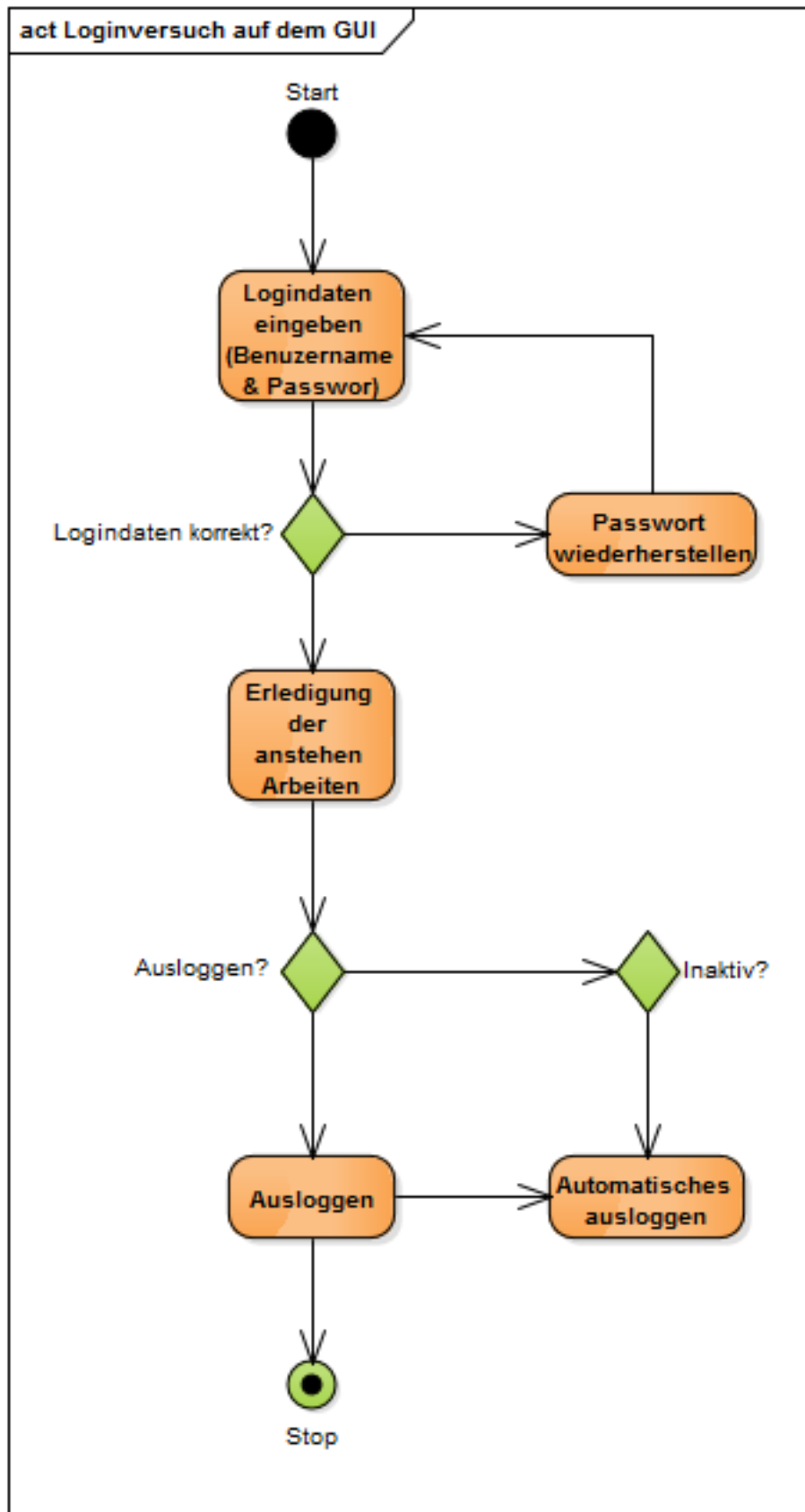


Abbildung 29

15.7 UseCase007

15.7.1 UseCase007 Beschreibung

UseCase007	Mängelliste ausdrucken
Autor	Sandro Ritz
Ziel	Der Ansprechperson der Subunternehmen kann von seinen Projekten die Mängelliste ausdrucken.
Kategorie	Sekundär, der Mangelmanager funktioniert auch ohne entsprechende Funktion.
Vorbedingungen Erfolg	Es müssen offene Mängel vorhanden sein.
Nachbedingungen Fehlschlag	PDF mit Mängel wird generiert.
Akteure	SU-Admin, SU-Ansprechperson
Auslösendes Ereignis	SU-Ansprechperson möchte eine Liste mit allen offenen Mängeln ausdrucken.
Beschreibung	<ol style="list-style-type: none"> 1. Die Ansprechperson öffnet ein ausgewähltes Projekt. 2. Durch die Filterfunktion kann die Ansprechperson die gewünschten Mängel anzeigen lassen. 3. Ein Klick auf „Drucken“ generiert ihm eine PDF Datei mit den vorhin ausgewählten gefilterten Mängeln.
Erweiterung	1a Auswählbar ob ein PDF geöffnet oder direkt gedruckt werden soll.



15.7.2 UseCase007 Visualisierung

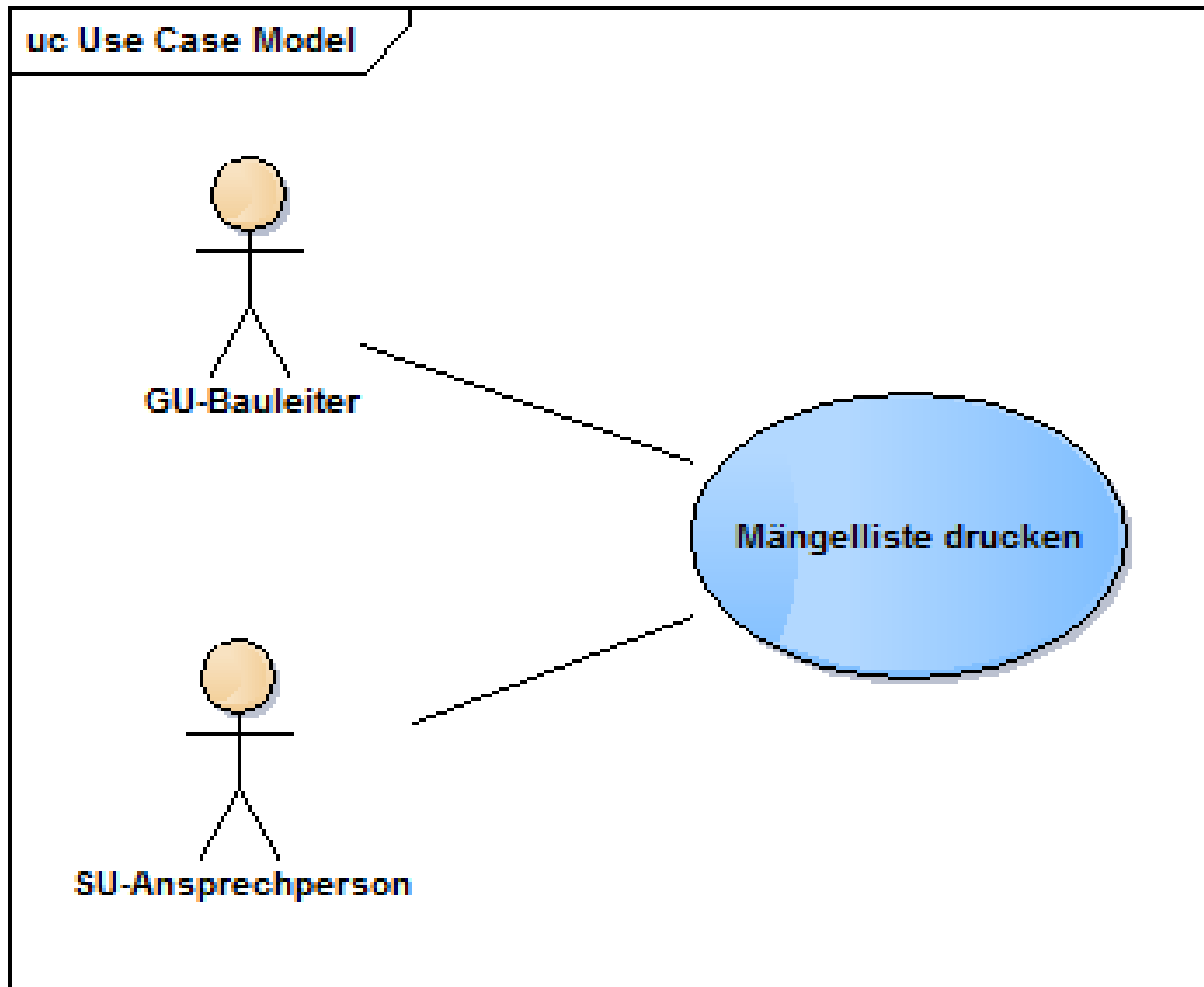


Abbildung 30

15.7.3 UseCase007 Aktivitätsdiagramm

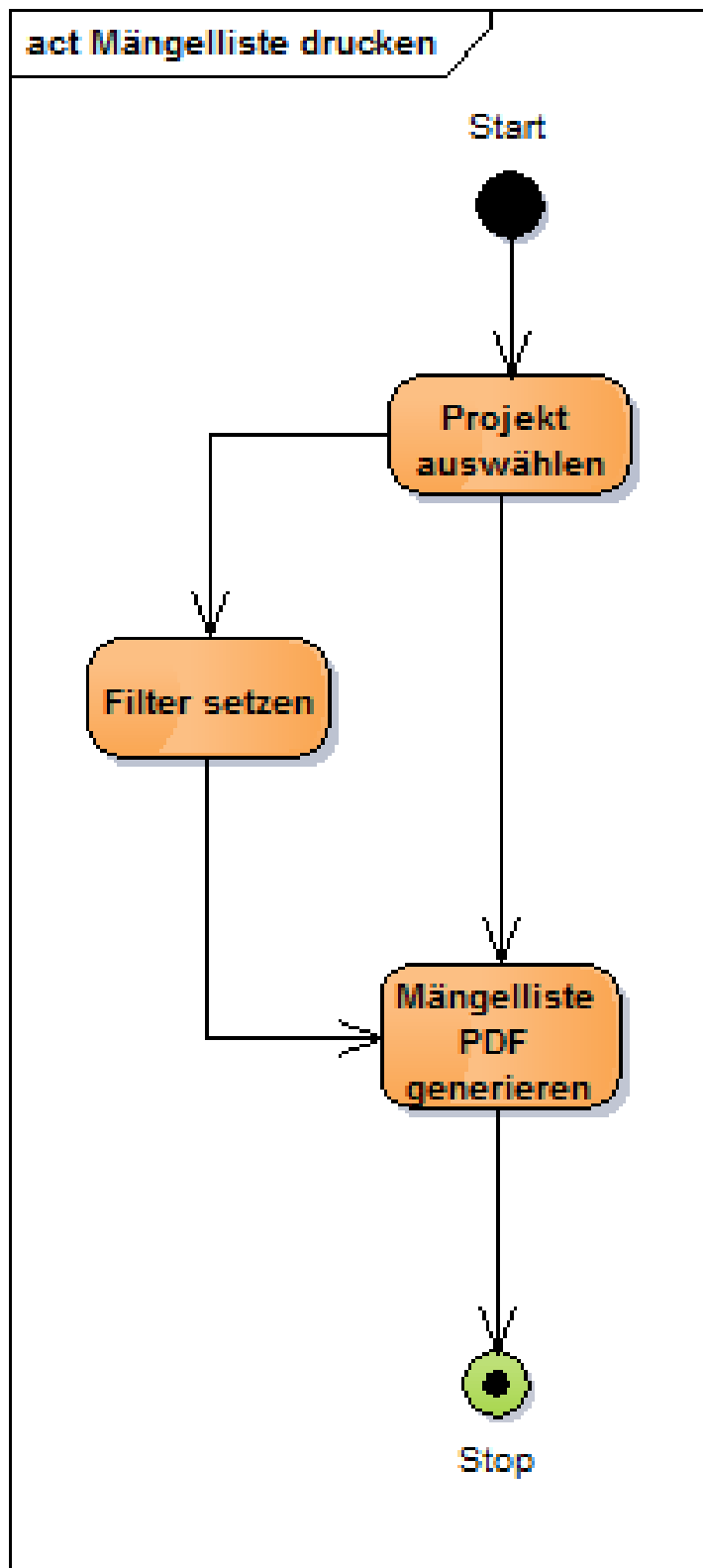


Abbildung 31

15.8 UseCase008

15.8.1 UseCase008 Beschreibung

UseCase008	Mängeldaten filtern
Autor	Mike Iten
Ziel	Die Uls müssen die Mängeldaten nach diversen Kriterien filtern können
Kategorie	Sekundär, der Mangelmanager funktioniert auch ohne entsprechende Funktion.
Vorbedingungen	Es müssen Daten vorhanden sein.
Nachbedingungen Erfolg	Die Daten werden korrekt nach dem gewählten Kriterium gefiltert.
Nachbedingungen Fehlschlag	Die Daten werden falsch gefiltert. z.B. werden bereits erledigte Mängel angezeigt bei einer Suche mit nicht erledigten Mängel.
Akteure	GU-Bauleiter, SU-Admin, SU-Ansprechperson
Auslösendes Ereignis	Die Mängeldaten müssen nach irgendeinem Kriterium gefiltert angezeigt werden.
Beschreibung	Sinnvolle Kriterien müssen eingebaut werden um z.B. alle Mängeldaten für ein Projekt nach Fälligkeitsdatum sortiert anzuzeigen oder z.B. alle Mängel für ein Projekt anzuzeigen, die noch nicht erledigt wurden.



