HSR
HOCHSCHULE FÜR TECHNIK
RAPPERSWIL

FHO Fachhochschule Ostschweiz

Übersetzungsservice für mehrsprachige Applikationen

Studienarbeit

Abteilung Informatik
Hochschule für Technik Rapperswil

Frühjahrssemester 2012

Autoren: Sandro Felicioni, Davide Giampa
Betreuer: Dr. sc. techn. Daniel Keller
Projektpartner: foryouandyourcustomers AG, Pfäffikon ZH
Experte: Dr. sc. techn. Daniel Keller

Abstract

AUSGANGSLAGE

Die Firma foryouandyourcustomers AG in Pfäffikon ZH möchte den heutigen Übersetzungsprozess für mehrsprachige Webapplikationen optimieren, da heutige Prozesse unstrukturiert sind und viele manuelle Schritte beinhalten. Das Hauptproblem liegt in der dezentralen Datenhaltung der Übersetzungstexte, da sie aktuell mehrere Male zwischen Projekt und Übersetzungsbüro ausgetauscht und wieder zusammengeführt werden müssen. Aus diesem Grund soll eine Webplattform realisiert werden, die eine zentrale Verwaltung von Übersetzungen für mehrsprachige Applikationen ermöglicht und viele manuelle Schritte eliminiert. Die Übersetzungen sollen in einer Datenbank gespeichert werden und in das für die Zielapplikation jeweilige Format exportiert werden können. Es müssen Exportformate für Java, PHP und C# unterstützt werden. Zudem soll der Übersetzungsprozess die Rollen Project Manager, Developer, Texter, Translator und Reviewer vorsehen und diese bei ihrer Arbeit unterstützen. Des Weiteren soll die Webapplikation eine REST-Schnittstelle zur Verfügung stellen, die das Anbinden der Webapplikation an andere Dienste ermöglicht.

RESULTATE

Im Laufe der Studienarbeit wurde mit Hilfe der Component Library Primefaces eine beinahe produktreife JSF-Applikation entwickelt. Die Applikation erlaubt die zentrale Verwaltung von Benutzern, Rollen, Projekten und Übersetzungen. Allein durch das zentrale Verwalten der Übersetzungen haben wir den Übersetzungsprozess auf einen Schlag vereinfacht. Zugrunde liegt ein Zustandsmodell, das einen einfachen und sequentiellen Ablauf garantiert. Dabei führt jede Rolle nur zu wohldefinierten Zuständen spezifische Teilaufgaben aus und leistet so seinen Beitrag zum Ganzen. Übersetzungen können in die Formate YAML, PHP, RESX, PDF, CSV, XML, JSON und als Properties exportiert werden. Vorhandene Übersetzungen können flexibel auf eine Vielzahl von Arten gefiltert werden. Unterstützt wird das Filtern nach Zustand, Sprache, Text, Key und Gruppe. Es wurde zudem eine REST-Schnittstelle zur Verfügung gestellt, womit Entwickler neue Keys erfassen und bestehende Übersetzungen exportieren können, ohne sich dabei im Web-GUI anmelden zu müssen. Zusätzlich wurden einzigartige Features wie das logische Gruppieren von Keys implementiert. Typischerweise würde man pro Screen eine eigene Gruppe erstellen, wodurch dem Benutzer sofort ersichtlich wird, auf welchem Screen und somit in welchem Kontext der Key verwendet wird. Abgesehen davon wurde auch noch das Exportieren dieser Gruppen mitsamt den Keys implementiert.

Management Summary

AUSGANGSLAGE

Die Firma foryouandyourcustomers AG in Pfäffikon ZH wünscht sich einen optimierten Übersetzungsprozess für mehrsprachige Webapplikationen, da heutige Prozesse unstrukturiert sind und viele manuelle Schritte beinhalten. Das Hauptproblem liegt in der dezentralen Datenhaltung der Übersetzungstexte, da sie aktuell mehrere Male zwischen Projekt und Übersetzungsbüro ausgetauscht und wieder zusammengeführt werden müssen. Aus diesem Grund soll eine Webplattform realisiert werden, die eine zentrale Verwaltung von Übersetzungen für mehrsprachige Applikationen ermöglicht und viele manuelle Schritte eliminiert. Dies erleichtert die Zusammenarbeit zwischen den verschiedenen Benutzern und erhöht zudem ihre Produktivität. Die Webapplikation LocaleApp bietet bereits ähnliche Funktionalität für Ruby on Rails Applikationen und dient deshalb auch als Referenzprojekt. Im Gegensatz zu LocaleApp soll unsere Webapplikation abgesehen von Ruby on Rails Applikationen auch Java, PHP und C# Applikationen unterstützen. Zudem soll ein klarer Workflow implementiert werden um den Prozess zu beschleunigen und manuelle Schritte zu reduzieren. Die zu entwickelnde Plattform soll folgende Basisfunktionalitäten unterstützen:

- Verwaltung von Projekten
- Verwaltung von Benutzern und Zuweisung an Rollen
- Strukturierter Workflow für die Rollen Project Manager, Developer, Texter, Translator und Reviewer
- Exportmöglichkeit von Übersetzungen für die Programmiersprachen Ruby, Java, PHP und C#

VORGEHEN, TECHNOLOGIEN

Also Softwareentwicklungsprozess wurde agil nach Scrum vorgegangen. In einer ersten Phase wurde die Referenzplattform localeapp.com genauer analysiert. In Phase zwei wurde ein Architekturprototyp basierend auf den gesammelten Erkenntnissen der ersten Phase entwickelt, mit welchem alle Risiken abgedeckt wurden. In Phase drei wurden die gewünschten Features des Kunden implementiert.

Mit Hilfe folgender Technologien wurde das Projekt umgesetzt:

- Java 1.6 als Programmiersprache
- JSF 2.0 als Web-Technologie
- Primefaces 3.2 als GUI-Component Library
- Spring-Framework 3.1.1 als Extension zu JSF und Hibernate
- Hibernate 4.1 als O/R-Mapper
- Jersey 1.12 als REST-Schnittstelle
- Mysql 5.1 als Datenbankserver
- Maven 3.0 als Buildsystem
- SVN 1.7.4 als Source Verwaltungssystem

ERGEBNISSE

Die gewünschten Anforderungen des Kunden wurden innerhalb der verfügbaren 15 Wochen erfolgreich umgesetzt. Die Applikation hat den Übersetzungsprozess, welcher in der untenstehenden Abbildung als IST-Situation dargestellt ist, vereinfacht und optimiert (SOLL). Alle beteiligten Benutzer eines Projekts können durch die neue Webapplikation auf die gemeinsamen Daten zugreifen und diese editieren. Übersetzungen können in die Formate YAML, PHP, RESX, PDF, CSV, XML, JSON und als Properties exportiert werden, welche unter anderem von Ruby, Java, PHP und C# Applikationen verwendet werden. Vorhandene Übersetzungen können flexibel auf eine Vielzahl von Arten gefiltert werden. Unterstützt wird das Filtern nach Zustand, Sprache, Text, Key und Gruppe. Es wurde zudem eine REST-Schnittstelle zur Verfügung gestellt, womit Entwickler neue Keys erfassen und bestehende Übersetzungen exportieren können, ohne sich dabei im Web-GUI anmelden zu müssen. Zusätzlich wurden einzigartige Features wie das logische Gruppieren von Keys implementiert. Typischerweise würde man pro Screen eine eigene Gruppe erstellen, wodurch dem Benutzer sofort ersichtlich wird, auf welchem Screen und somit in welchem Kontext der Key verwendet wird. Abgesehen davon wurde auch noch das Exportieren dieser Gruppen mitsamt den Keys implementiert.

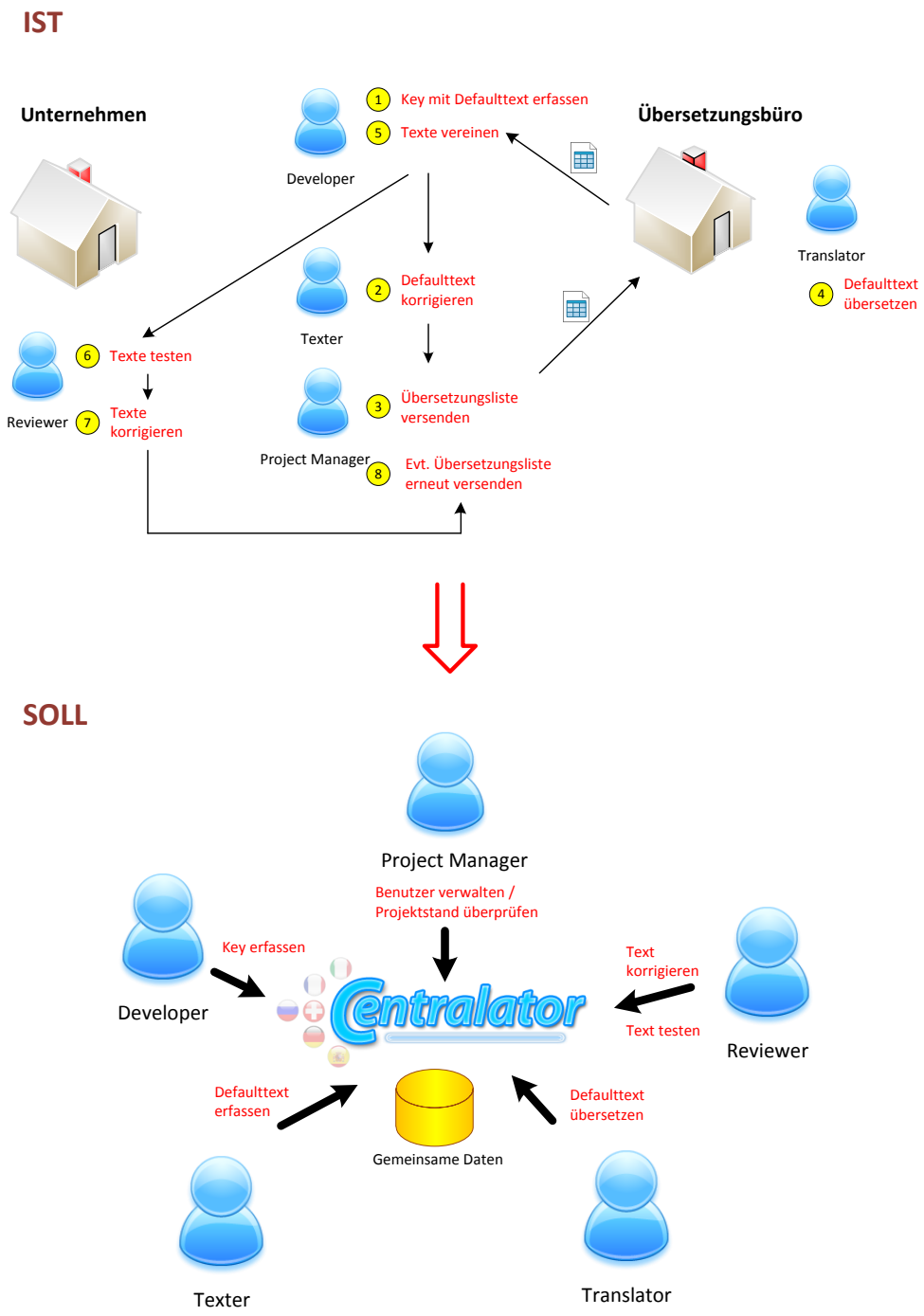


Abbildung 1: Ist- und Soll-Situation des Übersetzungsprozesses

AUSBLICK

Einige Features von Centralator, wie das logische Gruppieren und die REST Schnittstelle, sind einzigartig und nirgends auf dem Markt zu finden. Mit einer Ausarbeitung der Funktionalitäten und einem ansprechenderem GUI könnte die Plattform kommerzialisiert werden und im Markt durchaus bestehen.