15.8.2 UseCase008 Visualisierung

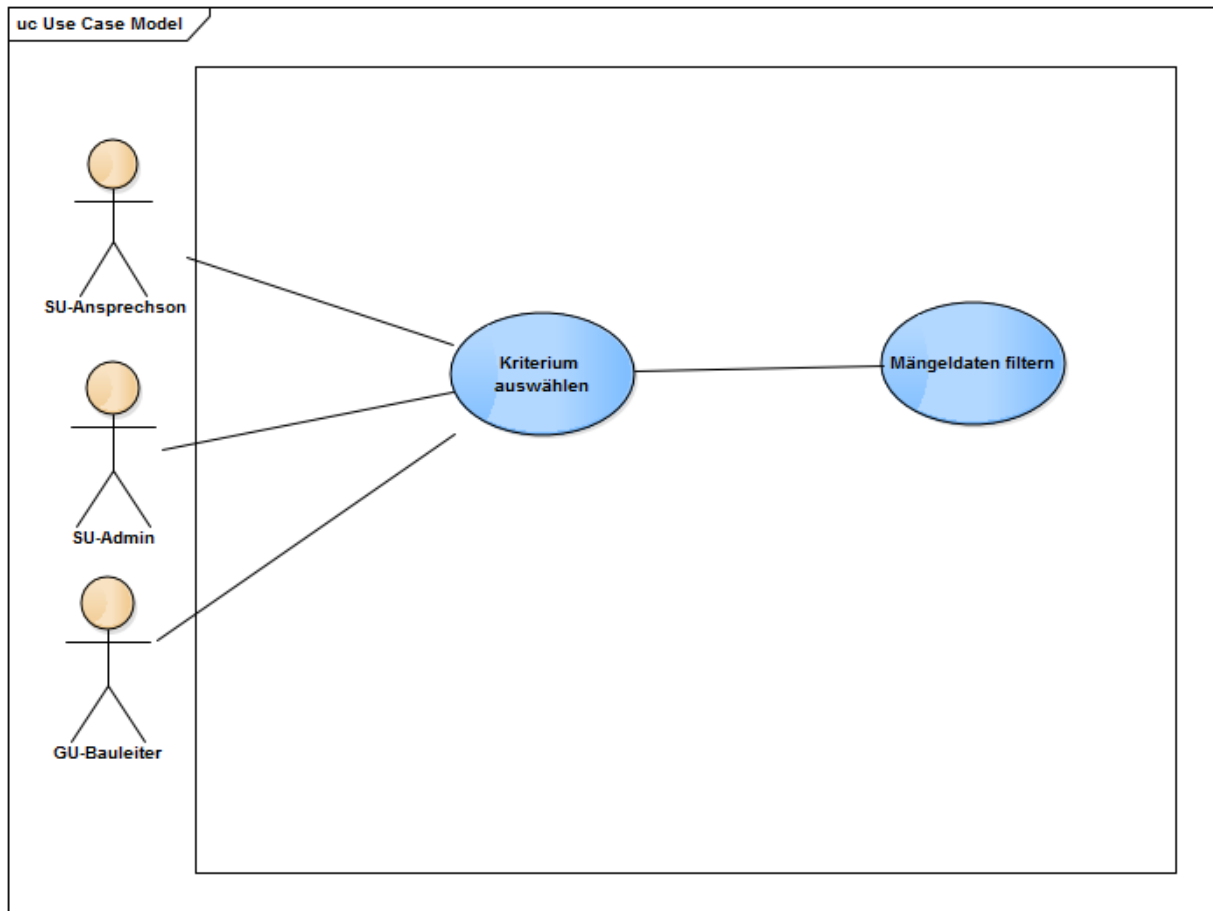


Abbildung 32

15.8.3 UseCase008 Aktivitätsdiagramm

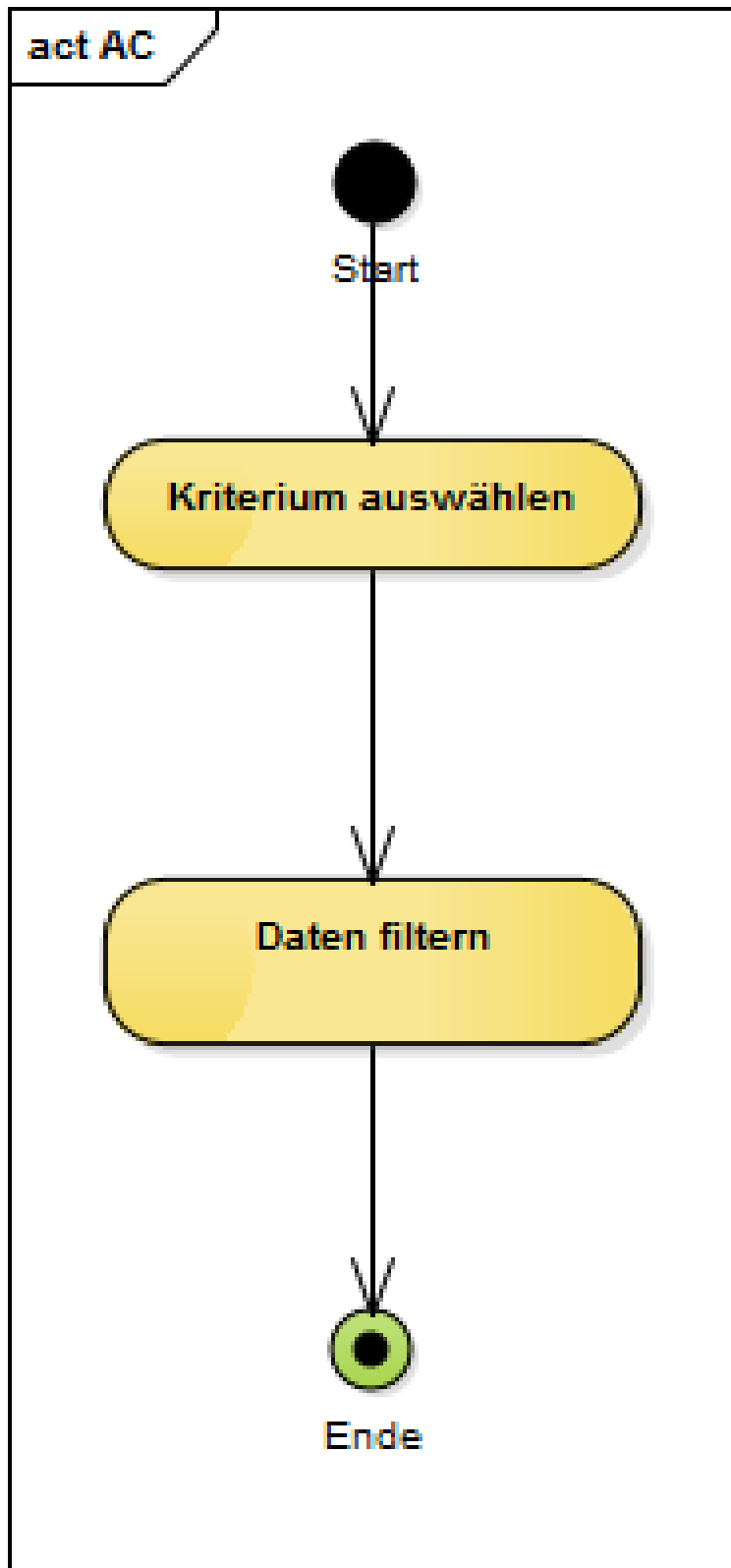
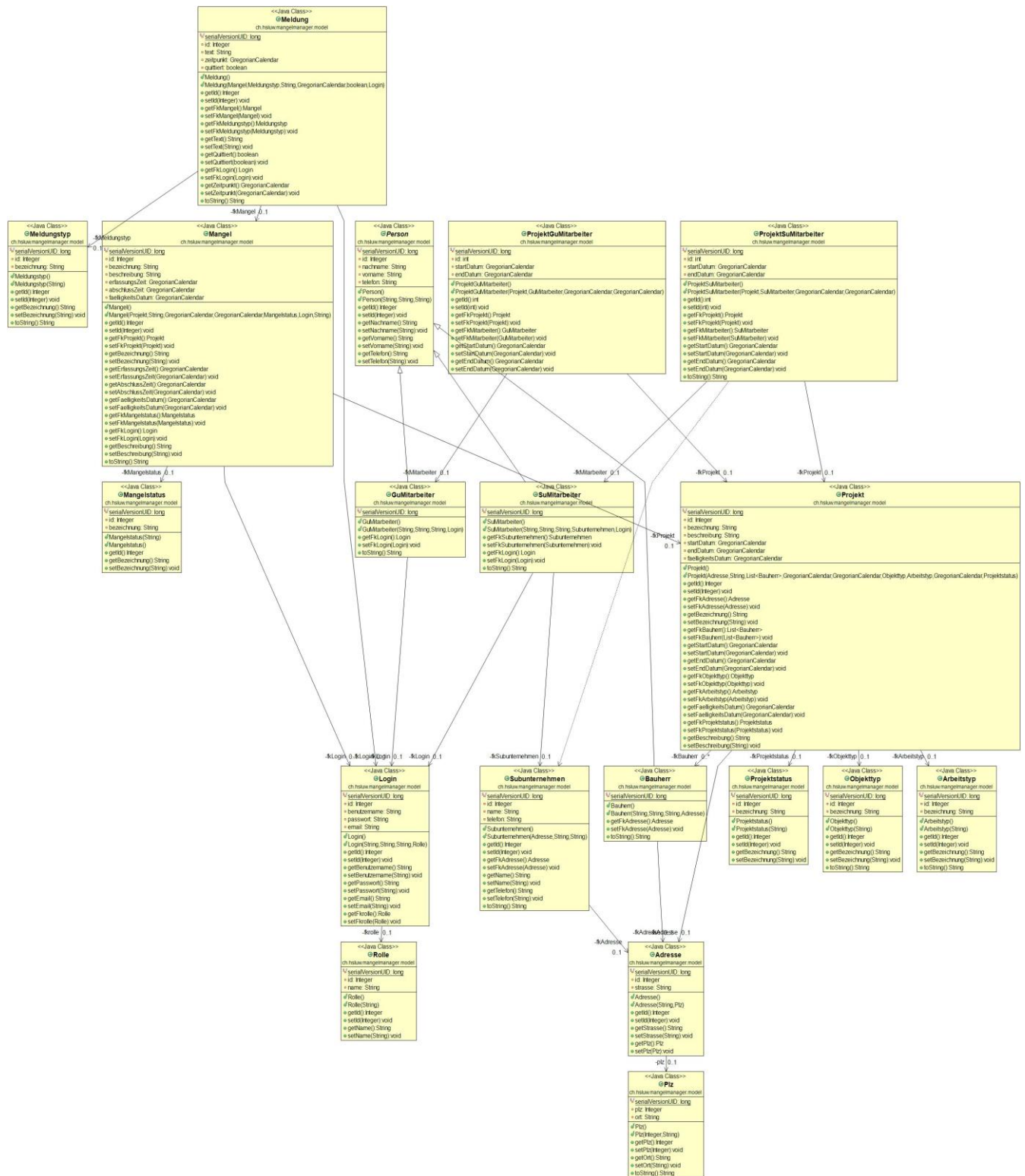