Inhaltsverzeichnis

ABSTRACT I

Ausgangslage I
Resultate I

MANAGEMENT SUMMARY II

Ausgangslage II
Vorgehen, Technologien II
Ergebnisse III
Ausblick IV

INHALTSVERZEICHNIS V

1 ANALYSE 1

1.1 Einleitung 1
1.2 Analyse von LocaleApp 1
1.2.1 Ausgangslage 1
1.3 Problemstellung 2
1.4 Ziele und scope 2
1.5 Anforderungsanalyse 3
1.5.1 Nichtfunktionale anforderungen 3
1.5.2 Funktionale anforderungen 4
1.5.3 Workflow 13
1.5.4 Zustandsdiagramm 14

2 IMPLEMENTATION / UMSETZUNG 15

2.1 GUI design 15
2.1.1 Paperprototyping 15
2.1.2 Externes design 19
2.2 Architektur 32
2.2.1 Architekturentscheide 32
2.2.2 Logische Architektur 37
2.2.3 Interaktionsdiagramm Key erfassen 40
2.2.4 Physische Architektur 41
2.2.5 Architekturkonzepte 42
2.2.6 Anbindung an rest-schnittstelle 46
2.2.7 Persistenz 49

3 TESTING 52

3.1 Systemtests 52
3.1.1 Voraussetzungen 52
3.1.2 Bemerkungen 52
3.1.3 Start Page 53
3.1.4 Project Dashboard 55
3.1.5 Project Info 57
3.1.6 Project Structure 59
3.1.7 Project Translations 61
3.1.8 Rest-Schnittstelle 67
3.2 Unit tests / integration tests 70
3.3 Usability-Feedback 71

4 ERGEBNISSE UND SCHLUSSFOLGERUNG 72

4.1 WAS WURDE ERREICHT? 72
4.2 WAS WURDE NICHT ERREICHT 73
4.3 SCHLUSSFOLGERUNGEN 73

ANHANG 75

5 QUALITÄTSMASSNAHMEN 76

5.1 REDMINE 76
5.2 JENKINS 76
5.3 STATIC CODE ANALYSIS 76
5.3.1 PMD 76
5.3.2 Findbugs 77
5.4 TESTABDECKUNG UND CODESTATISTIK 79
5.5 ARCHITEKTUR-ANALYSE MIT STRUCTURE 101 81
5.5.1 Design Level 3 81
5.5.2 Design Level 2 82
5.5.3 Design Level 1 82
5.5.4 Leaf Package 83

6 INSTALLATION UND DEPLOYMENT 85

6.1 ENTWICKLUNGSUMGEBUNG AUFSETZEN 85
6.1.1 Maven dependencies ändern 86
6.2 DEPLOYMENTARTIFAKT GENERIEREN 86
6.3 DEPLOYMENT 87
6.3.1 Datenbank aufsetzen 87
6.3.2 Mysql JDBC Driver verwenden 87
6.3.3 Data Definition Language 87
6.3.4 Spring-Configuration für Datenbank-Verbindung 89
6.3.5 Applikation deployen 89

7 PROJEKTMANAGEMENT 90

7.1 PROJEKTORGANISATION 90
7.1.1 Organisationsstruktur 90
7.1.2 Verantwortung und Aufgaben 90
7.1.3 Externe Schnittstellen 90
7.1 PROJEKTPLAN 91
7.2 SOFTWARENTWICKLUNGSPROZESS 92
7.2.1 SOLL-/IST-Aufwand der wichtigsten user Stories 93
7.2.2 Product backlog 94
7.2.3 Sprint Backlogs und aufwand 94
7.3 ZEITAUFWAND 95
7.3.1 Geleisteter Aufwand 95
7.3.2 Arbeitsverteilung 96

8 PERSÖNLICHE BERICHTE 98

8.1 DAVIDE GIAMPA 98
8.2 SANDRO FELICIONI 99

9 VERZEICHNISSE 100

9.1 ABBILDUNGSVERZEICHNIS 100
9.2 TABELLENVERZEICHNIS 101
9.3 LITERATURVERZEICHNIS 103

10 INHALT CD 105

11 AUFGABENSTELLUNG.....106

12 EIGENSTÄNDIGKEITSERKLÄRUNG.....107

1 ANALYSE

1.1 EINLEITUNG

Heutige Übersetzungsdienste unterstützen die Benutzer im alltäglichen Business. Trotz allem gibt es grundlegende Probleme, welcher jeder Dienst berücksichtigen sollte:

- Übersetzte Texte sind je nach Sprache unterschiedlich lang. Dies wird allerspätestens in der Zielapplikation bemerkt, falls z.B. ein deutscher Text in einer Tabelle keinen Platz findet, der englische Text aber schon.

- Zu übersetzende Texte werden ohne Kontext im System erfasst. Dies kann bei der Übersetzung zu semantischen Fehlern führen und dadurch könnte der Gesamtkontext missverstanden werden. Das Wort **Speichern** könnte im Englischen z.B. nach **Save**, **Backup**, **Submit**, **Post**, oder **Send** übersetzt werden.

- Die Unterscheidung von Texten im Singular und im Plural, sowie die Unterstützung von Platzhaltern müssen gewährleistet sein. Der Satz Es wurde **ein** Datens**atz** gefunden muss auch im Plural definiert sein: Es wurden **{0}** Datens**ätze** gefunden.

- Im Normalfall wird der aktuelle Stand der Keys exportiert, übersetzt und anschliessend wieder importiert. All diese Schritte müssen manuell gemacht werden. Während einer dieser Schritte kann sich der Stand der Keys bereits wieder ändern und beim Importieren könnte ein manuelles Abgleichen der beiden Stände notwendig werden.

1.2 ANALYSE VON LOCALEAPP

1.2.1 AUSGANGSLAGE

Die Online-Plattform LocaleApp [Url01] bietet einen zentralen Sprachübersetzungsservice für Ruby on Rails Applikationen an. Dabei können Benutzer Projekte verwalten, die aus Key/Value-Paaren aufgebaut sind. Pro Key/Value-Paar können verschiedene Sprachen (Locales) hinzugefügt werden, sodass die Values (Texte) für einen bestimmten Key in mehreren Sprachen übersetzt werden können. Übersetzte Texte können anschliessend exportiert und für andere Applikationen verwendet werden. Auf diese Art und Weise können Ruby on Rails Applikationen einfach und effizient anhand einheitlicher Keys internationalisiert werden.
LocaleApp bietet für Ruby on Rails Applikationen sowohl einen Server als auch mehrere Client-Teile an. Der Server-Teil besteht aus einer Webapplikation, welche von allen Beteiligten genutzt wird um insbesondere Keys zu erfassen und Übersetzungen durchzuführen. Die Client-Teile bestehen einerseits aus einem Daemon-Prozess, welcher periodisch die neusten Übersetzungen runterlädt und andererseits aus einer Erweiterung des Rails Frameworks. Die Erweiterung ist zur Unterstützung des Entwicklers gedacht, welche ihm zum einen erlaubt, in real-time die neusten Übersetzung runterzuladen und zum anderen automatisch neu definierte Keys nach LocaleApp hochzuladen.

1.2.1.1 WELCHE PROBLEME LÖST LOCALEAPP?

LocaleApp löst unter anderem folgende Probleme:

- **Komplexität des Übersetzungsprozesses**
  Dank der Zentralisierung dieses Dienstes können alle beteiligten Parteien, wie Developer, Texter, Translator, Project Manager oder Reviewer auf derselben Datenbasis arbeiten. Ein explizites exportieren wie auch importieren wird dadurch überflüssig, sodass der gesamte Übersetzungsprozess einfacher und effizienter wird.

- **Einschränkung der Entwickler-Freiheit beim Übersetzen von Texten**
  LocaleApp bietet speziell für Entwickler von Rails Applikationen einen Update-Mechanismus an (Daemon), der auf dem Entwickler-System installiert werden muss und mit dem zentralen Locale-Dienst kommuniziert. Anhand dieses Mechanismus können Entwickler entweder per Konsolen-Befehl oder automatisch Übersetzungen in LocaleApp importieren bzw. neue Key/Value-Paare erfassen, ohne sich dabei auf der Online-Plattform anmelden zu müssen. Neue Keys bzw. fehlende Values werden ebenfalls erkannt und an LocaleApp bekanntgegeben. Sobald ein Text übersetzt wurde, wird dieser automatisch in die Entwickler-Applikation importiert, nachdem die Applikation neu geladen wurde. Der Entwickler muss somit die Übersetzungsdateien nie selber editieren und verfügt über eine elegante und transparente Update-Lösung.

1.3 PROBLEMSTELLUNG

LocaleApp bietet ihren Dienst nur für Ruby on Rails Applikationen an. Dies stellt eine Einschränkung für andere Programmiersprachen dar.

1.4 ZIELE UND SCOPE

Das Ziel dieser Arbeit ist es, einen Service zu implementieren, der an LocaleApp anlehnt, jedoch Sprachunabhängigkeit gewährleistet, sodass auch Java-, PHP- und eventuell C#-Applikationen unterstützt werden können. Wie im Kapitel 1.2.1 beschrieben wurde, besteht LocaleApp aus einem Server und mehreren Client-Teilen. In dieser Arbeit wird der Fokus jedoch hauptsächlich auf den Server-Teil gelegt. Konkret bedeutet das, dass zum einen eine Webapplikation mit ähnlicher Funktionalität wie LocaleApp entwickelt wird und zum anderen eine REST-Schnittstelle angeboten wird, womit Entwickler über HTTP Keys erfassen und Übersetzungen exportieren können, ohne sich dabei in das Web-GUI anmelden zu müssen. Die zu unterstützenden Use Cases sind dem Kapitel 1.5.2.3 zu entnehmen. Ein wichtiger Grund weshalb auf einen Teil der Client-Funktionalität verzichtet wird, ist die Tatsache, dass nicht alle zu unterstützenden Sprachen & Frameworks so einfach zu erweitern sind wie dies zum Beispiel bei Ruby on Rails der Fall ist.

1.5 ANFORDERUNGSANALYSE

1.5.1 NICHTFUNKTIONALE ANFORDERUNGEN

Nachfolgend werden die nichtfunktionalen Anforderungen beschrieben, die zu Beginn des Projekts definiert wurden.

Server:

- Die Webapplikation muss die geforderte Performance auf einem Server mit 4 GB RAM und einer CPU Leistung von 2.0 GHz erbringen.

Client:

- Die Webapplikation muss vollständig mit den Browsern IE 8 und Firefox 10 funktionieren.

Performance:

- Die maximale Ladezeit einer beliebigen Seite mit einem Projekt mit 100 Keys und 4 Locales darf nicht länger als 5 Sekunden betragen.
- Der Export einer Locale mit 100 übersetzten Texten darf nicht länger als 2 Sekunden betragen.

Mengenanforderung:

- Mit der Webapplikation müssen Projekte mit mindestens 100 Keys und 4 Locales verwaltet werden können.
- Die Webapplikation muss alle UTF-8-fähigen Zeichensätze unterstützen.

Sicherheit:

- Die Webapplikation muss gegen gängige Hacker Attacken wie SQL-Injection geschützt sein.
- Es muss sichergestellt werden, dass nur Projektteilnehmer Zugriff auf die REST-Schnittstelle haben.

1.5.2 FUNKTIONALE ANFORDERUNGEN

Im Folgenden werden sowohl bestehende (von LocaleApp), als auch neue Akteure und Use Cases beschrieben.

1.5.2.1 AKTEURE & STAKEHOLDERS

Die aktuelle LocaleApp-Version ist im Beta-Status (Stand 28.02.2012) und deshalb noch nicht voll ausgereift. Es werden dabei folgende Akteure unterstützt:

Tabelle 1: Bestehende Akteure

Akteure	Ziele
Project Manager	• Erstellung, Löschung und Bearbeitung von Projekten • Benutzerzugriffsverwaltung für Projekte • Verifikation des Projektstatus
Developer	• Erfassung von neuen Keys mit einem Defaulttext
Translator	• Übersetzen des Defaulttextes

Die aufgelisteten Benutzer werden im Rahmen dieser Arbeit um folgende Akteure erweitert:

Tabelle 2: Neue Akteure

Akteure	Ziele
Texter	• Korrektur von Defaulttexten und Freigabe für die Textübersetzung
Reviewer	• Validierung und Korrektur von Defaulttexten und Textübersetzungen • Durchführung von Qualitätstests • Abnahme für die Produktion

1.5.2.2 USE CASE DIAGRAMM

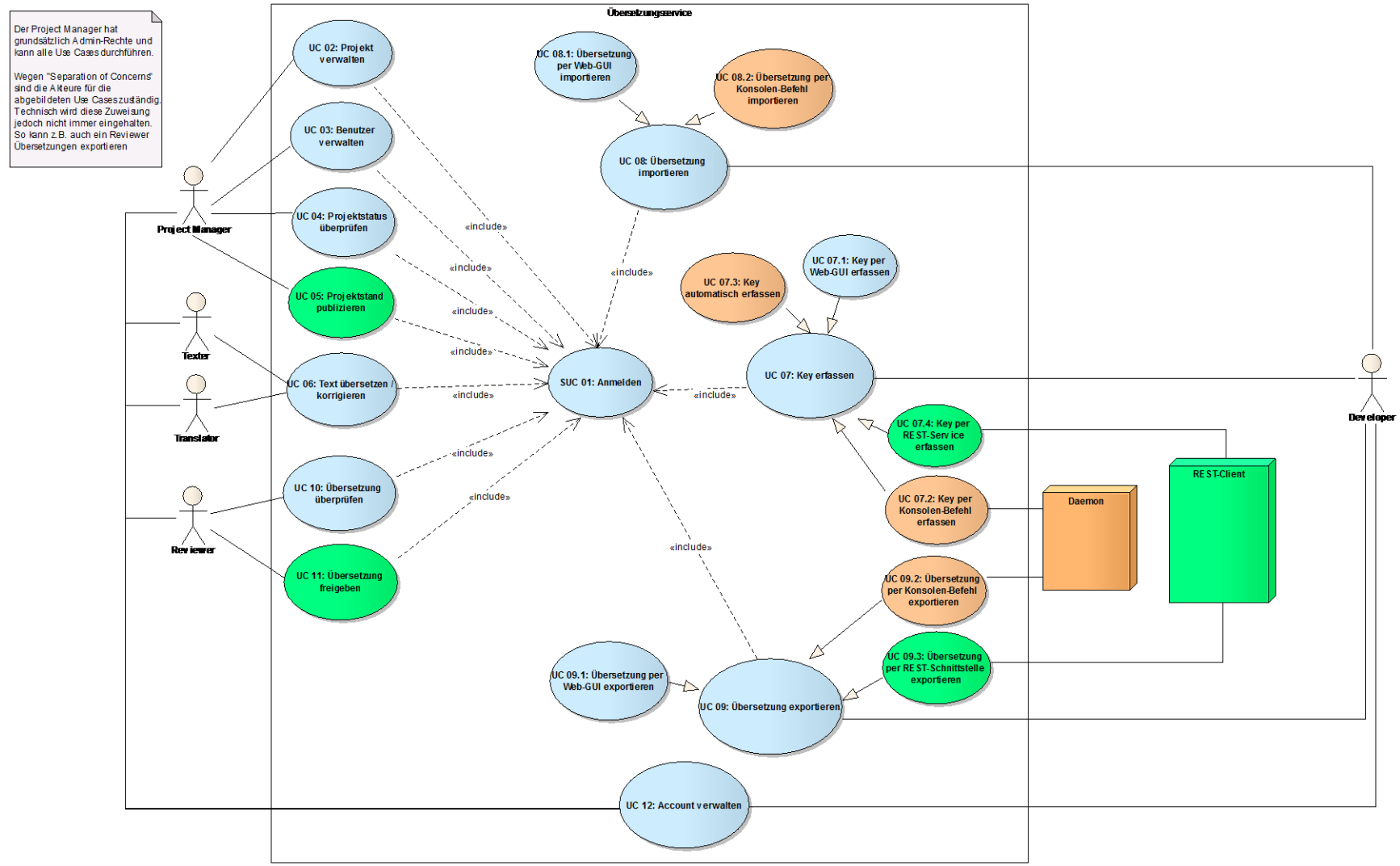


Abbildung 2: Use Case Diagramm

1.5.2.3 USE CASES

Nachfolgend werden die Use Cases von Centralator beschrieben. Die Farben der Use Cases sind mit dem Use Case-Diagramm [Larman11] im Kapitel 1.5.2.2 in Verbindung gesetzt.

Tabelle 3: Farb-Legende zu den Use Cases

Use Case Typ	Farbzuordnung
Bestehende UC's von LocaleApp, welche implementiert werden	
Neue UC's, welche nur von uns implementiert werden	
UC's die nicht implementiert werden, aber von LocaleApp angeboten werden	

Tabelle 4: Allgemeine Use Cases

	SUC 01: Anmelden Ein Benutzer meldet sich mit Name und Passwort beim System an. Falls berechtigt, zeigt das System die Arbeitsumgebung an.

Tabelle 5: Use Cases für Project Manager

	UC 02: Projekt verwalten Ein Project Manager erstellt, bearbeitet oder löscht ein Projekt im System. Das System zeigt den aktuellen Stand des Projekts an.
	UC 03: Benutzer verwalten Ein Project Manager erstellt, löscht oder bearbeitet Benutzer für ein bestimmtes Projekt. Das System zeigt die erfassten Benutzer und deren Rechte gemäss der zugehörigen Rollen an. Bei Erstellung und Löschung von Benutzern benachrichtigt das System die betroffenen Benutzer.
	UC 04: Projektstatus überprüfen Ein Project Manager selektiert ein Projekt. Das System zeigt für jede erfasste Sprache den Stand der übersetzten Texte an.
	UC 05: Projektstand publizieren Ein Project Manager selektiert ein Projekt welches bereits zu 100% übersetzt und reviewed ist und publiziert es anschliessend. Das System gibt den aktuellen Stand der Übersetzungen für den Daemon-Prozess frei.

Tabelle 6: Use Cases für Texter und Translator

	UC 06: Text übersetzen / korrigieren Ein Benutzer führt für einen bestimmten Key eine Übersetzung durch und erfasst den Text im System. Das System speichert den übersetzten Text und zeigt diesen an.

Tabelle 7: Use Cases für Reviewer

	UC 10: Übersetzung überprüfen Ein Reviewer testet die Applikation und findet syntaktische oder semantische Fehler im übersetzten Text. Er meldet sich beim System an nimmt selbst die nötigen Änderungen am Text vor. Das System speichert die überarbeiteten Texte und zeigt diese an.
	UC 11: Übersetzung freigeben Ein Reviewer meldet sich beim System an. Sind syntaktische und semantische Korrekturen erfolgt und für gut befunden, gibt der Reviewer alle erfassten Übersetzungen frei.

Tabelle 8: Use Cases für Developer

UC 07: Key erfassen Ein Developer erfasst einen neuen Key im System. Das System erstellt einen neuen Key und verknüpft den Key mit einem leeren Value.
UC 07.1: Key per Web-GUI erfassen Ein Developer meldet sich via Web-GUI beim System an. Er erfasst einen neuen Key via Web-GUI. Das System erstellt einen neuen Key und verknüpft den Key mit einem leeren Value. Das System zeigt für jede Sprache den neu erfassten Key an.
UC 07.2: Key per Konsolen-Befehl erfassen Ein Developer erfasst einen neuen Key via Konsolen-Befehl. Die Anmeldung erfolgt dabei implizit mit dem Abarbeiten des Konsolen-Befehls. Das System erstellt einen neuen Key und verknüpft den Key mit einem leeren Value. Das System zeigt für jede Sprache den neu erfassten Key an.
UC 07.3: Key automatisch erfassen Ein Developer erfasst einen neuen Key innerhalb einer Template View. Bei der ersten Verwendung dieses Keys innerhalb der Applikation erstellt das System automatisch einen neuen Key und verknüpft diesen mit einem leeren Value.
UC 07.4: Key per REST-Schnittstelle erfassen Ein Developer erfasst einen neuen Key über die REST-Schnittstelle des Systems. Das System erstellt einen neuen Key und verknüpft diesen mit dem mitgegebenen Value.
UC 08: Übersetzung importieren Ein Developer wählt eine Übersetzungsdatei aus und importiert diese in das System. Das System aktualisiert die bestehenden Keys und Values mit den neuen Texten. Falls neue Keys importiert werden, zeigt das System diese ebenfalls an.
UC 08.1: Übersetzung per Web-GUI importieren Ein Developer meldet sich via Web-GUI beim System an. Er wählt eine Übersetzungsdatei aus und importiert diese in das System. Das System aktualisiert die bestehenden Keys und Values mit den neuen Texten. Falls neue Keys importiert werden, zeigt das System diese ebenfalls an.
UC 08.2: Übersetzung per Konsolen-Befehl importieren Ein Developer importiert eine Übersetzungsdatei via Konsolen-Befehl. Die Anmeldung erfolgt dabei implizit mit dem Abarbeiten des Konsolen-Befehls. Das System aktualisiert die bestehenden Keys und Values mit den neuen Texten. Falls neue Keys importiert werden, zeigt das System diese ebenfalls an.
UC 09: Übersetzung exportieren Ein Developer exportiert alle verfügbaren Übersetzungen.
UC 09.1: Übersetzung per Web-GUI exportieren Ein Developer meldet sich via Web-GUI beim System an. Er exportiert alle verfügbaren Übersetzungen. Das System generiert eine Übersetzungsdatei und übergibt diese dem Entwickler.
UC 09.2: Übersetzung per Konsolen-Befehl exportieren Ein Developer exportiert eine Übersetzungsdatei via Konsolen-Befehl. Die Anmeldung erfolgt dabei implizit mit dem Abarbeiten des Konsolen-Befehls. Das System generiert eine Übersetzungsdatei. Das System schickt die Datei an die Applikation des Developers.
UC 09.3: Übersetzung über REST-Schnittstelle exportieren Ein Developer exportiert eine Übersetzungsdatei via REST-Client. Das System generiert eine Übersetzungsdatei. Das System schickt die Datei an die Applikation des Developers.

1.5.2.4 PRODUCT BACKLOG

Nachfolgend eine Liste der zu implementierenden User Stories (US) mit einer Aufwandschätzung in Stunden. Diese Liste stellt das Scrum Product Backlog [Url02] dar.

Tabelle 9: Legende zum Product Backlog

Erforderlichkeit	Kennzeichnung
Must	
Should	
May	

Tabelle 10: User Stories zu Anmelden

1. Task: Anmelden				
	1.1	US:	Ein existierender Benutzer kann sich via E-Mail & Passwort anmelden.	10h
	1.2	US:	Ein Benutzer wird gewarnt, falls seine Anmeldung nicht erfolgreich war.	3h
	1.3	US:	Ein Benutzer der sein Passwort vergessen hat, kann es sich via E-Mail zukommen lassen.	5h
	1.4	US:	Ein Benutzer der drei Mal in Folge das falsche Passwort eingegeben hat, wird für zehn Minuten gesperrt.	8h

Tabelle 11: User Stories zu Projekt verwalten

2. Task: Projekt verwalten				
	2.1	US:	Ein Benutzer kann ein Projekt mit einer Default-Locale erstellen und gilt ab da als Project Manager für dieses Projekt.	10h
	2.2	US:	Der Project Manager kann seine eigenen Projekte löschen.	3h
	2.3	US:	Der Project Manager kann einem Projekt eine neue Locale hinzufügen.	8h
	2.4	US:	Der Project Manager kann eine Locale von einem Projekt entfernen.	3h
	2.5	US:	Jeder Benutzer kann im read-only Modus Projektinfos ansehen. Dies umfasst zum Beispiel die erfassten Locales oder die Projektmitglieder.	6h

Tabelle 12: User Stories zu Benutzer verwalten

3. Task: Benutzer verwalten				
	3.1	US:	Der Project Manager kann seinem Projekt einen Benutzer zuordnen.	12h
	3.2	US:	Der Project Manager kann einen Benutzer von seinem Projekt entfernen.	3h
	3.3	US:	Der Project Manager kann einem Benutzer eine gewisse Rolle vergeben.	4h
	3.4	US:	Der Project Manager kann die Rechte für einen zugeordneten Benutzer bearbeiten.	20h

Tabelle 13: User Stories zu Projektstatus überprüfen

4. Task: Projektstatus überprüfen				
	4.1	US:	Jeder Benutzer kann die Anzahl Übersetzungen pro Locale ansehen.	10h
	4.2	US:	Jeder Benutzer kann die Anzahl Übersetzungen für alle Locales ansehen.	5h

Tabelle 14: User Stories zu Projektstand publizieren

5. Task: Projektstand publizieren			
5.1	US:	Der Project Manager kann einen Snapshot aller Übersetzungen eines Projekts erstellen.	20h
5.2	US:	Der Project Manager kann einen früheren Snapshot eines Projekts importieren und so den früheren Zustand wiederherstellen.	15h
5.3	US:	Der Project Manager kann erst einen Snapshot erstellen, wenn alle Entries eines Projekts freigegeben wurden.	3h

Obwohl dieser Task für diese Arbeit als neuer Use Case definiert wurde, wurden die Prioritäten dieses Tasks während dem Projekt herabgesetzt.

Tabelle 15: User Stories zu Text übersetzen

6. Task: Text übersetzen			
6.1	US:	Der Translator kann für eine Locale einen Text übersetzen und bearbeiten.	20h
6.2	US:	Der Translator kann mehrere Locales anwählen und dafür Texte übersetzen und bearbeiten.	12h

Tabelle 16: User Stories zu Key erfassen

7. Task: Key erfassen			
7.1	US:	Der Developer kann per Web-GUI ein Default Entry erfassen.	12h
7.2	US:	Der Texter kann per Web-GUI den Defaulttext bearbeiten.	8h
7.3	US:	Der Developer kann per Web-GUI einen existierenden Key umbenennen.	8h
7.4	US:	Der Developer kann per Web-GUI einen existierenden Key löschen.	5h
7.5	US:	Der Developer kann per Web-GUI mehrere existierende Keys auf einmal löschen.	20h
7.6	US:	Der Developer kann per Web-GUI einen existierenden Key nicht ein zweites Mal erfassen.	4h

Tabelle 17: User Stories zu Übersetzung importieren

8. Task: Übersetzung importieren			
8.1	US:	Der Developer kann bestehende Entries in Java-Property-Files für ein Projekt per Web-GUI importieren.	25h
8.2	US:	Der Developer kann bestehende Entries in YML-Files für ein Projekt per Web-GUI importieren.	15h
8.3	US:	Der Developer kann bestehende Entries in C#-Files für ein Projekt per Web-GUI importieren.	15h

Tabelle 18: User Stories zu Übersetzung exportieren

9. Task: Übersetzung exportieren			
9.1	US:	Jeder Benutzer kann die erfassten Entries für eine Locale als Java-Property-File per Web-GUI exportieren.	10h
9.2	US:	Jeder Benutzer kann die erfassten Entries für eine Locale als YML-File per Web-GUI exportieren.	5h
9.3	US:	Jeder Benutzer kann die erfassten Entries für eine Locale als C#-File per Web-GUI exportieren.	5h
9.4	US:	Jeder Benutzer kann für ein Projekt ein eigenes Exportformat festlegen und per Web-GUI alle Entries exportieren.	15h
9.5	US:	Der Daemon kann automatisch Entries für eine Locale im gewünschten Format exportieren und ins Zielverzeichnis kopieren.	25h
9.6	US:	Ein Nutzer von Centralator kann über den Webservice die neusten Übersetzungen exportieren und neue Keys erfassen.	20h

Tabelle 19: User Stories zu Übersetzung überprüfen

10. Task: Übersetzung überprüfen			
10.1	US:	Jeder Benutzer kann für ein Projekt alle Entries einer Locale ansehen.	4h
10.2	US:	Der Reviewer kann vorhandene Übersetzungen korrigieren.	4h
10.3	US:	Developer und Reviewer können zugewiesene Entries zurückweisen und einen Kommentar dazu erfassen.	15h
10.4	US:	Ein Texter, Translator, oder Reviewer, der für einen Entry zuständig ist, kann ihn nach seiner Bearbeitung zur nächsten zuständigen Rolle weiterleiten.	4h

Tabelle 20: User Stories zu Übersetzung freigeben

11. Task: Übersetzung freigeben			
11.1	US:	Der Reviewer kann eine Übersetzung für ein Projekt absegnen und somit freigeben.	4h

Tabelle 21: User Stories zu Account verwalten

12. Task: Account verwalten			
12.1	US:	Ein neuer Benutzer kann einen Account erstellen (registrieren).	6h
12.2	US:	Ein Benutzer kann seine Account-Einstellungen einsehen und bearbeiten.	6h

Tabelle 22: User Stories zu allgemeinen Features

13. Task: Allgemeine Features			
13.1	US:	Ein Benutzer kann die Sprache der Webapplikation ändern.	8h
13.2	US:	Ein Benutzer kann innerhalb eines Projekts nach Übersetzungen (Texte) suchen.	3h
13.3	US:	Einem Benutzer werden nur Entries angezeigt, für die seine Rolle zuständig ist.	10h
13.4	US:	Der Benutzer kann den Übersetzungstext in einem Rich GUI Editor erfassen. Dadurch kann der Text grundlegend formatiert werden (fett, kursiv etc.).	10h
13.5	US:	Der Benutzer kann alle erfassten Keys zu einem Projekt in einer hierarchischen Baumstruktur einsehen. Durch einen Klick auf einen Key werden die angezeigten Übersetzungen gefiltert.	15h
13.6	US:	Der Benutzer kann manuell die erfassten Keys zu Gruppen hinzufügen und anzeigen lassen. Durch einen Klick auf eine Gruppe werden die angezeigten Übersetzungen gefiltert.	20h
13.7	US:	Der Benutzer kann die manuell gruppierten Keys mitsamt den zugehörigen Texten in einer beliebigen Sprache exportieren.	12h
13.8	US:	Der Benutzer kann Übersetzungen nach Status filtern.	6h

Zusätzlich zu den erwähnten Features wurden am 18. April 2012 anlässlich der Kurzpräsentation von Centralator bei foryouandyourcustomers folgende Ideen eingebracht, welche zukünftig implementiert werden könnten:

- **Platzhalter**
  Es soll möglich sein, Platzhalter (z.B. Es gibt {0} Studenten) in den Übersetzungstexten zu nutzen und die Singular/Plural-Problematik zu unterstützen.