Abbildung 33

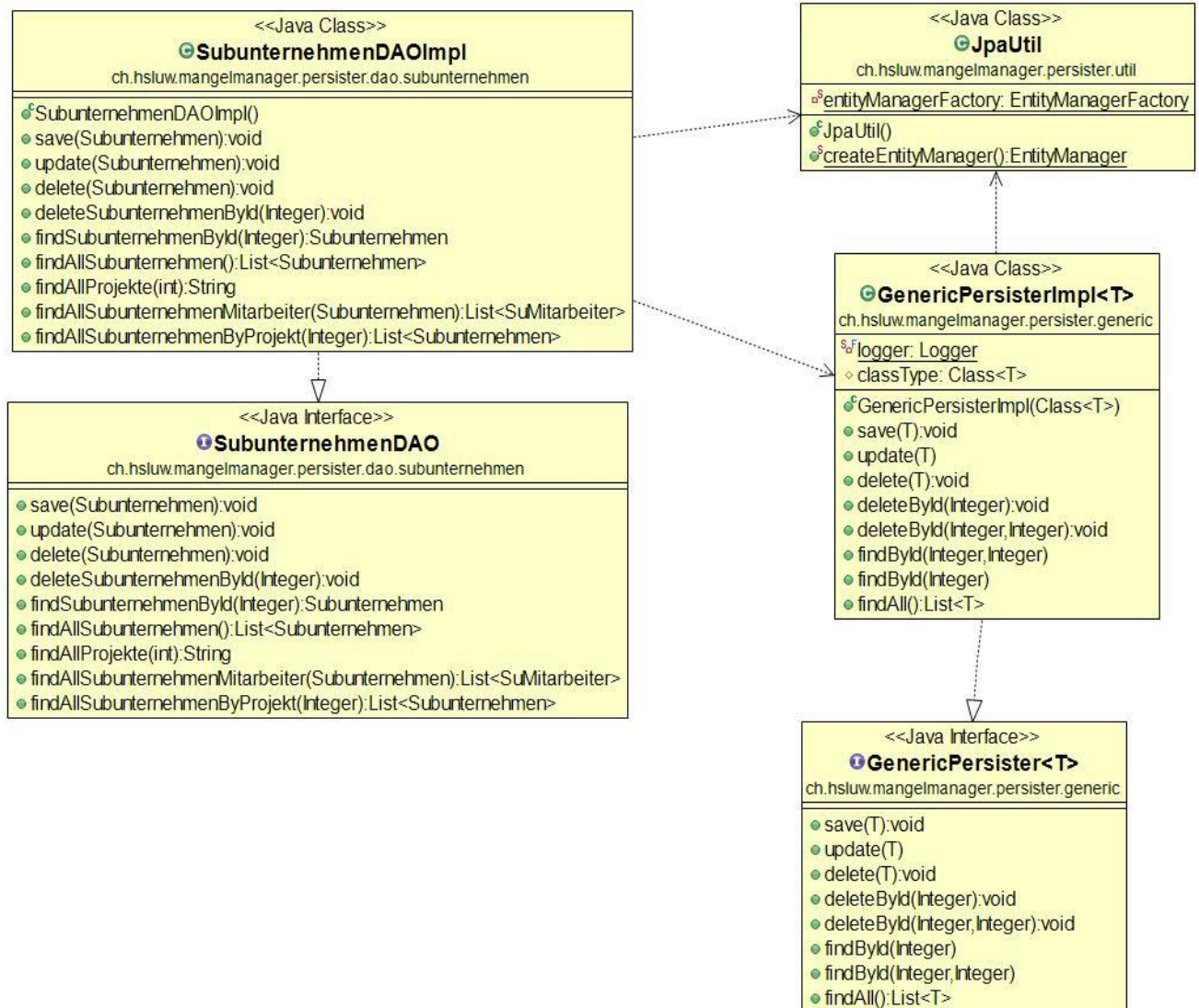
16. Klassendiagramme

16.1 Klassendiagramm Models



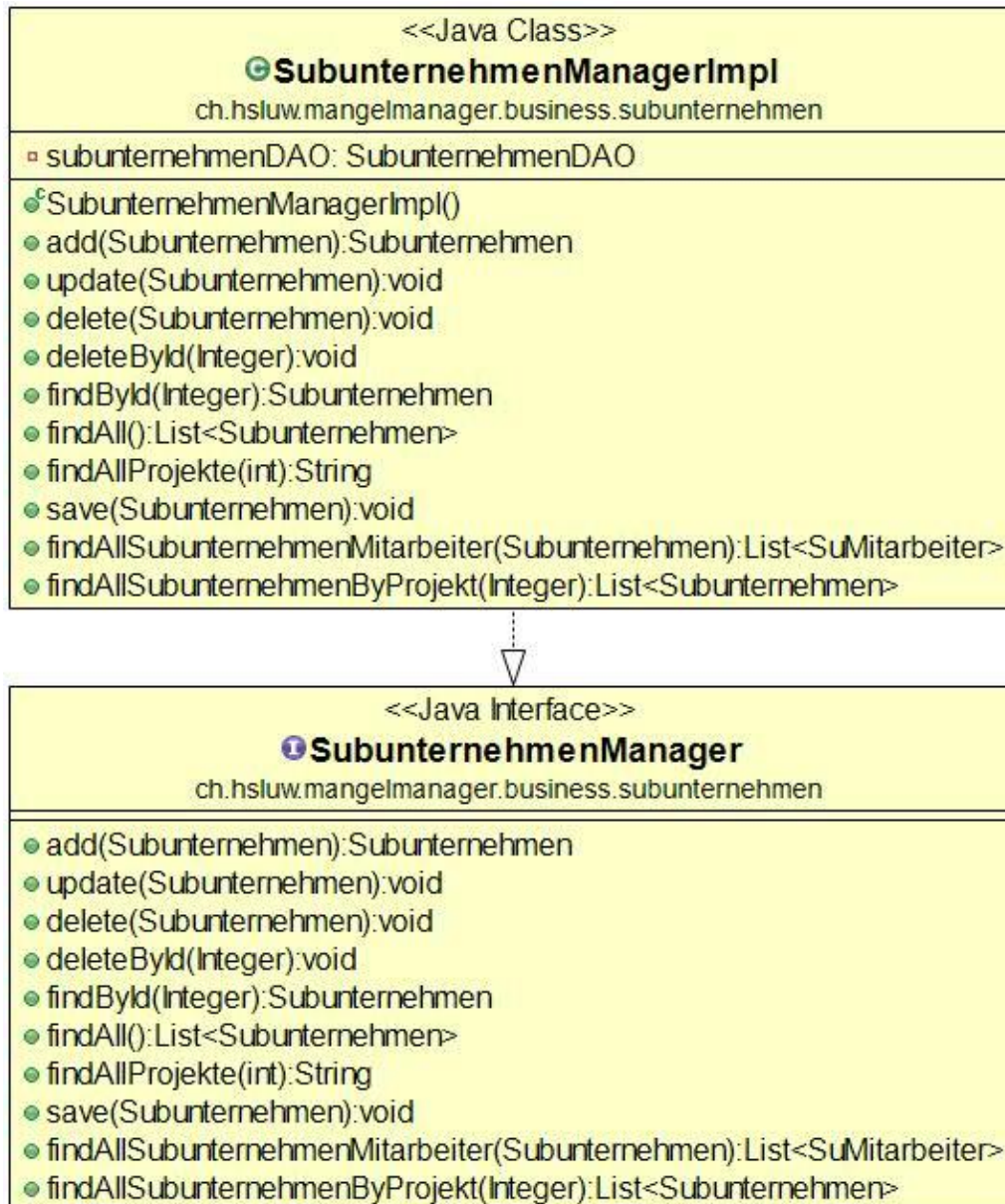
16.2 Klassendiagramm Persister

Das Klassendiagramm der Persister-Schicht wird anhand des Models Subunternehmen aufgezeigt. Folgende Strukturen gelten auch für alle anderen Models.



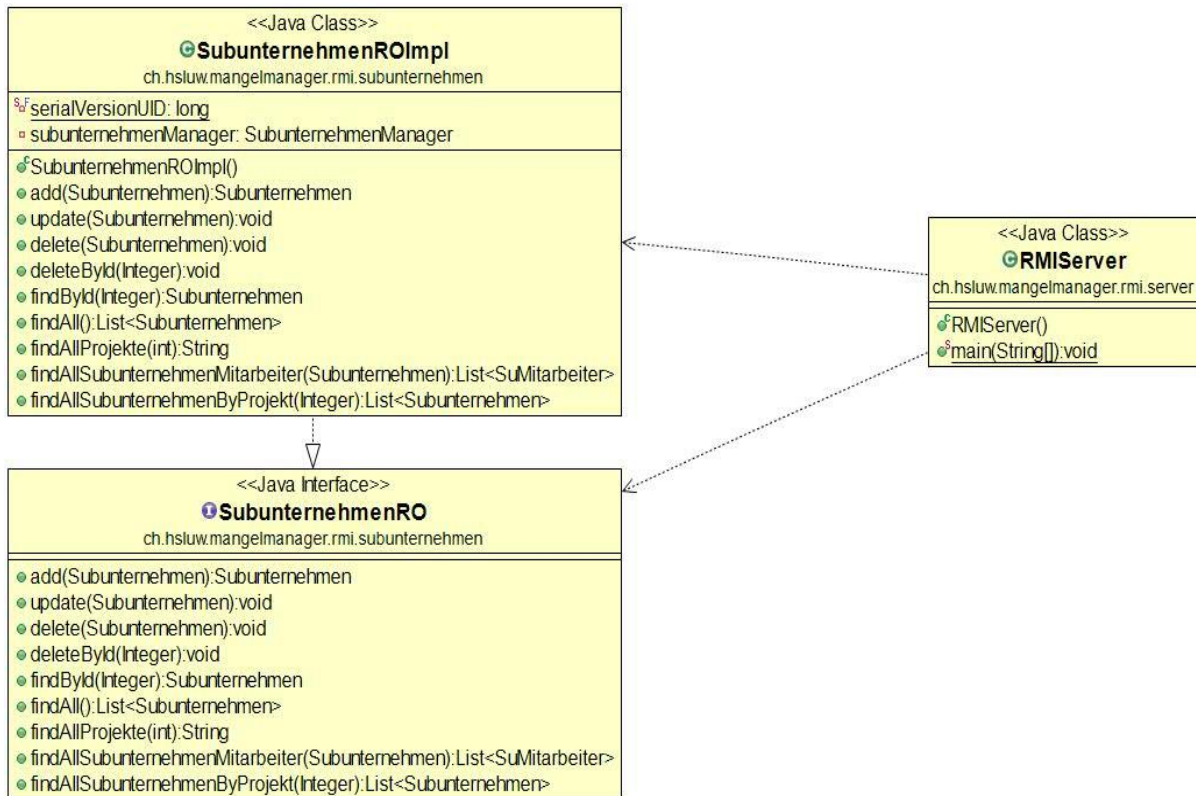
16.3 Klassendiagramm Business

Das Klassendiagramm der Business-Schicht wird anhand des Models Subunternehmen aufgezeigt. Folgende Strukturen gelten auch für alle anderen Models.

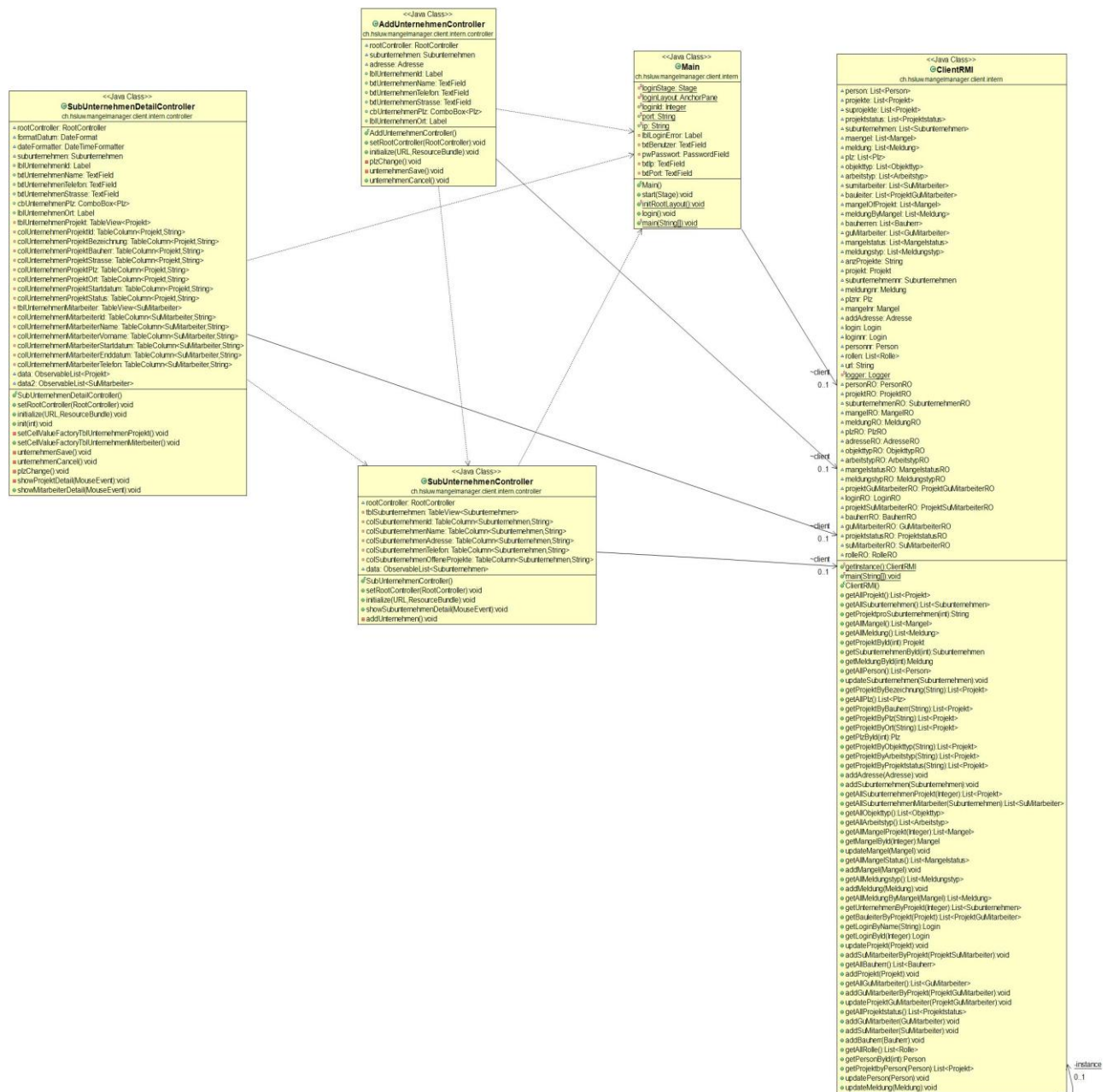


16.4 Klassendiagramm RMI

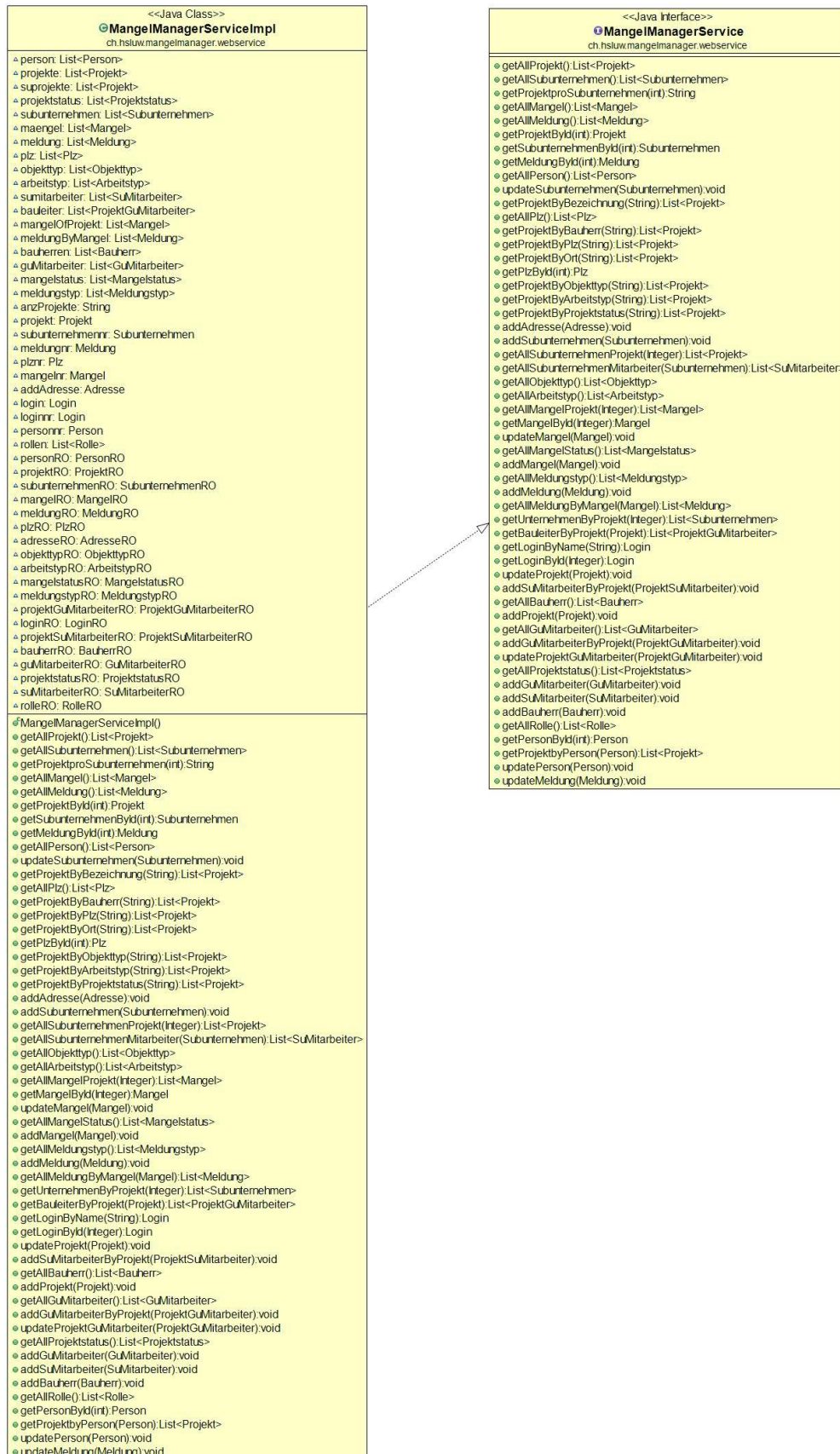
Das Klassendiagramm der RMI-Schicht wird anhand des Models Subunternehmen aufgezeigt. Folgende Strukturen gelten auch für alle anderen Models.



Das Klassendiagramm des RMI Clients wird anhand des Models Subunternehmen aufgezeigt. Folgende Strukturen gelten auch für alle anderen Models.

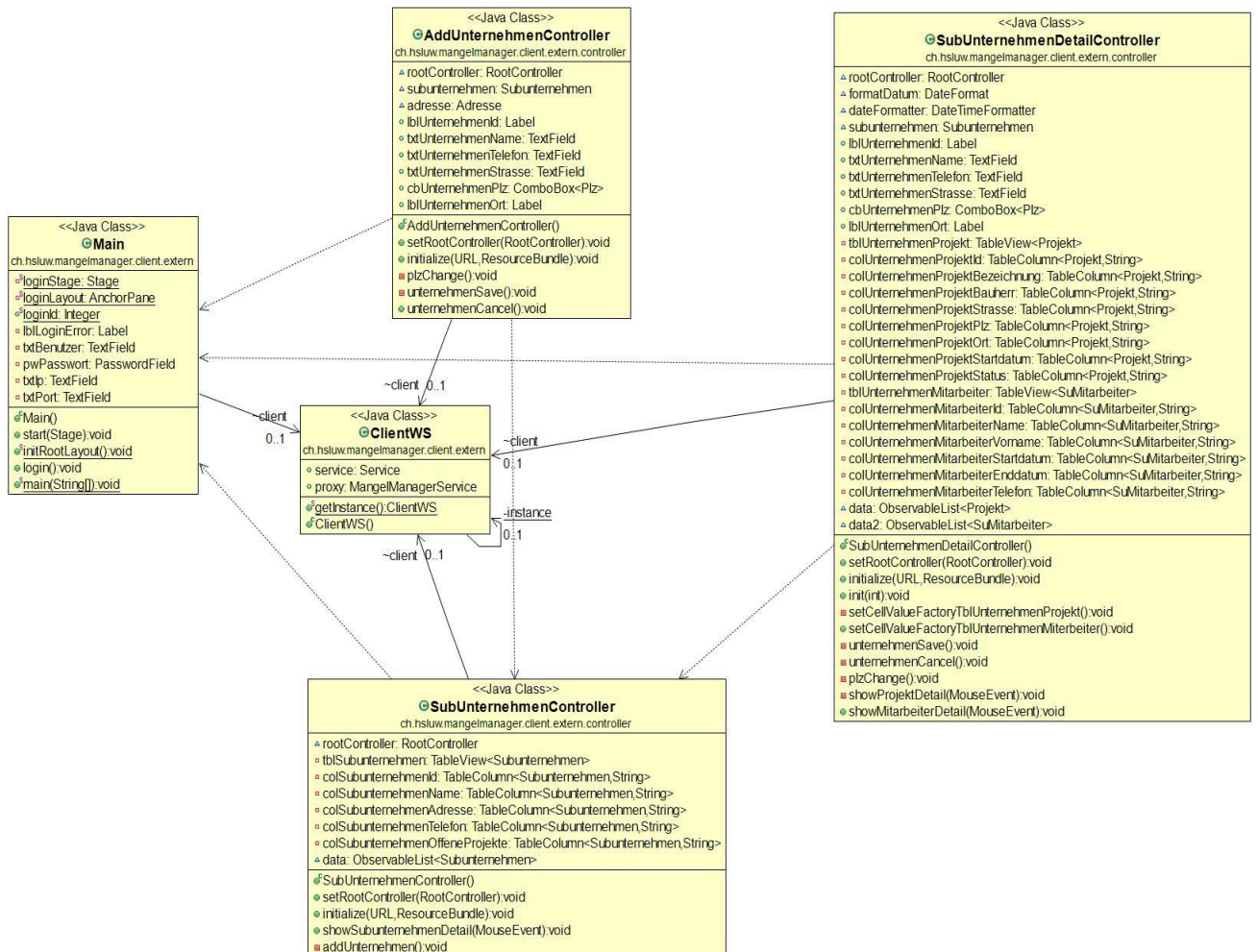


16.6 Klassendiagramm WebService




16.7 Klassendiagramm Externer Client (WebClient)

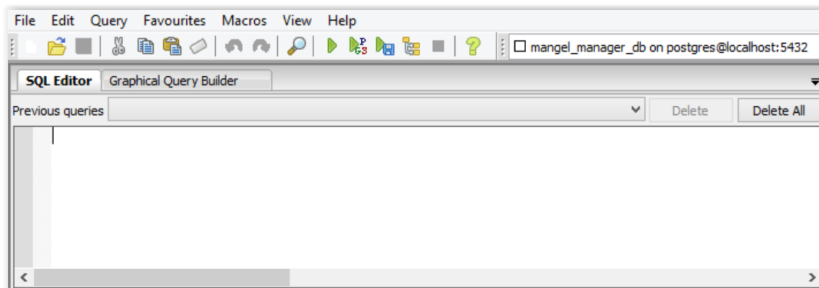
Das Klassendiagramm des Externen WebClients wird anhand des Models Subunternehmen aufgezeigt. Folgende Strukturen gelten auch für alle anderen Models.