- **Translation Memory**
  Wenn ein Text in einem Projekt schon mal vorkam, kann es im neuen Projekt vorgeschlagen werden.

- **Google Translate**
  Übersetzungsvorschläge von Google Translate [Url03] anzeigen lassen, um den Translator zu unterstützen.

1.5.3 WORKFLOW

Der Workflow von Centralator ist im untenstehenden Aktivitätsdiagramm dargestellt. Es ist deutlich erkennbar, welche Aufgaben welche Rolle ausführen muss. Das Abarbeiten einer Übersetzung erfolgt immer sequentiell, nachdem ein neuer Key erfasst worden ist. Die Aktivität **Projektstand publizieren** wurde nicht implementiert, zeigt aber, wann und wer einen Snapshot des Projektstandes erzeugen könnte.

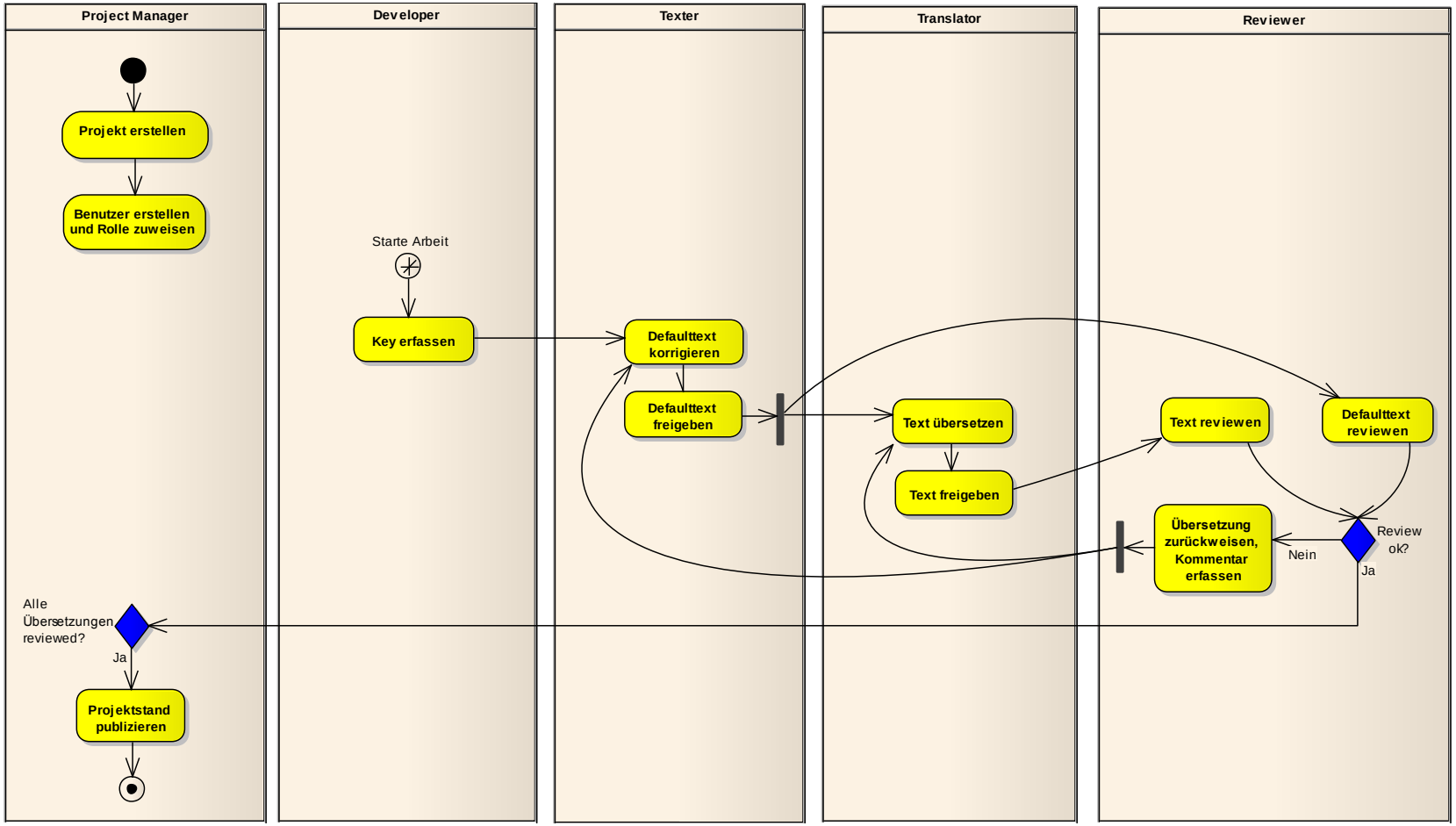


Abbildung 3: Workflow Aktivitätsdiagramm

1.5.4 ZUSTANDSDIAGRAMM

Für den Übersetzungsmechanismus ist eine Zustandsmaschine von zentraler Bedeutung.
Eine Übersetzung durchläuft im Normalfall die Zustände **new, ready, translated, reviewed** und **released** (für Erklärungen siehe Zustandsübergängen unten). Die Zurückweisung einer Übersetzung aus den Zuständen **ready**, **translated** und **reviewed** bewirkt eine Zustandsänderung in den Zustand **rejected**. Eine Übersetzung im Zustand **released** kann in den Zustand **reopened** wechseln, falls für den nächsten Release eine Änderung nötig ist. Vom Reviewer und Texter speziell behandelt werden Übersetzungen für die Default-Sprache. Die Übersetzung für eine Nicht-Default-Sprache geht erst in den Zustand **ready** über, falls die Übersetzung für die Default-Sprache vom Texter freigegeben wurde. Eine Übersetzung für die Default-Sprache, welche sich im Zustand **ready** befindet und vom Reviewer geprüft wurde, überspringt den Zustand **translated**.

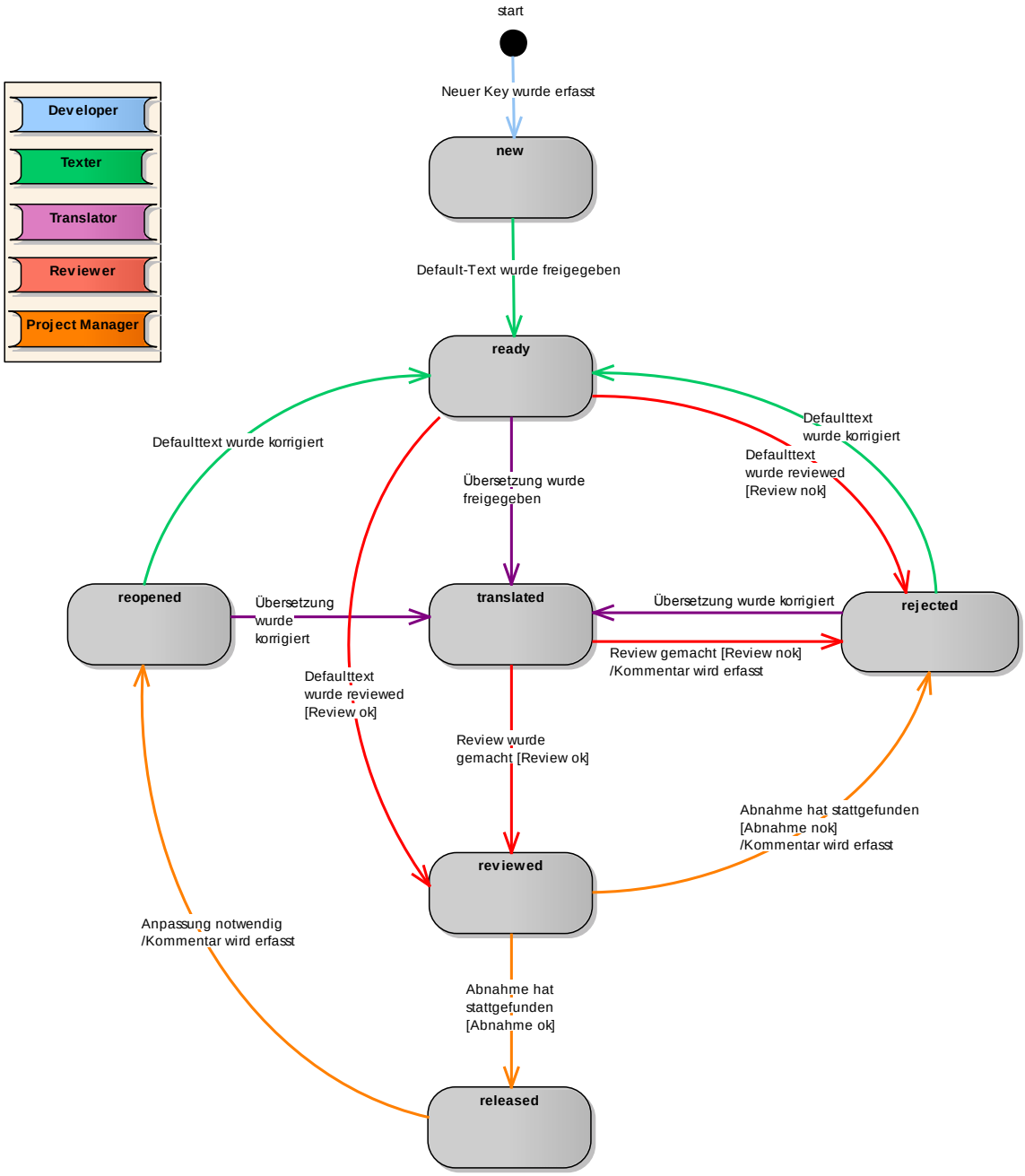


Abbildung 4: Zustandsdiagramm

2 IMPLEMENTATION / UMSETZUNG

2.1 GUI DESIGN

2.1.1 PAPERPROTOTYPING

Die folgenden Paperprototypen wurden mit Hilfe der Wireframe-Software Axure [Url04] konzipiert. Eine detaillierte Beschreibung der Screens und Funktionalitäten ist im Kapitel 2.1.2 vorzufinden.

2.1.1.1 STARTPAGE

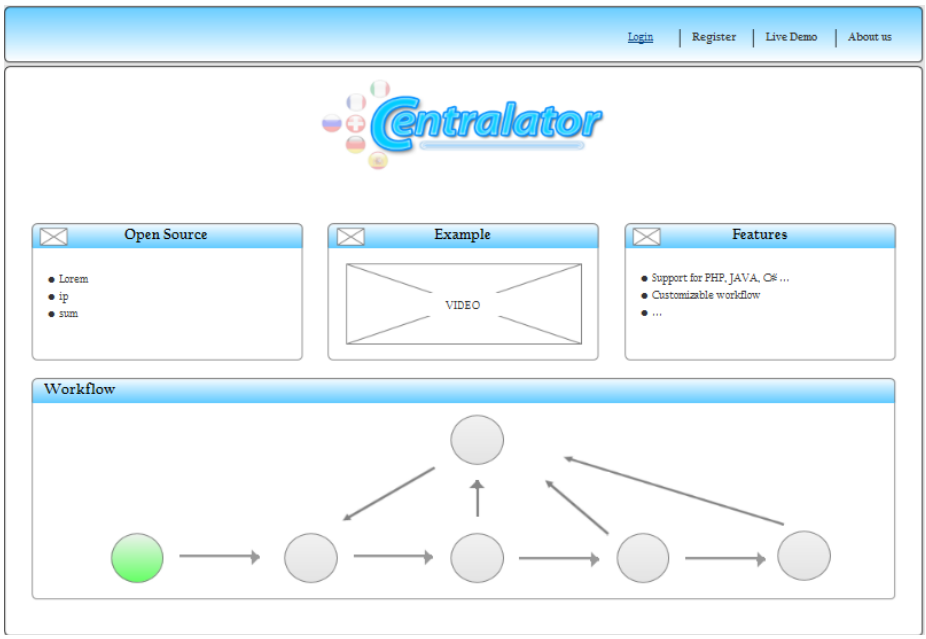


Abbildung 5: Paperprototyp Startpage

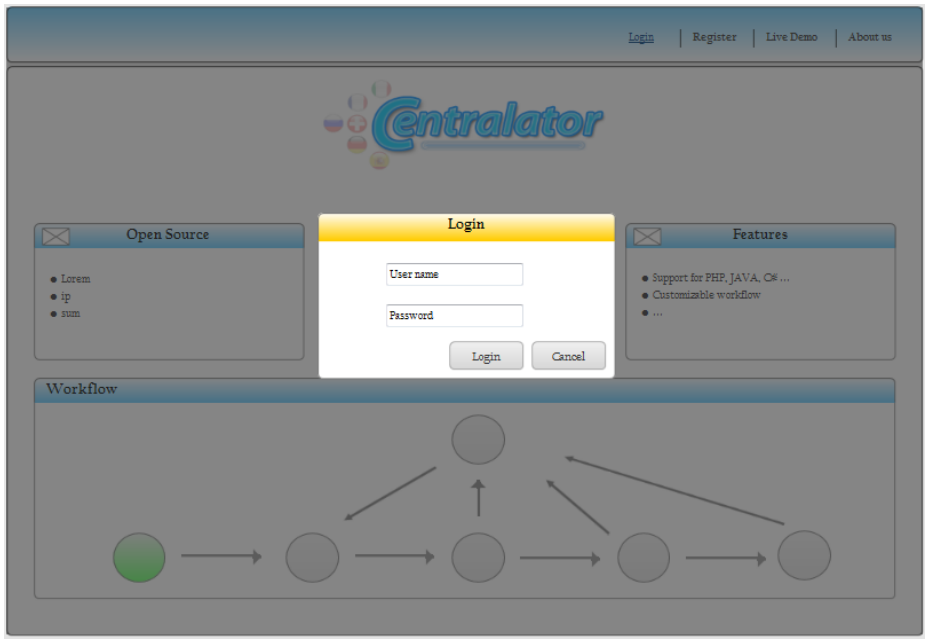


Abbildung 6: Paperprototyp Startpage – Login Dialog

2.1.1.2 PROJECT DASHBOARD

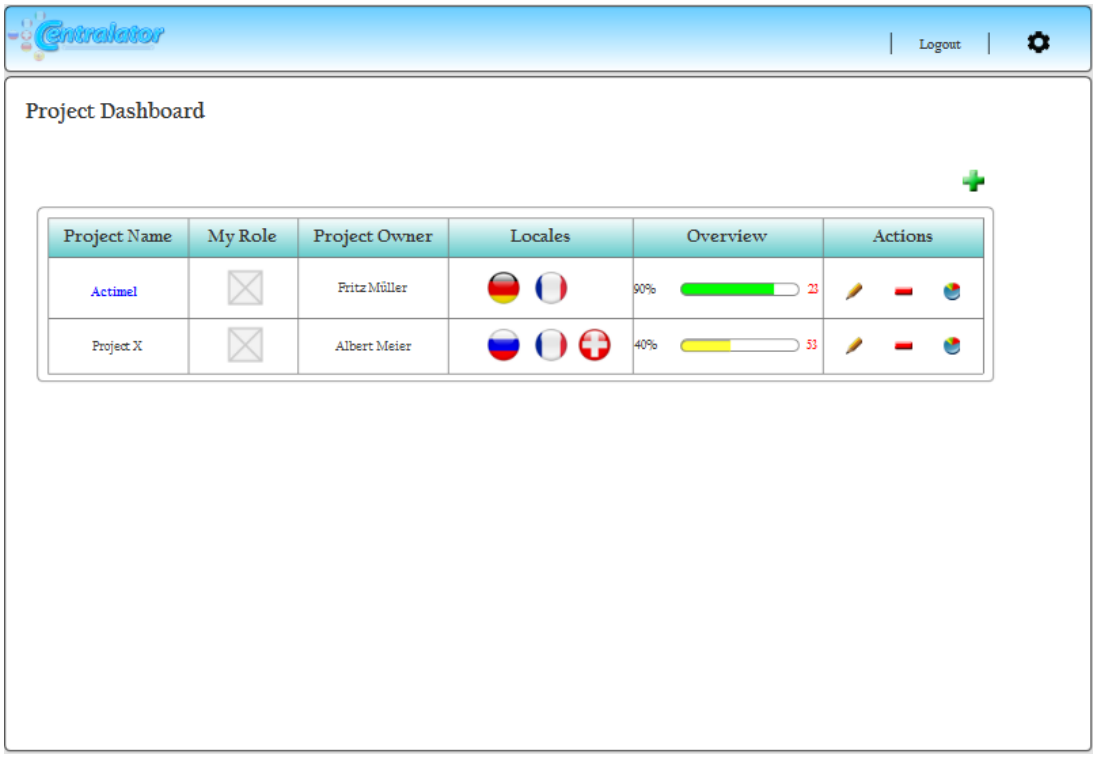


Abbildung 7: Paperprototyp Project Dashboard

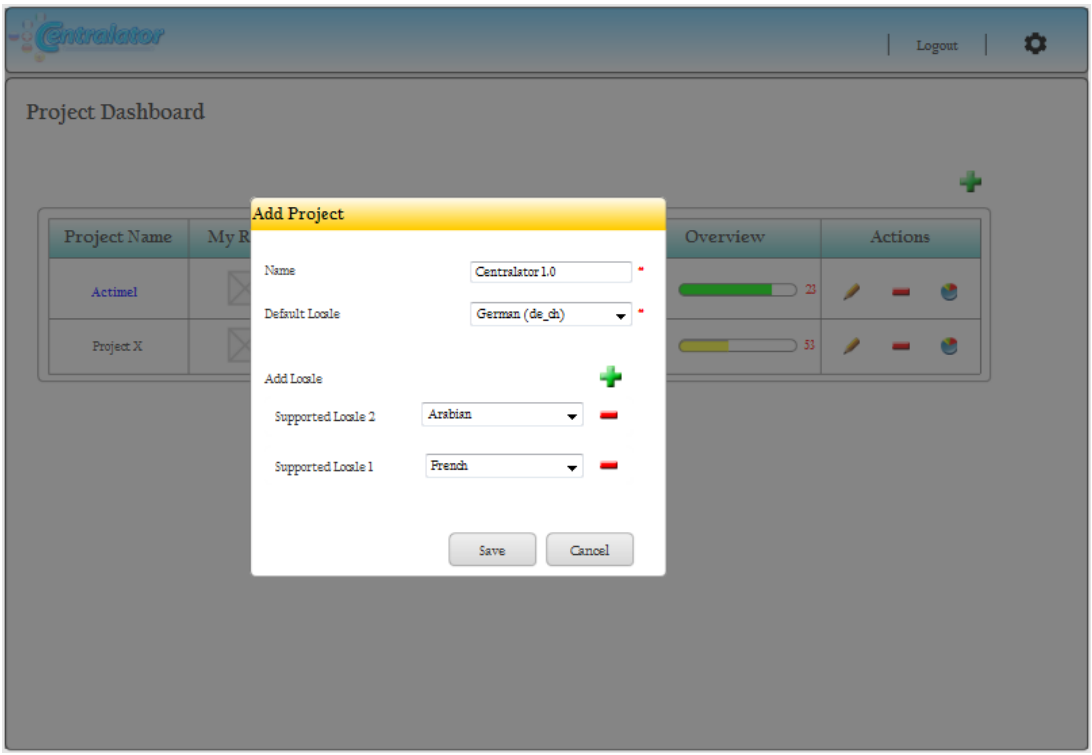


Abbildung 8: Paperprototyp Project Dashboard – Add Project Dialog

2.1.1.3 PROJECT INFO

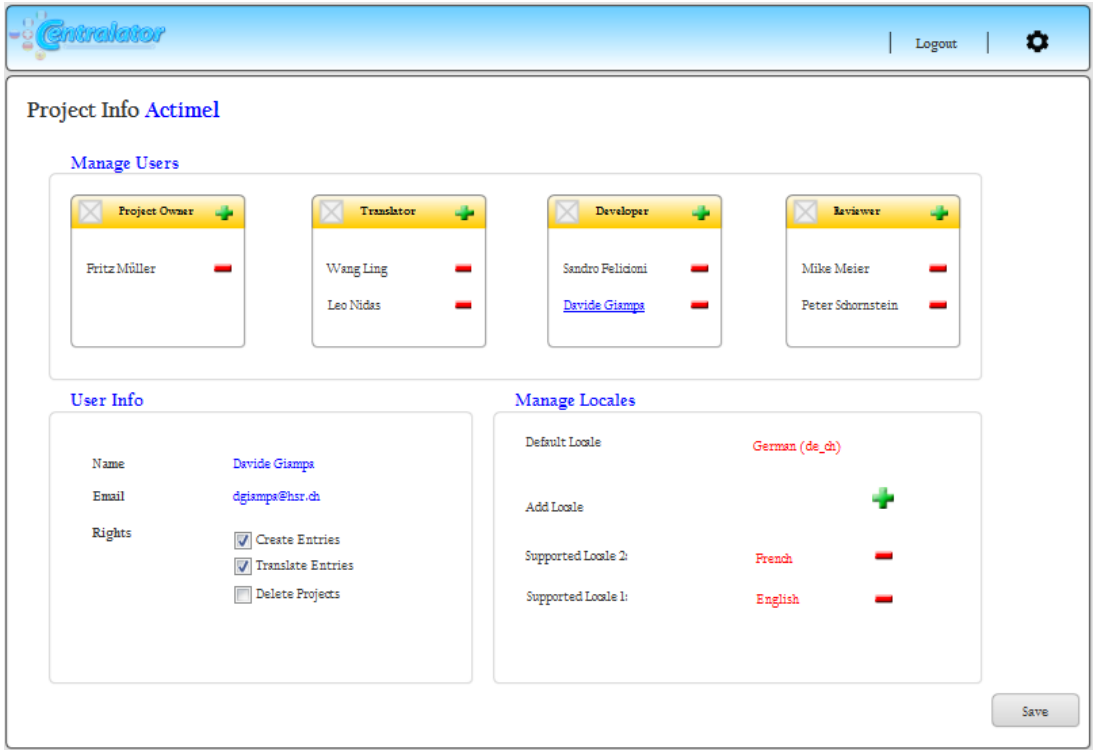


Abbildung 9: Paperprototyp Project Info

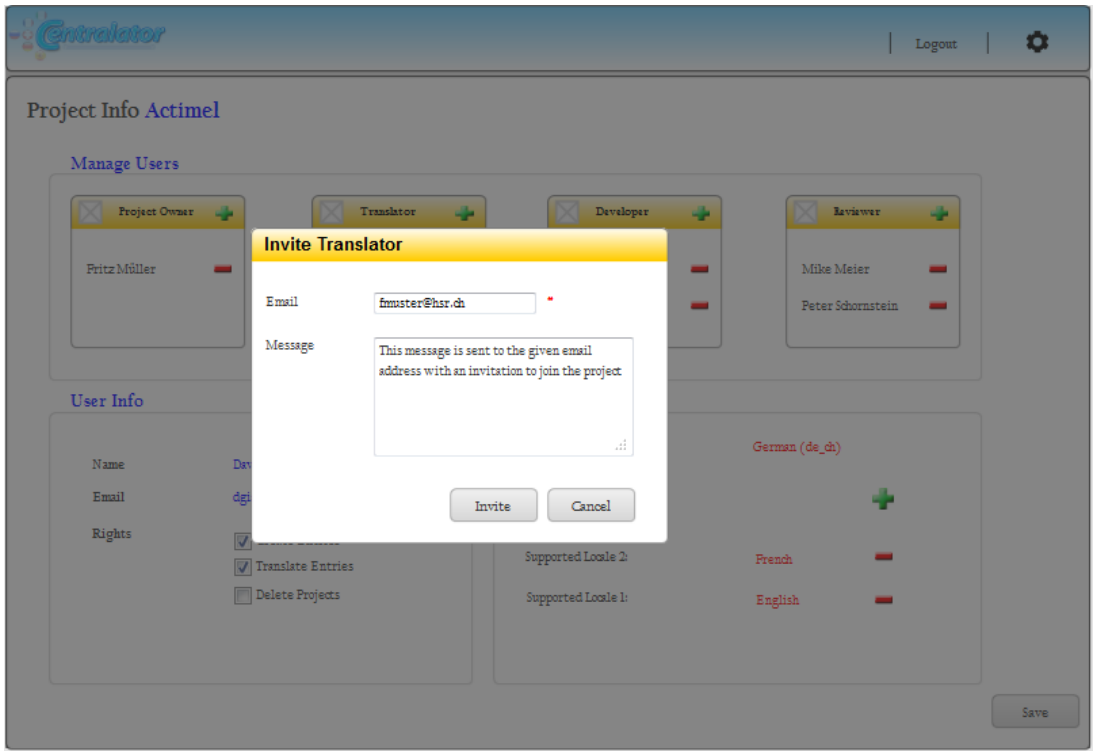


Abbildung 10: Paperprototyp Project Info – Invite User Dialog

2.1.1.4 PROJECT TRANSLATIONS

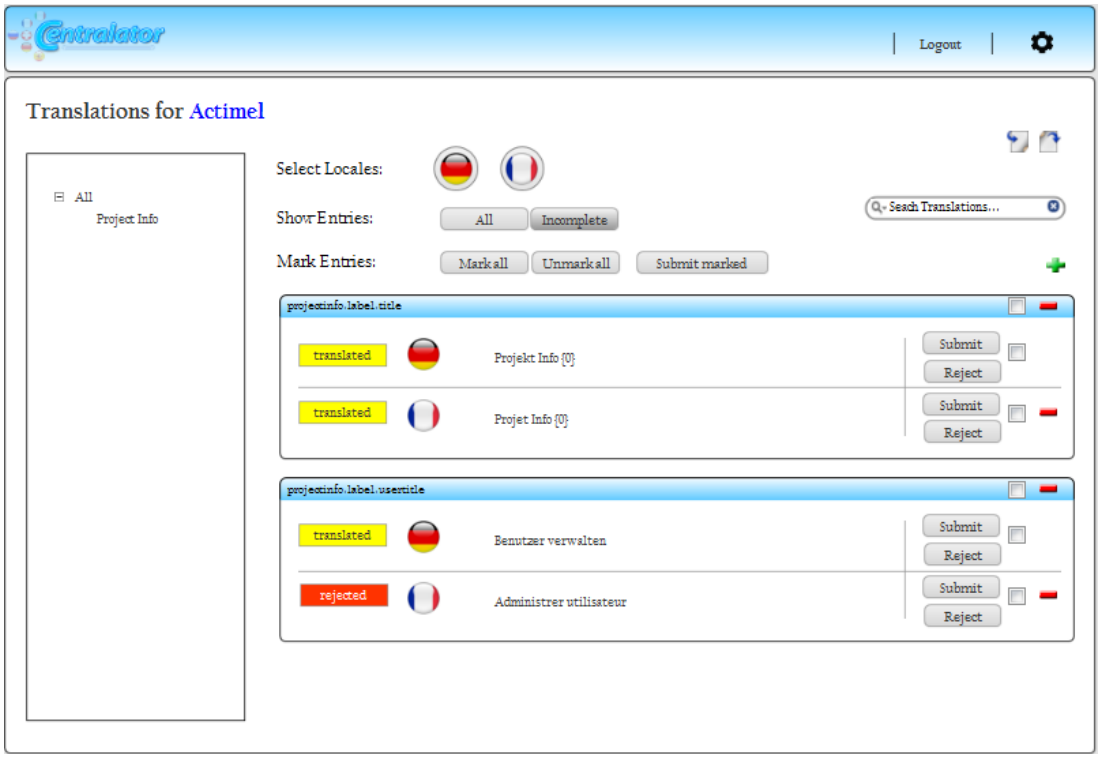


Abbildung 11: Paperprototyp Project Translations

2.1.2 EXTERNES DESIGN

Im Folgenden wird das GUI von Centralator beschrieben. Der **Logout** Link, welches ausser auf der Startseite auf allen anderen Seiten vorzufinden ist, wird aus Wiederholungsgründen nicht nochmals bei jeder dieser Seiten beschrieben. Das gleiche gilt für die Navigationslinks.

2.1.2.1 STARTPAGE

Auf der Startseite kann sich ein Benutzer anmelden oder registrieren. Über den **Login** Link wird der Login Dialog aufgerufen, womit sich ein Benutzer anmelden kann. Über den **Register** Link wird der Register Dialog aufgerufen, mit dem sich ein neuer Benutzer registrieren kann.

Abbildung 12: Startpage

Login Dialog

Ein Benutzer kann sich mit seiner **E-Mail Adresse** und einem **Passwort** anmelden. Der Dialog führt Validierungen für leere Felder, falsche oder nicht im System existierende E-Mail Adressen durch.

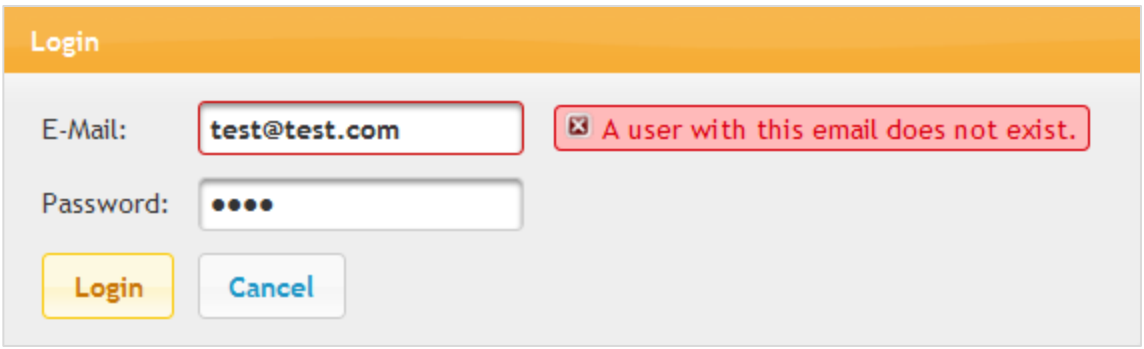


Abbildung 13: Startpage - Login Dialog

Register Dialog

Für die Registrierung muss ein neuer Benutzer die im Bild unten dargestellten Feldern ausfüllen.