17. Deployment-Infos

17.1 Postgres und Datenbank:

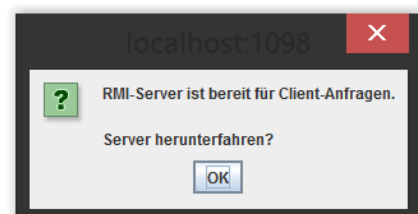
1. Installieren sie Postgres auf ihrem Server
2. Legen sie einen neuen Benutzer mangeldb mit dem Passwort mangelpw an.
 - a. Rechtsklick auf LoginRoles
 - b. New Login Role
 - c. Reiter Properties: Benutzername eingeben
 - d. Reiter Definition: Passwort eingeben
 - e. OK Klicken
3. Postgres-Server mit Einstellungen (localhost:5432) öffnen
4. Nun eine neue Datenbank mangel_manager_db anlegen
 - a. Rechtsklick auf Databases
 - b. New Database
 - c. Datenbanknamen eingeben
 - d. Owner mangeldb setzen
5. Auf Datenbank klicken
6. SQL Query Excecuter öffnen 
7. Dann Zuerst das File „seedfile-schema.sql“ per Drag-and-Drop in das Query-Feld ziehen



8. Auf den grünen Pfeil ganz Links klicken
9. Das Selbe mit dem File „seedfile-data.sql“ ausführen
10. Die Datenbank ist nun bereit.

17.2 Starten RMI-Server:

1. In den Ordner RMI wechseln
2. rmi.properties anpassen wenn gewünscht. (Port und/oder IP)
3. „RMI-Server.jar“ starten
4. Wenn der Start erfolgreich ist sollte es so aussehen:



17.3 RMI-Client starten:

1. In den Ordner Clients wechseln
2. „Client-Intern.jar“ Starten
3. IP und Port auswählen (Identisch mit Daten aus rmi.properties)
4. Login Daten eingeben (z.B. Benutzer: Bauleiter, Passwort: Bauleiter)
5. Login Klicken
6. Der Client ist nun bereit

17.4 WebService starten:

1. In den Ordner WebService wechseln
2. „MangelManager.war“ in das webapps Verzeichnis des Tomcat Servers kopieren
3. „rmi.jar“ in das lib Verzeichnis von Tomcat Kopieren
4. Den Server Starten
5. Die .war Datei kann nun wieder aus dem webapps entfernt werden. Im generierten Ordner MangelManager unter WEB-INF\classes\ findet sich ein ws.properties File. Hier bitte identische IP und Port wie in rmi.properties angeben.
6. Server neustarten, damit Änderungen in aus der Dateien ws.properties übernommen werden.

17.5 Web Client Starten:

1. In den Ordner Clients wechseln
2. „Client-Extern.jar“ starten
3. Der Client hat nun Zugriff via WebService auf die Datenbank

18. TDD und JUnit

18.1 TDD

Wir haben die Tests erst geschrieben, als das Programm fast fertig programmiert war. Wir haben daher nicht zuerst die Tests geschrieben und anschliessend die dafür nötigen Funktionen programmiert sondern zuerst programmiert.

Aus zeitlichen Gründen beschränken wir uns auf das Testen der wichtigsten Klassen.

18.2 Test-Files

Für unseren Mängelmanager gibt es 3 Test-Files:

CreateEntityTest im Persister. Dieser Test dient als Seed-File. Damit wird die Datenbank mit Beispieldaten gefüllt, anhand welcher wir die übrigen Tests ausführen können. Die Daten werden in Lists gespeichert und anschliessend persistiert. Diese Datei wird nicht mithilfe von Assert überprüft. Die Korrektheit der Daten wird manuell in der Datenbank überprüft.



EntityTest im Persister. Dieser Test testet die Methoden von ProjektDAO und ProjektDAOImpl, welche ebenfalls im Persister auffindbar sind. Dabei wird jede Methode des ProjektDAO mittels einer erzeugten Instanz anhand der Beispieldaten des Seed-Files getestet. Hier wird getestet, ob Daten richtig ausgelesen und persistiert werden können.

ClientRMITest im Client. Dieser Test testet die Methoden von ClientRMI, welches ebenfalls im Client auffindbar ist. Dabei wird jede Methode des ClientRMI mithilfe einer erzeugten Instanz des ClientRMI anhand der Beispieldaten des Seed-Files getestet. Hier wird getestet, ob die Daten, welche wir im GUI wiedergeben, korrekt ausgelesen und verändert werden können.

18.3 JUnit

Die Tests werden alle mithilfe von JUnit4 durchgeführt und mittels Assert überprüft. Die Tests liefern alle positive Rückmeldungen.

19. Funktionale Test's

Test ID	1
Beschreibung	Logininformationen wurden nicht korrekt angegeben
Resultat	Fehlermeldung und erneuter Loginversuch.
Test ID	2
Beschreibung	Ein Mangel wurde erfasst.
Resultat	Der Mangel wird für das jeweilige Projekt angezeigt und der Bauleiter kann diese korrekt einsehen.
Test ID	3
Beschreibung	Ein Mangel wurde abgearbeitet.
Resultat	Der Bauleiter kann den Mangel als erledigt kennzeichnen.
Test ID	4
Beschreibung	Ein neues Projekt wird hinzugefügt
Resultat	Jeder kann das neue Projekt begutachten, der Bauleiter aber mit erhöhter Berechtigung.
Test ID	5
Beschreibung	Dem Projekt wird ein Subunternehmen zugeteilt
Resultat	Das Subunternehmen und zusätzlich die Ansprechperson wird hinzugefügt.

Test ID	6
Beschreibung	Ein Mitarbeiter kann die Mängel nicht bearbeiten
Resultat	Nur der Bauleiter hat die Berechtigung die Mängel zu bearbeiten oder als erledigt zu kennzeichnen.

Test ID	7
Beschreibung	Ein neues Projekt wurde erfasst
Resultat	Das Projektobjekt, der Projekttyp und das Datum ist klar ersichtlich, sowie die Adresse des Projekts und der Bauherr wurden angegeben.

Test ID	8
Beschreibung	Der Bauleiter will alle Mängel anschauen
Resultat	Alle Mängel und Arbeiten des Bauleiters sind zeitlich gegliedert und aufgeführt.

Test ID	9
Beschreibung	Der Bauleiter will das Subunternehmen kontaktieren
Resultat	Die involvierten Subunternehmen sind klar ersichtlich und die Ansprechpersonen sind zeitlich aufgeführt.

Test ID	10
Beschreibung	Der Bauleiter will das Subunternehmen kontaktieren
Resultat	Die involvierten Subunternehmen sind klar ersichtlich und die Ansprechpersonen sind zeitlich aufgeführt.

Test ID	11
Beschreibung	Der Bauleiter des Projektes muss ausfindig gemacht werden
Resultat	Es kann auch möglich sein, dass ein Projekt durch mehrere Bauleiter betreut wird. Jedoch ist der Hauptbauleiter ersichtlich und dessen Adresse aufgeführt.

Test ID	12
Beschreibung	Der Bauleiter möchte das Projekt eines Kollegen einsehen
Resultat	Dies ist nicht möglich, da jeder Bauleiter nur sein eigenes Projekt einsehen kann.

Test ID	13
Beschreibung	Das Subunternehmen begutachtet das Projekt
Resultat	Jegliche Mängel sind aufgeführt und können auch über das Subunternehmen quittiert werden.



Test ID	14
Beschreibung	Die Meldungen werden eingesehen
Resultat	Jeder Mangel kann mehrere Meldungen beinhalten, die durch das Generalunternehmen oder Subunternehmen erfasst werden können, sie sind für den Bauleiter immer sofort zur Verfügung.
Test ID	15
Beschreibung	Der Mangel wird vom Subunternehmen abgearbeitet
Resultat	Der Bauleiter kann den Mangel einsehen und bestätigen, dass dieser abgearbeitet wurde und kann demnach den Mangel als erledigt kennzeichnen. Ist dies nicht der Fall, dann kann er den Mangel erneut an das Subunternehmen schicken.
Test ID	16
Beschreibung	Die Meldung soll verändert werden
Resultat	Niemand kann die Meldung nach dem Speichern verändern.
Test ID	17
Beschreibung	Die Mängel sollen nach einem Kriterium gefiltert werden
Resultat	Der Bauleiter möchte alle Mängel bezüglich der Hauswand aufgeführt haben.
Test ID	18
Beschreibung	Der Bauleiter möchte alle Mängel in Papierform haben
Resultat	Der Bauleiter kann alle Mängel in Listenform anzeigen lassen und diese dann ausdrucken.

20. DB – Dokumentation

Siehe Datei „Diagramme/ERD/definitivesERD.pdf“ (zu gross für eine A4 Seite)



21. Beiträge pro Projektmitglied

21.1 Luca Kündig

INM 21 - Individuelles Portfolio			
Name:	Luca Kündig		
Datum eventuell Phase	Dauer Aufwand	Tätigkeit / Aufgabe	Bemerkung/Erkenntnis
04.03.2015	1.5 Std.	Ich habe zusammen mit Mike Monticoli und Sandro Rytz die Requirements für unser Projekt definiert. Diese sind im Dokument 2015_FS_INM21_Requirements_Mangel-Manager.xlsx festgelegt.	Durch die Requirements hat unser ganzes Projektteam einen guten Gesamtüberblick über unsere zukünftigen Tätigkeiten und die Anforderungen die wir Realisieren müssen.
15.03.2015	1.5 Std.	Ich habe die grob definierten UseCases verfeinert und in 8 Haupt-UseCases aufgeteilt. Die UseCases habe ich dann gleichmässig auf die Gruppenmitglieder verteilt, dass alle ihren Anteil erledigen können.	Durch das anwenden von CRUD konnte ich unsere UseCases drastisch verringern. Da ich mich zentral mit den UseCases befasse habe ich einen guten Überblick über den Fortschritt der Arbeit
17.03.2015	1.5 Std.	Ich habe meinen beiden UseCase Beschreibe ausgefüllt und in unser Repository gestellt.	UseCases geben einen guten Überblick über die Funktionalitäten die die Applikation enthalten muss
18.03.2015	1.0 Std.	Ich habe meine UseCase Berichte noch einmal überarbeitet.	
18.03.2015	1.0 Std.	Zusammen mit Sandro Ritz habe ich unser erstes ERD noch einmal verfeinert und wir sind mittlerweile zu einer Definitiven version gekommen	Ein gutes ERD vereinfacht die Spätere planung der Applikation. So können wir unnötige redesigns verhindern
25.03.2015	2.0 Std.	Die UseCase Diagramme und die dazugehörigen Aktivitätsdiagramme habe ich im EnterpriseArchitect erfasst.	Diagramme vereinfachen das Verständniss der einzelnen Abläufe.
01.04.2015	2.0 Std.	Sämtliche UseCases wurden durch mich reviewed. Einige habe ich persönlich überarbeitet. Andere müssen noch einmal durch den entsprechenden Autor verbessert werden.	
06.04.2015	2.0 Std.	Ein Zeitplan mit klar definierten Meilensteinen und Zeiten ist durch mich erstellt worden. Spezifische Aufgaben sind klar den jeweiligen Personen zugeordnet. Weiter habe ich die zukünftigen Arbeiten definiert und Aufträge erteilt.	



12.04.2015	3.0 Std.	Erstellen des ersten Klassendiagramm entwurfs. Darin enthalten sind sämtliche Models und ihre beziehungen	
19.04.2015	2.5 Std.	Programmieren von 5 Models für das Projekt. Das heisst. Sämtliche Attribute, getter und setter, alle annotationen pro Klasse.	
22.04.2015	3.5 Std.	Ersellen aller nötigen Klassen vom Persister bis zur Rmi Schicht der 5 klassen für die ich bereits das Model erstel-let habe.	
30.04.2015	3.5 Std.	Erstellen der Äusseren Tabellenansicht der entitäten Sub-unternehmen und Meldungen	
04.05.2015	2.0 Std.	Innere Ansicht + Erster ansatz für update Methode für Subunternehmen geschrieben.	
05.05.2015	2.0 Std.	Innere Ansicht für Meldungen geschrieben	
08.05.2015	4.0 Std.	Innere Tabelle Projekt pro subunternehmen	
08.05.2015	3.0 Std.	Innere Tabelle SuMitarbeiter pro subunternehmen	
11.05.2015	2.0 Std.	Add Mangel geschrieben	
11.05.2015	1.0 Std.	add Meldung geschrieben	
13.05.2015	3.0 Std.	added Login for GUI	
14.05.2015	6.0 Std.	created webservice + webservice client	
14.05.2015	3.0 Std.	tried to fix bidirectional marshalling problem	
15.05.2015	2.0 Std.	fixed Projekt add	
15.05.2015	4.0 Std.	added Person add	
16.05.2015	3.0 Std.	deleted all bidirectional relationships	
16.05.2015	4.5 Std.	replaced all getters and setters which used bidirectional relationships	
16.05.2015	1.0 Std.	fixed seedfile	



17.05.2015	4.0 Std.	implemented Person add controller	
17.05.2015	4.0 Std.	updated webservice and webclient	
17.05.2015	1.0 Std.	Made Classdiagramms for model and persister	
18.05.2015	1.0 Std.	Dokumentation schreiben. Kapitel 8 vervollständigen	
19.05.2015	3.0 Std.	finished all classdiagramms + classdiagramm doku + laufzeitschicht	



21.2 Sandro Ritz

INM 21 - Individuelles Portfolio			
Name:	Ritz Sandro		
Datum eventuell Phase	Dauer Aufwand	Tätigkeit / Aufgabe	Bemerkung/Erkenntnis
01.03.2015	2.0 Std.	Durcharbeiten Aufgabenstellung, Gedanken erstellt zum Aufbau der Applikation	
04.03.2015	1.0 Std.	Requirements Engineering	
11.03.2015	2.0 Std.	Usecase definieren	Benötigte guten Durchblick von der Aufgabenstellung
17.03.2015	3.0 Std.	Provisorisches ERD von Luca vervollständigt	
18.03.2015	2.0 Std.	ERD angenommen, Usecases beschrieben	
19.03.2015	2.0 Std.	Generierung von SQL aus Schema mit Workbench	Problem -> Constraints Name muss eindeutig sein. Dies ist ein wenig versteckt im Programm MySQL Workbench.
24.03.2015	2.5 Std.	UseCase 1 + 7 mit Aktivitätsdiagrammen gezeichnet	Bei der Erstellung der Aktivitätsdiagramme stiess ich mit Enterprise Architekt auf einige Probleme. Da alle Diagramme recht ähnlich sind kopierte ich meine Diagramme immer ins neue Diagramm und passte diese dann an. Das Problem war, dass die Referenz zum vorherigen Diagramm genommen wurde und ich am Schluss bemerkte, dass es mir alle Diagramme mit dem letzten überschrieben hat.
25.03.2015	1.5 Std.	Kontrolle aller UseCases der Projektmitglieder und Aktivitätsdiagrammen mit Luca	Falsche oder unvollständige korrigiert.
27.03.2015	2.0 Std.	Entwicklungsumgebung + Projekt eingerichtet für GitHub	Den Dienst GitHub, TomCat kennengelernt. Es ist sehr mühsam TomCat mit dem Webservice zum Laufen zu bringen. Entweder nimmt man die normale Tomcat Version und muss mühsam die einzelnen Libs einbinden, oder man nimmt TomCatEE und man merkt das gewisse Libs von WS schon vorhanden sind und manche fehlen. In Zukunft werde ich sicher per REST einen Webservice integrieren, da diese Technologie eindeutig besser unterstützt wird.
30.03.2015	3.0 Std.	Programmierkonzept angefangen zu erstellen	
31.03.2015	2.0 Std.	Programmierkonzept fortgefahren	



01.04.2015	0.5 Std.	Diagramme aus Enterprise Architekt exportiert und als BMP Bild abgespeichert	
08.04.2015	2.0 Std.	GUI Reviewed, ERD aktualisiert	
15.04.2015	6.5 Std.	Einbinden des Referenzprojekts; Datenbank lokal eingerichtet ; Eclipse Projekte für Projektmitglieder erstellt ; GitHub konfiguriert, Abhängigkeiten der Projekte Model und Persister hinzugefügt ; persistence.xml geschrieben; diverse Klassen implementiert ; Model: Person, Bauherr, Adresse, Plz, Projekt, Projektstatus, Arbeitstyp, Objekttyp implementiert und Vorlage für Projektmitglieder erstellt ; Klasse zum Testen, ob die Entities/Models richtig erstellt und verknüpft worden sind, erstellt.	
16.04.2015	3.5 Std.	Tasks planen und an Projektmitglieder vergeben, Business Logic von Model Projekt implementiert	
17.04.2015	2.0 Std.	RMI Client und Server Logik implementiert	
19.04.2015	3.0 Std.	Client Schnittstelle bis zur Persister Schicht von Projekt implementiert , NamedQueries in Projekt Model implementiert	
20.04.2015	2.0 Std.	NamedQueries für Projekt geschrieben + letzte Änderungen committed	
21.04.2015	0.5 Std.	NamedQuery findByDatumFromTillEnd für Projekt geschrieben	
21.04.2015	0.5 Std.	Geeignete Konzepte für Authorisierung gesucht	Keine gute Lösung für unsere Architektur gefunden, da wir nicht alles über die Webservice Schnittstelle zugreifen.
22.04.2015	1.0 Std.	Korrektur aller Models	Viele Fehler in den Models mancher Mitglieder. Habe diverse Korrekturen vorgenommen.
22.04.2015	0.3 Std.	Zuteilung Aufträge bis 23.04.2015	
22.04.2015	1.5 Std.	Arbeitstyp und Objekttyp in allen Schnittstellen Methoden implementiert	
28.04.2015	3.5 Std.	Client Projektstruktur angepasst; Alle benötigten Controller Klassen hinzugefügt ; Bidirektionale Beziehung im Model Projekt und Mangel hinzugefügt; RootController implementiert, um die Navigation zwischen den verschiedenen Views zu bewerkstelligen ; Im Projekt-Controller die Projekt-Übersichtstabelle programmiert ; Problem beim Darstellen von verknüpfen Tabellen in TableViews gelöst.	
04.05.2015	4.5 Std.	Controller Architektur mit Navigation zwischen verschiedener Views angepasst ; Personcontroller anfangen zu implementieren; Auftrag an Monti bezüglich Views vorbereitet und zugewiesen	Ich habe herausgefunden, dass wir bei jedem Controller, welcher für eine View zuständig ist, eine Referenz auf den RootController benötigen um die Views im "Center" Bereich austauschen zu können (Hatten zuerst mit "new" ein neues Objekt vom Controller als Referent erstellt. Dies führe natürlich wie ich später gelernt habe zu Problemen -> NullPointerException). Konzept für die Weitergabe der jeweiligen

			Referenzen von der Äusserenansicht zur Detailansicht entwickelt und implementiert.
05.05.2015	4.0 Std.	Person Business, RMI, Client, Persister Schnittstelle implementiert + Bind an RMI ; Person AussereView erstellt und PersonController zum Anzeigen der Overview Tabelle implementiert ; PersonDetailController angefangen zu implementieren	Probleme beim Zugriff auf Fremdschlüssel Attribute anderer Models in der TableView. Mittels eigener ValueFactory konnte dieses Problem jedoch zügig gelöst werden.
06.05.2015	1.0 Std.	Auftrag an Monticoli zugewiesen und detailliert erklärt	
07.05.2015	3.0 Std.	Gestartet mit der Filterung von Projekten, Team Hilfestellung gegeben ; Konzept Innere Projekt View auf Blatt skizziert für Monticoli	Hatten anfangs die Attribute in den ComboBoxen als String geladen. Dies war aber zur Weiterverarbeitung recht mühsam wie sich später herausstellte. Daher wechselten wir zu einer besseren Methode -> Das Laden von Objekten in die ComboBox und Mittels toString() die Ausgabe der jeweiligen Models definieren
08.05.2015	3.0 Std.	Projektfilter fertiggestellt, Projekt Innere View Fields ausgefüllt + tbloffeneMängel in Projekt implementiert	
12.05.2015	4.0 Std.	In Projekt Inner View habe ich die Meldungen zu jeweiligem Mangel in der Tabelle hinzugefügt. Zudem die Subunternehmertabelle und die Bauleitertabelle implementiert	
13.05.2015	4.5 Std.	In Projekt Inner View: Ein Subunternehmen mit Ansprechperson zu Projekt hinzufügen implementiert; "Neues Projekt erstellen" implementiert	
14.05.2015	9.0 Std.	ProjektInner Weiterleitungen implementiert ; Add Meldung Controller implementiert ; Views angepasst, Bauleiter zu Projekt hinzufügen implementiert, Testdaten laden, Fehler in diversen Controllern behoben, Plz-Ort Methoden in allen Controllern korrigiert.	Problem-> fkAdresse scheitert irgendwo. Von dem sind relative viele Entities abhängig. Dadurch gibt es nun Probleme bei allen "Hinzufügen" Operationen. Das Problem liegt wohl irgendwo an den Cascade Konfigurationen eines Models.
18.05.2015	3.0 Std.	Diverse kleine Fehler behoben zB Datumseingaben, Model toString Funktionen geschrieben, "Hinzufügen" Button in Unternehmen View entfernt, Kontrolle Klassendiagramme, Problembeschrieb im Portfolio verbessert	



	3.0 Std.	Software zum Generieren von PostgreSQL Schema Diagrammen heruntergeladen und installiert. Definitives ERD generieren lassen. Letzte Schritte im Projekt -> Hilfestellung für Teammitglieder	
19.05.2015	3.0 Std.	Management Summary geschrieben + Blackbox & Whitebox Diagramm erstellt	



21.3 Mike Iten

INM 21 - Individuelles Portfolio			
Name:	Mike Iten		
Datum eventuell Phase	Dauer Aufand	Tätigkeit / Aufgabe	Bemerkung/Erkenntnis
25.03.2015	3 Std.	Use Case Modellierung für die Filterung, welche der Ablauf der Filterung aufzeigen soll (Kriterium auswählen, Daten auflisten)	Durch die UseCases haben wir eine gute Übersicht über die einzelnen Abläufe bekommen und haben uns die Logiken etwas vereinfacht. Hier habe ich das erste mal ein Use Case und ein Aktivitätsdiagramm erstellt. War aber durchaus interessant, was für eine Übersicht es bieten kann, wenn man für alle Abläufe Use Cases erstellt.
05-17.04.2015	6 Std.	GUI Moqups erstellt , als vordefiniertes Layout wie das GUI später aussehen soll	Durch die Moqups haben wir ein ungefähres Bild von den GUI's bekommen und konnten dadurch die GUI-Implementierung um einiges vereinfachen. Zum ersten Mal hab ich mit diesem Programm gearbeitet, war aber ein gutes und übersichtliches Programm. Mit Mike hab ich noch Besprechungen gemacht, wie wir die GUI's darstellen sollen, mit einigen Ratschlägen bei Luca, da er schon des öfteren Projekte machte.
23-28.04.2015	10 Std.	Login und Rolle programmiert für Model, Persister, Business und RMI.	Habe die Models Login und Rolle programmiert für jeden einzelnen Layer. Mit etwas Hilfe von Sandro konnte ich die Models zufriedenstellend anpacken und erstellen



Durch die gesamte Projektzeit immer wieder.	30 Std.	Mehrere Dokumentationspunkte abgearbeitet	<p>Mit Arc42 mehrere Dokumentationspunkte geschrieben und gestaltet.</p> <p>Die Dokumentation hab ich hauptsächlich mit Cihan zusammen gemacht. Wir haben am Anfang Teile zusammen erarbeitet und uns danach die Dokupunkte aufgeteilt und alle einzeln erarbeitet.</p> <p>Ich hatte zu Beginn Schwierigkeiten die ganzen Themen zu verstehen, was unter den einzelnen Punkten erwartet wird. Konnte aber gute Beispiele aus dem Internet finden und diese dann interpretieren. Auf die Dokuarbeit bin ich sehr stolz.</p>
17-18.05.2015	15 Std.	Funktionale Tests erarbeitet und getestet anhand des Managers	<p>Ich wusste dass die Testcases ein sehr wichtiger Bestandteil eines Projektes ist und hab mir auch entsprechend Gedanken darüber gemacht und mich lange damit auseinandergesetzt.</p> <p>Diesen Schritt werde ich in Zukunft mit Sicherheit sehr oft brauchen und hab mir so viele Informationen angeeignet wie nur möglich.</p>