Der Dialog führt Validierungen für leere Felder, falsche oder bereits im System existierende E-Mail-Adressen durch.

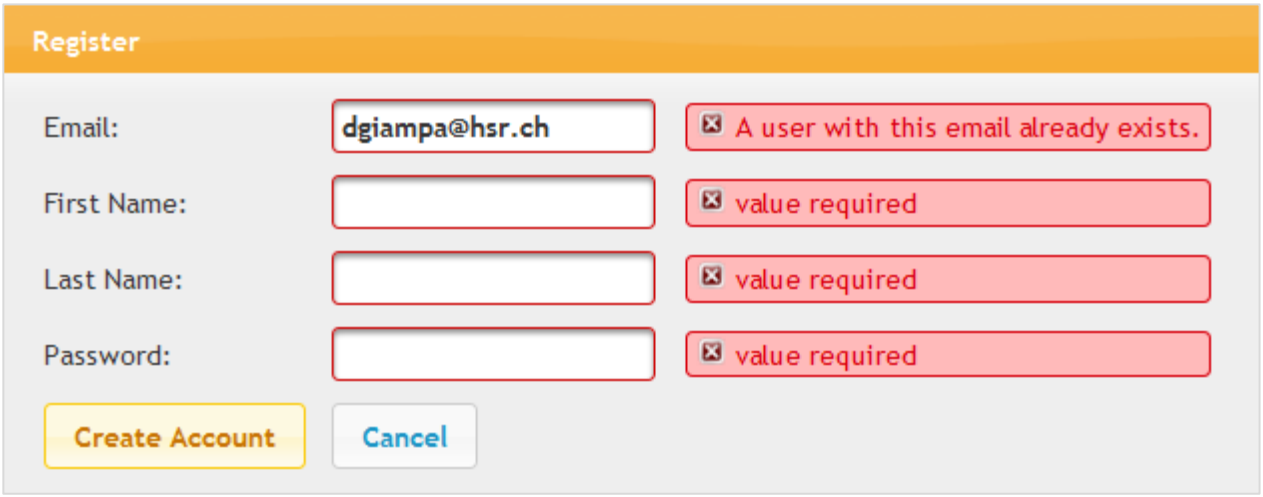


Abbildung 14: Startpage - Register Dialog

2.1.2.2 PROJECT DASHBOARD

Auf der Project Dashboard Seite werden alle Projekte aufgelistet, bei denen der angemeldete Benutzer Mitglied ist. Für jedes Projekt wird der **Name**, die aktuelle **Rolle** des angemeldeten Benutzers für dieses Projekt, die verfügbaren **Locales**, die **Anzahl übersetzten Keys** und mögliche **Aktionen** angezeigt.

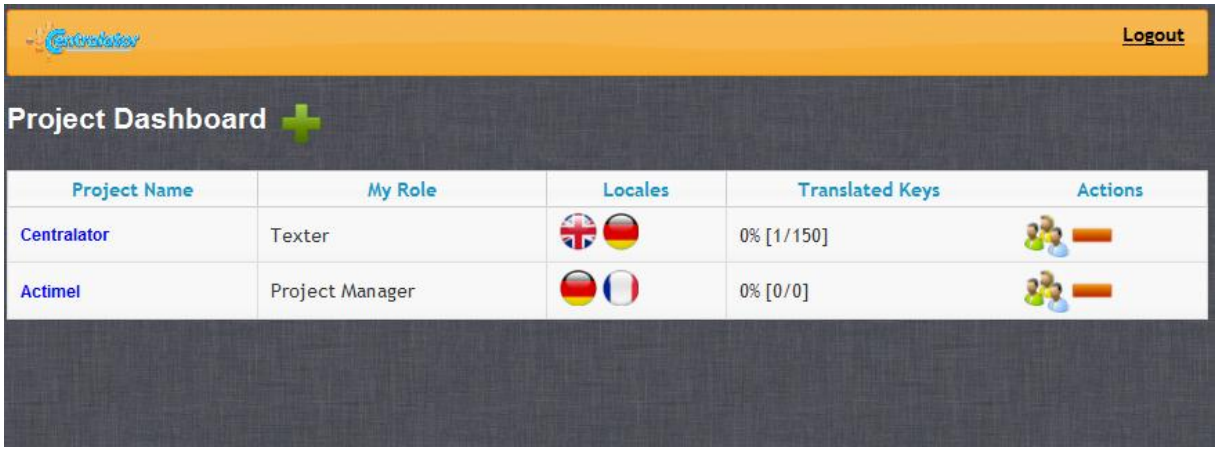


Abbildung 15: Project Dashboard

Tabelle 23: Project Dashboard Aktionen

Aktion	
	Projekt löschen.
	Auf die **Project Info** Seite für dieses Projekt wechseln.
	Add Project Dialog öffnen, um neue Projekte zu erstellen.

Add Project Dialog

Mit Hilfe des Add Project Dialogs können neue Projekte erstellt werden. Im Dialog muss als erstes ein Name eingegeben werden. Weiter muss die **Default-Locale** und eine erste **Supported Locale** aus dem Dropdown-Element selektiert werden. Für die Locales wird immer zuerst eine Beschreibung und am Ende der Länder-Code in Klammern gezeigt.

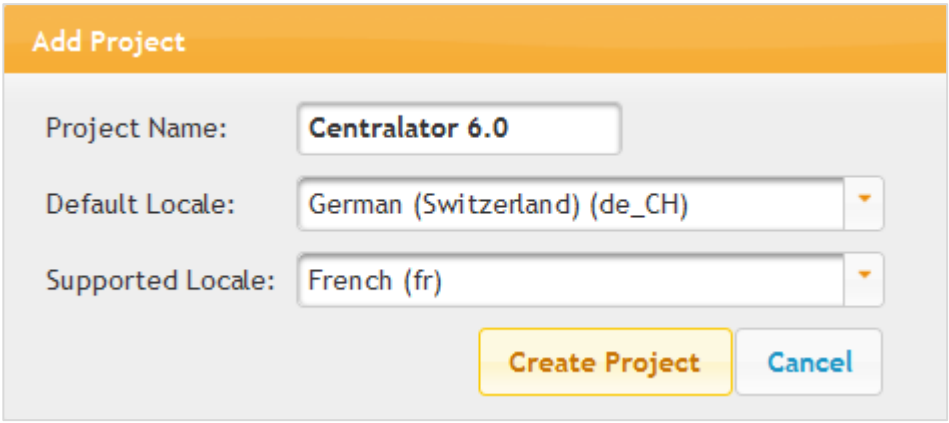


Abbildung 16: Project Dashboard - Add Project Dialog

2.1.2.3 PROJECT INFO

Auf der Project Info Seite können drei verschiedene Aktionen durchgeführt werden:

- Projektmitglieder einsehen, hinzufügen beziehungsweise löschen
- Locales einsehen, hinzufügen beziehungsweise löschen
- Informationen zum Rest-Service einsehen

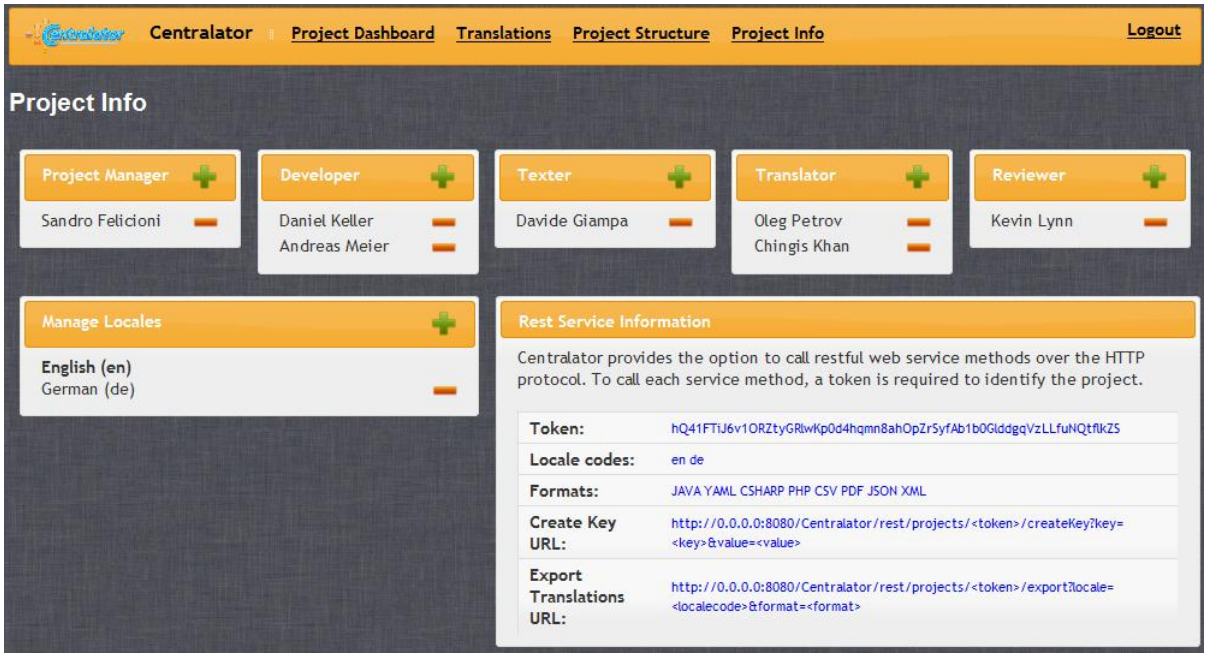


Abbildung 17: Project Info

Benutzer verwalten

Für jede Rolle ist eine Liste der zugehörigen Projektmitglieder dargestellt.