21.4 Mike Monticoli

INM 21 - Individuelles Portfolio			
Name:	Mike Monticoli		
Datum eventuell Phase	Dauer Aufand	Tätigkeit / Aufgabe	Bemerkung/Erkenntnis
04.03.2015	1.5 Std.	Ich, Sandro und Luca haben die Requirements festgelegt.	Dies war das erste Mal, dass ich mit Requirements gearbeitet habe und es mir einen guten Einblick gegeben was die Ziele des Projekts sind.
11.03.2015	1.5 Std.	Wir haben die Requirements in Use-Cases geschrieben und diese verteilt.	Wir haben pro Usecase mehrere Requirements abgedeckt. Ursprünglich dachten wir, dass wir pro Requirement mehrere Usecases haben.
17.03.2015	0.5 Std.	Ich erledigte meinen Usecase005 und hab ihn auf Github hochgeladen	Mir war nicht ganzklar was in einen Usecase gehört weshalb ich mir vor allem an den Usecases von Luca hielt.
25.03.2015	1.0 Std.	Ich habe den UseCase noch einmal überarbeitet um in unseren Standards gerecht zu erlediegen.	Nachdem mir gesagt wurde, wie genau wir den Usecase erledigen müssen, konnte ich ihn den Anforderungen gerecht ausfüllen
26.03.2015	1.3 Std.	Ich habe zu meinem Usecase das passend Diagramm mittels Enterprise Architect erstellt.	Enterprise Architect war ein nicht ganz einfach zu verstehendes Programm. Die Arbeit in der Gruppe ermöglicht mir allerdings schnelles verstehen.
01.04.2015	1.5 Std.	Ich habe die Activity-Diagramme für meinen Usecase005 erstellt	Ich konnte mich beim Activity-Diagramm an die Vorlage von Sandro Ritz halten und meine Aktivitäten so nach seinen modellieren
05.04.2015	2.0 Std.	Erstellen erster GUI Moqups auf der Seite www.moqup.com	Ich hatte keinerlei Erfahrung im Bereich GUI Bau weshalb das erstellen der ersten Vorlagen etwas länger dauerte
12.04.2015	2.0 Std.	Erstellen der restlichen GUI's nach besprochener Vorlage	Mithilfe von Sandro und Luca konnte ich die genauen Inhalte der GUI's ausarbeiten und so die Vorlage fertigstellen

16.04.2015	0.5 Std.	Moqups auf Github geladen und fertiggestellt	Letzte verfeinerungen
16.04.2015	1.5 Std.	Erstellen der ersten GUI's mit Sceneviewer anhand von Moqups	Ich bekam Feedback von Luca und Sandro. Mir wurde gesagt was ich anpassen muss.
21.04.2015	1.3 Std.	Mangelstatus und Mangel Modelle erstellt	Sandro erstelle eine gute Vorlage anhand welcher ich meine Files schreiben konnte
22.04.2015	3.0 Std.	Ich habe am GUI weitergearbeitet und meine Modelle verbessert	Weiter Vertiefung in Arbeit mit Scenebuilder und den verschiedenen Panes
24.04.2015	3.0 Std.	Mangelstatus und Mangel Modelle inklusive NamedQuerries in alle Schichten implementiert	Der Aufwand war grösser als erwartet. Vor allem die vielen NamedQuerries waren sehr Aufwändig
27.04.2015	4.0 Std.	Ich habe die Implementierung von Mangel / Mangelstatus abgeschlossen und alle Sichten des GUI's auf Github publiziert	Das GUI war hier noch nicht Final. Ich habe alle nötigen Felder und Buttons wie mit Sandro und Luca abgesprochen implementiert
30.04.2015	3.0 Std.	Bugfixes, Vorgabe von FX:ID's für das GUI und arbeiten an der Skalierbarkeit des GUI's	Die Skalierbarkeit war ein grosses Problem aufgrund fehlendes Know-hows
01.05.2015	1.0 Std.	Aussere Views überarbeitet (Tableviews) und Bugfixes damit Sandro / Luca weiterarbeiten können	Alexander Hauck hat mir Hinweise gegeben und ich habe diese benutzt um unsere GUI's zu verbessern
03.05.2015	2.0 Std.	Tableview Columns skalierbar gemacht und InnereView komplett überarbeitet mit neuen FXID's anhand Vorlage von Sandro	Sandro hat mir gesagt wie ich die InnereView anpassen muss damit es für ihn optimal ist
04.05.2015	3.0 Std.	InnereViews überall überarbeitet und neue Buttons für weitere Funktionen hinzugefügt	Teils waren die Informationen unklar übermittelt worden und teils gab es neue Felder, die ich einbauen musste
06.05.2015	4.0 Std.	Neue Views zum Erstellen von Entitäten (ADD_GUI's) erstellt	Grösstenteils anhand der InnerenViews
07.05.2015	3.8 Std.	Skalierbarkeit mit Gridpanes finalisiert	Alexander Hauck hat mir erklärt wie ich die Skalierbarkeit hinkriege anhand von seines GUI's
08.05.2015	0.2 Std.	Kleinere Fixes am GUI	Janik von Rotz hat mir einige kleinere Hinweise gegeben wie ich ein paar meiner Probleme lösen kann



11.05.2015	3.5 Std.	LoginView überarbeitet und erstes CSS erstellt	Das CSS erstellen war für mich neuland und brauchte deshalb Einarbeitungszeit. Ich musste anschliessend auch alle Views überarbeiten um die neu entstandenen Bugs zu fixen
14.05.2015	9.5 Std.	Datenbank mit Datengefüllt anhand CreateEntityTest und dann die Daten mit EntityTest getestet	Unsere bisherige Datenbank war "minderwertig" und ich erstellte deshalb eine neue Datenbank mit richtigen Werten, mit denen wir auch arbeiten konnte. Dies hat uns zahlreiche Bugs aufgezeigt
15.05.2015	1.5 Std.	Entitytest finalisiert und Fehler mitgeteilt	Fleissarbeit. Ich musste lediglich die übrigen Funktionen einfügen.
16.05.2015	1.5 Std.	GUI vollendet	Ich habe alle Buttons und Tabellen neu angepasst
16.05.2015	1.5 Std.	CSV Export für Mangel Tableview hinzugefügt	Die SourceForge Seite für die benötigte Library ist zurzeit nicht ansprechbar. Deswegen habe ich eine alternativ Lösung gebastelt was zusätzlich Zeit kostete.
17.05.2015	6.5 Std.	Client RMI Test geschrieben	Die über 50 Tests waren sehr aufwandreich. Vor allem da ich die zu testenden Methoden nicht selbst geschrieben habe und anhand der Namensgebung nicht immer klar war was gemeint ist
18.05.2015	3.0 Std.	Tests, Export Funktion und CSS finalisiert	Mithilfe von Sandro und Luca konnte ich die Funktionen ergänzen und teils falsche Funktionen anhand meiner Tests anpassen
18.05.2015	3.0 Std.	Dokumentation von Tests und Client sowie ausfüllen ManagementSummary	Schlussarbeit und Zusammenfassung des gelernt



21.5 Cihan Demir

INM 21 - Individuelles Portfolio			
Name:	Cihan Demir		
Datum eventuell Phase	Dauer Aufwand	Tätigkeit / Aufgabe	Bemerkung / Erkenntnis
06.03.2015	1 Std.	GitHub Account erstellen und die Desktop-Anwendung davon installiert, konfiguriert. Repository geklont	Endlich weiss ich wie GitHub & Git funktioniert und wann es zur Anwendung kommt!
18.03.2015	1.5 Std.	Use Case006 in Tabellenform erstellt.	
18.03.2015	1 Std.	arc42 Dokument für unser Java Projekt angepasst, alle Füllinformationen entfernt und die zusätzlich nötigen Kapitel erstellt.	
30.03.2015	2.5 Std.	Enterprise Architect (EA) installiert. UseCase006 Diagramm mit dem EA erstellt.	Es hat eine kleine Eingewöhnungsphase gebraucht um zuerst einmal mit den Grundfunktionen klarzukommen.
08.04.2015	1.5 Std.	UseCase006 Aktivitätsdiagramme erstellt mit dem EA erstellt.	
09.04.2015	2 Std.	Das fehlende Image des UseCase002 mit dem Enterprise Architect erstellt. arc42: alle UseCases hinzugefügt.	
22.04.2015	3 Std.	Mithilfe der GitHub Desktopanwendung ein Klon unserer momentanen Workbench erstellt und im Eclipse verlinkt. Die beiden Models Meldung.java und Meldungstyp.java anhand des ERD implementiert. Die nötigen Anpassungen im Buildpath angepasst (Benötigte Projekte und libraries)	



23.04.2015	4 Std.	Meldung.java und Meldungstyp.java richtig formatiert. Die Interfaces MeldungManager.java, MeldungstypManager.java und die Klassen MeldungManagerImpl.java, MeldungstypManagerImpl.java für die Business Schicht implementiert. Die Interfaces MeldungDAO.java, MeldungstypDAO.java und die Klassen MeldungDAOImpl.java, MeldungstypDAOImpl.java für die Persister Schicht implementiert. Die Interfaces MeldungRO.java, MeldungstypRO.java und die Klassen MeldungROImpl.java, MeldungstypROImpl.java für die RMI Schicht implementiert.	
26.04.2015	3 Std.	Importe, Kommentare, Formatierungen für die obigen Klassen und Interfaces angepasst und korrigiert. Zusätzliche Methoden für alle der obigen Klassen und Interfaces implementiert und kleine Fehler im Code verbessert.	Immer ganz genau auf Gross- bzw Kleinschreibung achten, Fehler schleichen sich sehr schnell ein. Eclipse automatische Import organizer Shortcut Strg+Shift+O gelernt.
27.04.2015	2.5 Std.	Alle Pakete der verschiedenen Schichten Model, Persister, RMI, Business Organisiert und angepasst um eine bessere Übersicht zu gewährleisten.	Eclipse macht z.T. Probleme mit dem ausschneiden und einfügen von einem Paket in ein anderes, kopieren und einfügen funktioniert besser. GitHub gab eine Fehlermeldung da ich, das Löschen und Neuerstellen der Klassen und Pakete nicht auf einmal Synchronisieren konnte. Deswegen zuerst synchronisieren, dass es gelöscht wird und danach die neue Struktur synchronisieren.
30.04.2015	3 Std.	arc42: Kapitel 1. "Einführung und Ziele" hinzugefügt, Einstieg angepasst, einige Teile direkt übernommen. Kap. 1.1 "Aufgabenstellung" übernommen Kap. 1.2 "Qualitätsziele" anfänglich Stichworte notiert Kap 1.3 "Stakeholder" Rollen aufgeschrieben.	Mike Iten und ich haben gemerkt, dass es wenig Sinn macht zu zweit an gleichen Kapiteln zu arbeiten weswegen wird die arc42 Dokumentation auf OneDrive hochgeladen um ein paralleles Arbeiten am gleichen Dokument zu ermöglichen. Nur bei grösseren Veränderungen wurde das Dokument auch auf GitHub committed.



04.05.2015	3.5 Std.	<p>Fehler in der Klasse Meldung.java der Model Schicht korrigiert. Dummy Data anhand der CreateEntityTest.java Klasse erstellt um zu teste ob Informationen im GUI korrekt angezeigt werden. Persistence.xml Datei der betreffend der Test Klasse so angepasst, dass zuerst alle tables gedropt und danach erstellt werden, da einige Felder Testweise mit null werten gefüllt worden sind und Fehler erzeugt haben. Importe angepasst.</p> <p>arc42: Kap. 1.3 "Stakeholder" ergänzt.</p>	<p>Beim Erstellen von Dummy Data für die Datenbank um zu testen ob die Informationen in der GUI angezeigt werden habe ich gemerkt das die Reihenfolge der Instanzvariablen nicht gemäss ERD implementiert wurden.</p>
07.05.2015	3.5 Std.	<p>arc42: Kap. 2.1 "Technische Randbedingungen" Kap. 2.2 "Organisatorische Randbedingungen" Kap. 2.3 "Konventionen" ergänzt.</p>	<p>Die Aufgabenstellung noch einmal genau lesen um die "versteckten" Randbedingungen zu eruieren.</p>
15.05.2015	5.5 Std.	<p>arc42: Kap. 4 "Lösungsstrategie" Kap. 8.3 "Typische Abläufe" im Microsoft Visio visualisiert Kap. 8.4 "Persistenz" Kap. 8.10 "Sicherheit" Kap. 8.15 "Management des Systems & Administrierbarkeit" Kap. 8.16 Logging, Protokollierung, Tracing Kap. 8.20 "Internationalisierung" Kap. 8.21 "Migration" Kap. 8.22 "Testbarkeit" Kap. 8.24 "Hochverfügbarkeit" Kap. 12. "Glossar" (Teilweise) ergänzt</p> <p>alle Use Case Tabellen nach unserem Standard-Tabellen-Style angepasst.</p>	
16.05.2015	1 Std.	<p>Plan für weiteres Vorgehen der arc42 Dokumentation erstellt, offene Punkte erfasst und Aufgaben weitergeleitet.</p>	<p>Hier konnte ich auch einmal Aufgaben aufteilen, was mich ein bisschen in den Genuss der Projektleiterfunktion gebracht hat.</p>



17.05.2015	5 Std.	<p>Präsentationsvorlage mit den relevanten Punkten für die anstehende Präsentation erstellt. Folie "Verwendete Technologien" direkt erledigt.</p> <p>arc42: Alle Tabellen richtig formatiert. Allgemeine Dokumentformatierungen angepasst. Kap. 12. "Glossar" erledigt. Kap. 17. "Deployment Infos" ergänzt.</p>	
18.05.2015	6.5 Std.	<p>Präsentationsvorlage besprochen, angepasst und mit weiteren Punkten ergänzt. Image vom Use-Case008 Diagram, mit dem Enterprise Architect erstellt.</p> <p>arc42: UseCase008 Diagramm-Image ergänzt. Alles Use Cases mit den betreffend Autoren versehen. Kap. 8.3 "Typische Abläufe" im Microsoft Visio angepasst & ergänzt. Kap. 2.1 "Technische Randbedingungen" angepasst & verbessert. Kap. 2.2 "Organisatorische Randbedingungen" angepasst & verbessert. Viele Kapitel auf Rechtschreibung überprüft und korrigiert.</p>	
19.05.2015	ca. 8.5 Std.	<p>Gesamter Zeitplan & individuelles Management Summary erstellen.</p> <p>arc42: Kap. 20. "DB - Dokumentation" Alle übrigen Dokumente in arc42 ergänzen (Portfolios, Mgt. Summary, Zeit Projektmitglied Zuordnung, Anwesenheit) Den ganzen Code des Mängel-Manager einbauen. Letzte Arbeiten & Feinschliff der arc42 Dokumentation. Drucken der arc42 Dokumentation.</p>	Zeitplan hat wahrscheinlich insgesamt mehr Zeit gekostet als hier eingeplant, da sie über die ganze Zeit hinweg gepflegt wurde.
20.05.2015	ca. 3 Std.	<p>Präsentation vervollständigen, anpassen und letzte Korrekturen vornehmen. Für Präsentation vorbereiten.</p> <p>arc42: Doku in Elektronischer- und Papierform im Sekretariat bis 12:00 Mittag abgeben.</p>	
21.05.2015	ca. 0.7 Std.	Präsentation	

22. Weitere Dokumentationen

22.1 Individuelle Management Summarys

22.1.1 Luca Kündig

Wichtigste Aktivität(en) im Projekt Mängel-Manager :

Organisatorisches 25%

Als kurzfristiger Projektleiter habe ich ca. ab Mitte Projekt die Leitung übernommen. Hauptsächlich habe ich hier die Aufgaben geplant und an die Gruppenmitglieder verteilt. Ebenfalls habe ich den Output der Gruppenmitglieder kontrolliert.

Architektur 15%

Eigentlich wäre ich hauptsächlich als Architekt geplant gewesen, das ist aber durch meine Übernahme des Projektleiters etwas zu kurz gekommen. Ich habe sämtliche Klassendiagramme erstellt (Dokumentation Kapitel 16) und die UseCases U003, U004.

Programmierung 50%

Von mir sind die Models: Subunternehmen, SuMitarbeiter, ProjektSuMitarbeiter, ProjektGuMitarbeiter, GuMitarbeiter. Ebenfalls von mir sind die Klassen für die vorherigen Models in den Schichten Persister, Business, RMI.

Weiter sind von mir alle Native Queries wie z.B. alle in der Klasse ProjektDAOImpl. Ich habe diverse Views für den Inneren Client erstellt z.B. addPersonController, PersonDetailController, MangelController, addMangelController, MangelDetailController, MeldungDetailController, addMeldungController, MeldungController, SubUnternehmenController, SubUnternehmenDetailController, addUnternehmenController. Den gesamten Webservice habe ich alleine eingerichtet ebenso wie den WebClient (Client-Extern)

Weitere Aktivität(en):

Dokumentation 10%

Von mir ist der Text bei Kapitel 6 Laufzeitschicht, Alles von Kapitel 7 Verteilungsschicht, ca. 40% vom Kapitel 8 Konzepte, Kapitel 16 Klassendiagramme, Kapitel 17 Deployment-Infos, Abschnitt 23.3 Aussergewöhnliches



22.1.2 Sandro Ritz

Als Softwarearchitekt und Ingenieur gehörte vor allem Programmieren zu meinen Hauptaufgaben. Da der ehemalige Projektleiter nicht weiter am Projekt teilnahm kam mir auch noch die Position als Stellvertretender Projektleiter zu. Der erhöhte Aufwand liess sich aber durch interessante Abwechslung gut meistern. Nach dem die Requirements erarbeitet wurden und die Use Cases definiert und gezeichnet waren, widmete ich mich der Software Architektur. Da ich finde, dass man Programmieren nur lernen kann wenn man ein Projekt als Ganzes umsetzt, setzte ich mir ein Ziel: Am Schluss des Projekt sollen alle Projektmitglieder an allen Bereiche etwas programmiert haben.

Um dieses Ziel zu erreichen erstellte ich Musterbeispiel. Als erstes programmierte ich meine zugeteilten Models (Projekt, Person, Bauherr, Adresse, Plz, Projektstatus, Arbeitstyp, Objekttyp). Anschliessend implementierte ich für diese Models alle Schnittstellen (Persister, Business, RMI). So konnten die Teammitglieder sich an meinen Beispielen orientieren und dasselbe für ihre zugeteilten Models programmieren.

Wichtigste Aktivität(en) im Projekt Mängel-Manager:

Wie bereits erwähnt war meine Hauptaufgabe die Programmierung der Applikation. Dazu gehören die Entwicklung des Konzepts und die Einrichtung der Entwicklungsumgebung aller Projekte mit Versionsverwaltung. Dadurch durfte ich neue Kenntnisse erwerben, unter anderem das Tool GitHub kennenlernen. Die meiste Zeit beanspruchte die Implementierung der Controllerkomponenten des Clients. Die Arbeitsbereiche fingen sich allmählich an zu überschneiden, so arbeitete man auch ab und zu an den Ansichten, Controllern von anderen Models. Gegen Schluss des Projekts investierte ich meine Zeit weniger mehr auf den Funktionsumfang - z.B übernahm ich meine Suchfunktion in der Projektansicht aus zeitlichen Gründen nicht mehr auf die anderen Ansichten - sondern behob Problem im Projekt die sich mit der Zeit eingeschlichen haben. (Etwa 65% meines Aufwands).

Neben der Programmierung setzte ich einige Zeit in die Unterstützung und Hilfestellung für die Teammitglieder ein. (Etwa 20-25% meines Aufwands)

Weitere Aktivität(en):

Als Stellvertretender Projektleiter gehörte auch die Vergabe von Aufträgen zu meinen Aufgaben. Diese mussten koordiniert, kontrolliert und verbessert werden. (Etwa 10-15% meines Aufwands).



22.1.3 Mike Iten

Meine Aktivitäten im Projekt

40 % der Dokumentation, unter anderem die Einführung und Ziele, Kontextabgrenzung, Qualitätsszenarien, Requirements, Funktionalen Tests, UseCases und Risiken

10% Desweiteren noch einen kleinen Teil bei den Models programmieren (Models Login und Rolle für jede Ebene, Persister(DAO), Buisness, RMI, und Model)

100 % Das funktionale Testing als „Kunde“ der den Manager benutzt.

30 % bei den GUI Entwürfen mitgearbeitet und mit Herr Monticoli besprochen. Unter anderem auch die Moqups erstellt.

10% UseCase008 und Activity Diagram erstellt für die Filterung.



22.1.4 Mike Monticoli

Wichtigste Aktivität(en) im Projekt Mängel-Manager:

Erstellen GUI & Controller 50%

Die GUI's sind im Code im Bereich Client\Intern\ SRC\Views zu finden. Ich habe sämtliche Views selbst mit Scene Builder / Eclipse erstellt.

Die Controller sind ebenfalls im Client Intern unter Controller zu finden. Ich habe nur in den Controllern nur einige Funktionen wie den Export von Mängeln in die .csv Datei geschrieben. In der Dokumentation gehe ich auf GUI im Kapitel 9 Entwurfsentscheidungen ein.

Tests von RMI & Client 30%

Ich habe sämtliche JUnit4 Tests geschrieben. Zu finden sind diese im Persister Test Folder als CreateEntityTest (Seed File) & EntityTest. Im EntityTest habe ich alle Methoden des Files ProjektDAO getestet.

Desweiteren habe ich den ClientRMI getestet im Bereich Client\Intern\Test als ClientRMITest. In diesem File habe ich alle Methoden des ClientRMI's getestet.

In der Dokumentation gehe ich auf die Tests im Kapitel 18 ein.

Weitere Aktivität(en):

10% UseCase und Activity Diagram erstellen sowie Requirements & Usecases definieren

Zu Beginn des Projekts haben wir in der Gruppe die Requirements und die dazugehörigen Usecases definiert. Anschliessend kümmerte ich mich um den UseCase005 und habe auch das dazugehörige Activity Diagramm erstellt.

In der Dokumentation ist es im Kapitel 15.5 zu finden.

10% Modells erstellen

Ich habe das Model von Mangel sowie Mangelstatus auf den drei Schichten (Business – RMI – Persister) implementiert. Der Code ist an den folgenden Stellen auffindbar:

rmi\mangel

rmi\mangelstatus

persister\dao\mangel

persister\dao\mangelstatus

mangelmanager\model



22.1.5 Cihan Demir

Wichtigste Aktivität(en) im Projekt Mängel-Manager:

Als Dokumentations-Leiter war ich nach dem Bereitstellen der arc42-Dokumentation grundsätzlich mit der schrittweisen Erarbeitung der einzelnen Kapiteln beschäftigt. Dementsprechend habe ich auch einen erheblichen Teil meiner Projektzeit in die Dokumentation investiert (detaillierte Kapitelangaben im individuellen Portfolio zu finden). Auch bei Fragen und Problemen betreffend der Dokumentation diene ich stets als Ansprechperson und habe die Projektmitglieder bei jeder möglichen Gelegenheit unterstützt. Auch das Anpassen der Formatierungen und erledigte Dokumentationsarbeiten in die arc42-Dokumentation einzuflechten gehörten zu meinem Tätigkeitsbereich.

arc42-Dokumentation (80%)

Weitere Aktivität(en):

Neben der arc42-Dokumentation war ich natürlich auch in der Implementierung des Mängel-Managers tätig. Wobei ich auf Basis der Models Meldung.java und Meldungstyp.java, etliche Klassen und Interfaces in den verschiedenen Schichten RMI, Persister, Business implementiert habe (detaillierte Klassen- und Interfacenamen im individuellen Portfolio zu finden). Zusätzlich

Implementierung (15%)

Zusätzlich habe ich die Beschreibung des UseCase008 abgefasst inklusive der Visualisierung des Diagramms und Aktivitätsdiagramms, welches den Login des Users auf dem GUI beschreibt.

Use Case Beschreibung & Visualisierung (5%)



22.2 Aussergewöhnliches

22.2.1 Weitere Diagramme und Dokumente

Wichtige Diagramme und Dokumente sind in Elektronischer Form in den Verzeichnissen mit dem Pfad „Laufwerksbuchstabe:\07_arc42\“ zu finden.

22.2.2 Projektleiter verlässt das Team

Zu Beginn des Projekts hat sich Max von Gellhorn freiwillig als unser Projektleiter gemeldet. Diese Rolle nahm er dann anfangs auch wahr. Mit der Zeit wurde sein Engagement jedoch immer schwächer. Als wir kaum mehr Rückmeldung oder überhaupt etwas von ihm gehört haben, hat Luca Kündig in seiner Rolle als stellvertretender Projektleiter die Aufgaben von Max von Gellhorn temporär übernommen.

Später meldete er sich dann doch noch. Er wolle sich wieder in das Projekt einarbeiten und seine Position wieder übernehmen. Jedoch geschah das nie und so hat Luca Kündig das Amt des Projektleiters ganz übernommen.

Dies führte dazu dass unsere ganze Aufgabenplanung durcheinander geriet. Was wiederum zu erhöhtem Zeitdruck führte. Durch die Absenz von Max von Gellhorn fehlte es uns einerseits an Know-How als auch an Man-Power. Darum konnten wir diverse Funktionen wie z.B. ein funktionierendes Berechtigungskonzept nicht umsetzen.



23. Source-Code von selber beschriebenem Code

Siehe nächste Seite