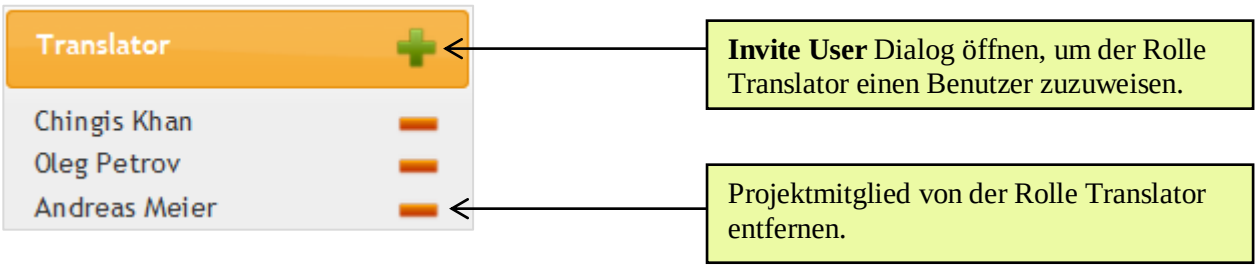


Abbildung 18: Project Info – Benutzer verwalten

Invite User Dialog

Um dem aktuellen Projekt einen Benutzer hinzuzufügen, muss eine **bestehende E-Mail Adresse** im Invite User Dialog eingegeben werden. Der Dialog führt Validierungen für leere Felder (ausser das Message-Feld, welches optional ist), nicht existierende oder bereits im Projekt genutzte E-Mail Adressen durch (siehe Bild). Nach einem erfolgreichen Einladen eines Benutzers müsste die Message über E-Mail an den Benutzer geschickt werden. Die E-Mail Benachrichtigung wurde aber aus zeitlichen Gründen nicht mehr implementiert.

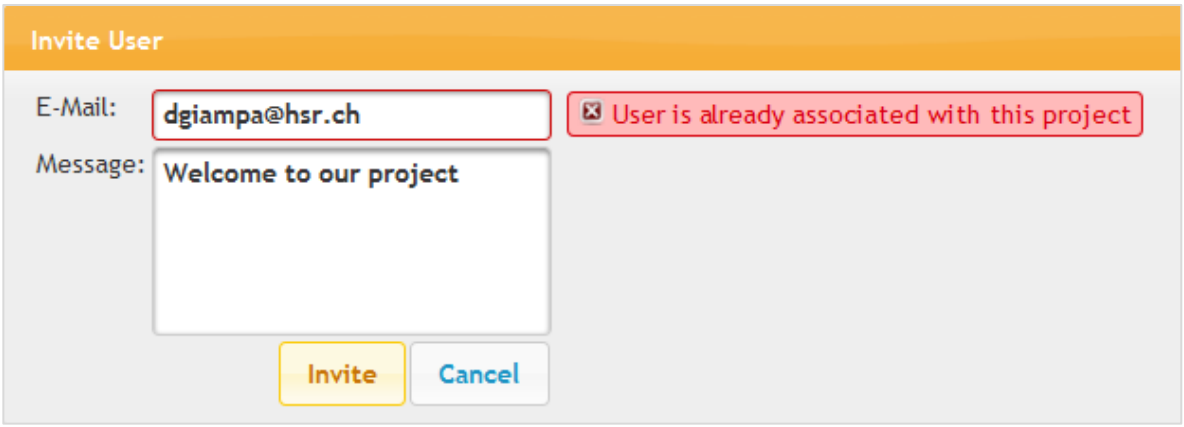


Abbildung 19: Project Info – Invite User Dialog

Locales verwalten

Für jede Rolle ist eine Liste der zugehörigen Projektmitglieder dargestellt.

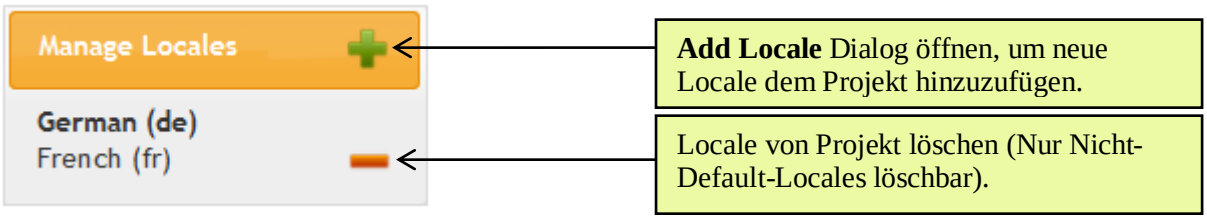


Abbildung 20: Project Info – Locales verwalten

Add Locale Dialog

Mit Hilfe des **Add Locale** Dialogs können dem aktuellen Projekt neue Locales hinzugefügt werden:

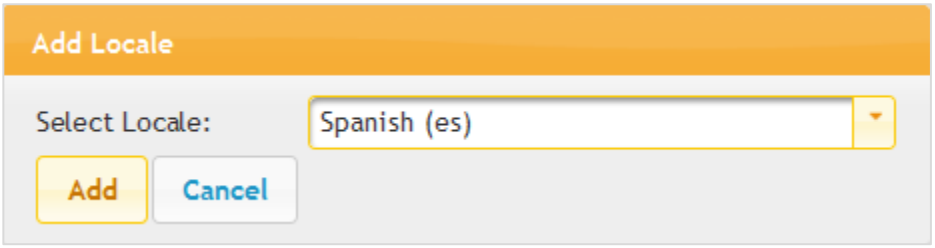


Abbildung 21: Project Info – Add Locale Dialog

Informationen zum REST-Service

Um REST Methoden auf Centralator aufzurufen, besitzt jedes Projekt einen Token, welches in der URI mitgegeben werden muss, um ein Projekt eindeutig zu identifizieren. Die **Rest Service Information** Sektion zeigt den **Token**, die verfügbaren **Locale Codes**, die **Formate** und die **URI' s**, um Keys zu erfassen oder Übersetzungen zu exportieren, an.

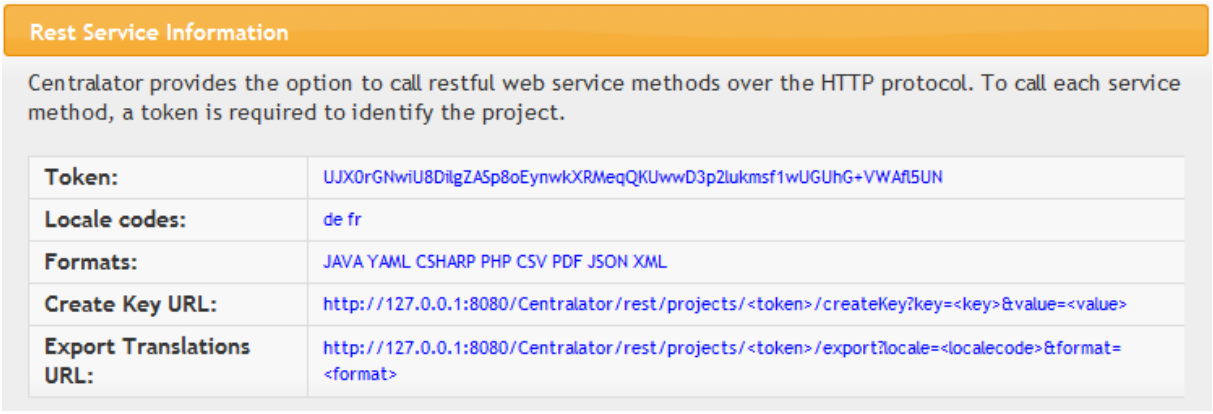


Abbildung 22: Project Info – REST Service Information

2.1.2.4 PROJECT STRUCTURE

Auf der Project Structure Seite können Keys zu Gruppen zusammengefasst werden, sodass eine Kontextbildung möglich ist. Auf der linken Seite ist der Gruppenbaum mit Projektnamen als Root-Gruppe dargestellt. Auf der rechten Seite sind die zum Projekt gehörenden Keys angezeigt.

Folgende Aktionen sind auf dieser Seite möglich:

- Gruppen erstellen und löschen
- Keys an Gruppen zuweisen
- Keys von Gruppen löschen
- Gruppen mitsamt Keys und Subgruppen exportieren

Über das **Kontextmenü**, welches beim Rechtsklick auf eine Gruppe oder Key erscheint, können die oben erwähnten Aktionen ausgeführt werden.

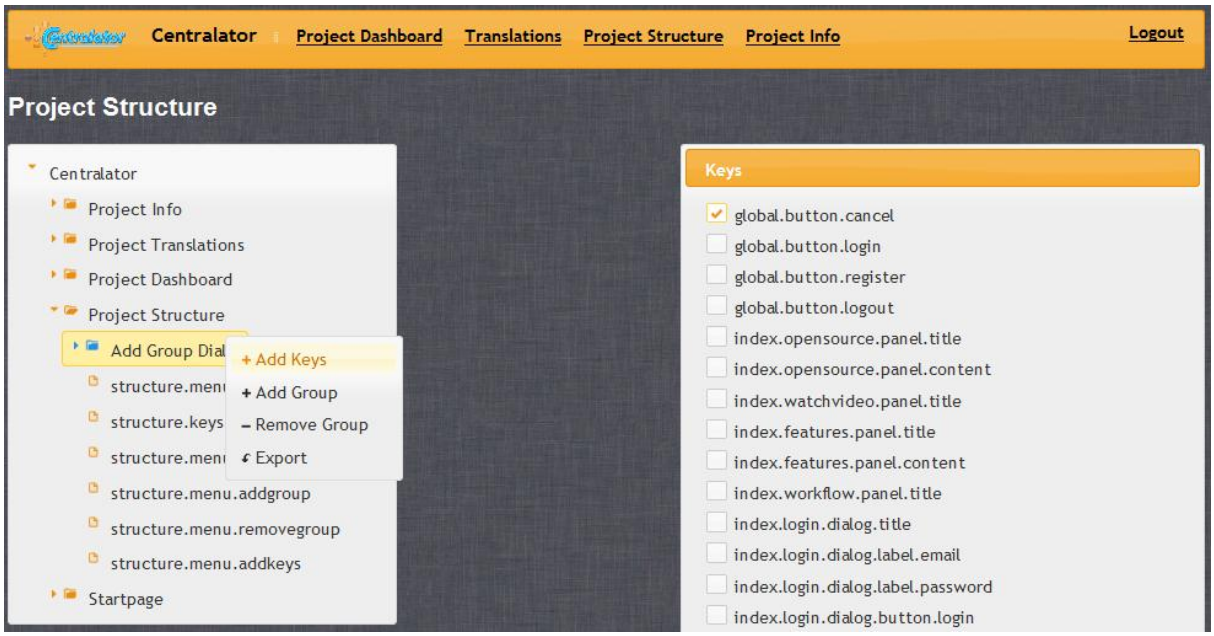


Abbildung 23: Project Structure

Kontextmenü für Root-Gruppe

Im Kontextmenü der Root-Gruppe kann mit **Add Group** der **Add Group** Dialog geöffnet werden. Mit **Export** wird der gesamte Baum als PDF-Datei exportiert.

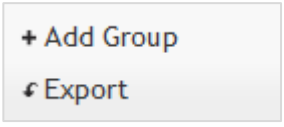


Abbildung 24: Project Structure – Kontextmenü Root-Gruppe

Kontextmenü für Gruppe / Subgruppe

Im Kontextmenü einer Subgruppe kann mit **Remove Group** die selektierte Gruppe gelöscht werden.
Um Keys mit **Add Keys** zur selektierten Gruppe hinzuzufügen, muss mindestens ein Key auf der rechten Seite selektiert werden.

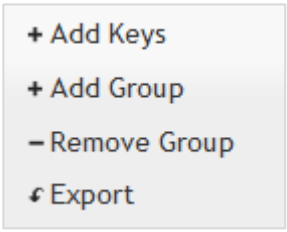


Abbildung 25: Project Structure – Kontextmenü für Gruppe / Subgruppe

Kontextmenü für hinzugefügten Key

Im Kontextmenü eines in einer Gruppe assoziierten Keys kann mit **Remove Key** der selektierte Key aus der nächst näheren Gruppe gelöscht werden.

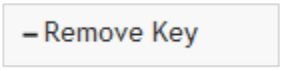


Abbildung 26: Project Structure – Kontextmenü für Key

Add Group Dialog

Mittels des **Add Group** Dialogs kann dem Projekt eine neue Gruppe hinzugefügt werden.
Der Dialog führt Validierungen für leere Felder (ausser das Description Feld, welches optional ist) durch.

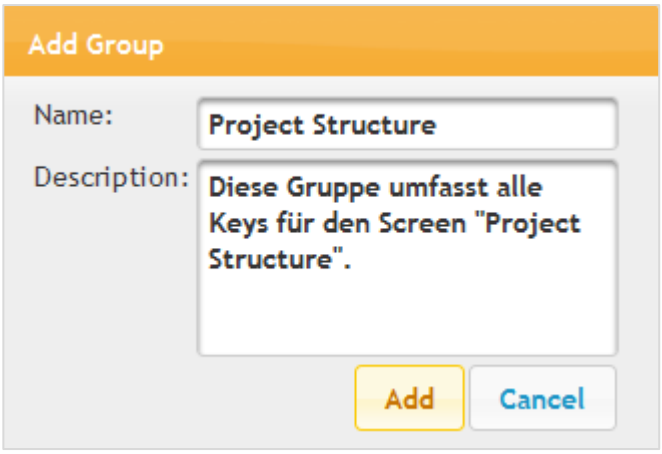


Abbildung 27: Project Structure – Add Group Dialog

2.1.2.5 PROJECT TRANSLATIONS

Die Project Translations Seite ist die wichtigste Seite von Centralator, weil hier der Übersetzungsworkflow abgebildet ist und die Übersetzungen gemacht werden.

Die Seite besteht aus drei Teilen:

- Filter-Sektion
- Struktur-Sektion: Logische und hierarchische Key-Filterung
- Translations-Sektion

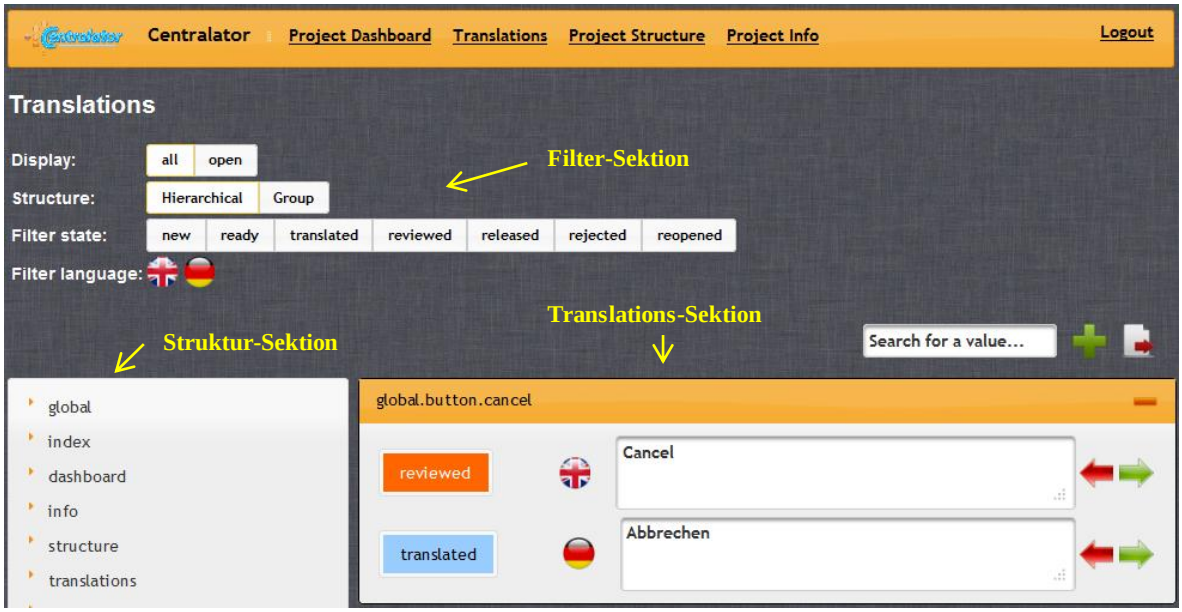


Abbildung 28: Project Translations

Filter-Sektion

Mit **all**/**open** können alle oder nur die für die Rolle offenen Übersetzungen angezeigt werden. Standardmässig wird nach offenen Übersetzungen gefiltert. Mit **Hierarchical**/**Group** kann entschieden werden, wie die Keys in der **Struktur**-Sektion dargestellt werden sollen, mit der die Keys dann gefiltert werden können. Jede Übersetzung kann zudem nach Status gefiltert werden. Es können gleichzeitig mehrere Stati ausgewählt sein. Weiter können Übersetzungen nach Sprachen gefiltert werden. Auch hier können nach allen oder nur nach einzelnen Sprachen gefiltert werden.

Abbildung 29: Project Translations – Filter-Sektion

Zu guter Letzt kann nach Texten gesucht werden. Die Filterung ist Case Sensitive.

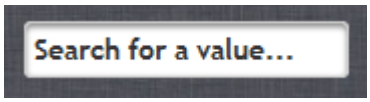


Abbildung 30: Project Translations – Text-Filter

Struktur-Sektion

In der Struktur-Sektion können die Keys sowohl hierarchisch als auch logisch dargestellt werden. Wählt man ein Sub-Key innerhalb der hierarchischen Struktur aus, so werden alle Keys nach diesem Sub-Key gefiltert. Wählt man hingegen eine Gruppe aus, so werden alle Keys nach dieser Gruppe gefiltert.

Abbildung 31: Project Translations - Hierarchische Gruppierung

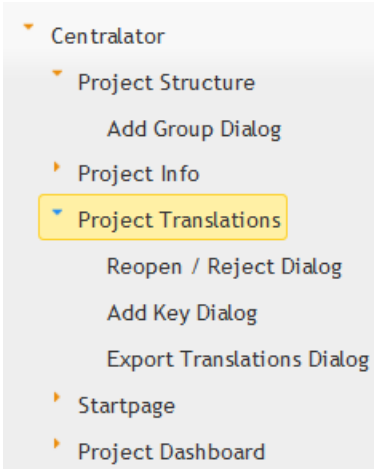


Abbildung 32: Project Translations - Logische Gruppierung

Translations-Sektion

Die Translations-Sektion zeigt die Übersetzungen eines Projekts in Blöcken an. In jedem Block wird zuerst der Key dargestellt, anschliessend für alle Sprachen einen Entry.

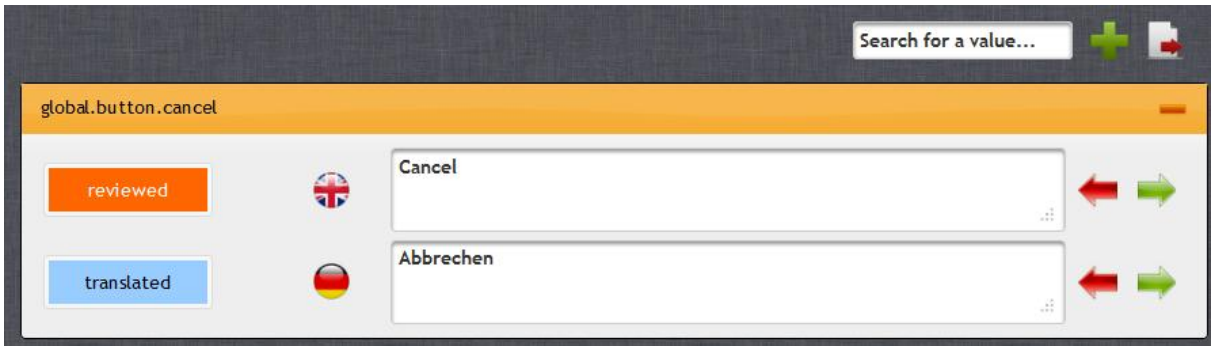


Abbildung 33: Project Translations – Translations Sektion

Die Symbole für diese Sektion sind hier erklärt:

Tabelle 24: Project Translations Aktionen

Aktion	
	**Add Key** Dialog öffnen, um einen neuen Key hinzuzufügen.
	**Export Translations** Dialog öffnen, um alle Übersetzungen zu exportieren.
	Key und alle zugehörigen Entries löschen.
	Kommentar-Icon. Beim Mouse-Over wird die Liste der Kommentare zum Entry angezeigt.
	Entry weiterleiten.
	Entry zurückweisen. **Reject** Dialog wird geöffnet.
	Entry wiedereröffnen. **Reopen** Dialog wird geöffnet.

Add Key Dialog

Mit dem **Add Key** Dialog kann ein neuer Key mit Defaulttext erfasst werden. Der Dialog führt Validierungen für leere oder bereits existierende Keys durch (siehe Bild).

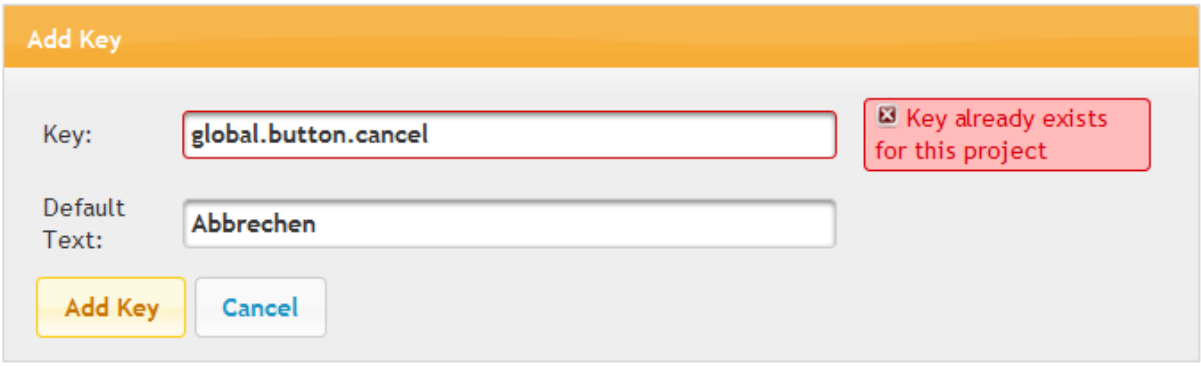


Abbildung 34: Project Translations – Add Key Dialog

Export Translations Dialog

Mittels des **Export Translations** Dialogs können alle Übersetzungen einer Locale eines Projekts in eines der Formate **Java Properties**, **YAML**, **PHP**, **RESX**, **XML**, **JSON**, **CSV** und **PDF** exportiert werden.

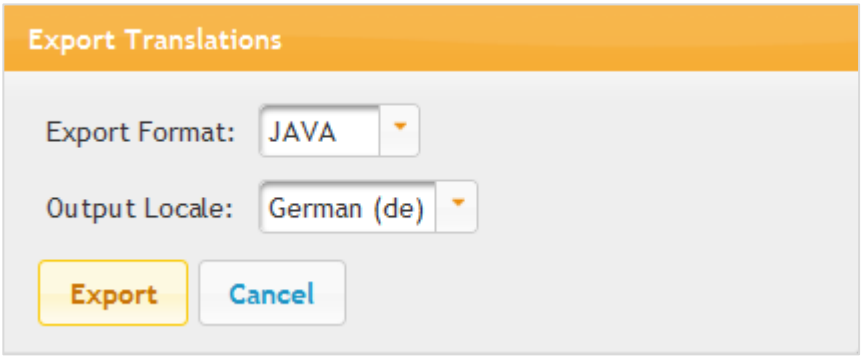


Abbildung 35: Project Translations – Export Translations Dialog

Reopen/Reject Dialog

Werden Entries wiedereröffnet bzw. zurückgewiesen, kann vor dem Zustandswechsel ein Kommentar erfasst werden.

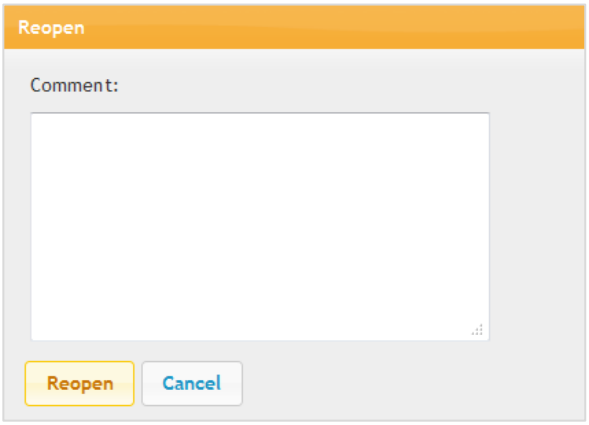


Abbildung 36: Project Translations – Reopen Dialog

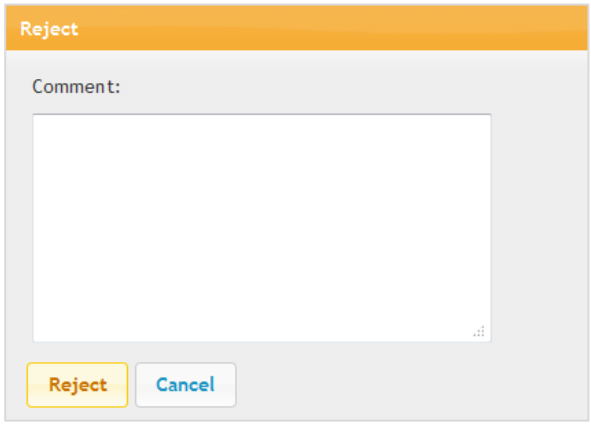


Abbildung 37: Project Translations – Reject Dialog

2.2 ARCHITEKTUR

2.2.1 ARCHITEKTURENTSCHEIDE

2.2.1.1 SVN

Der Source Code und die Dokumente wurden mit SVN 1.7.4 verwaltet. Der Hauptgrund wieso SVN und nicht ein anderes, moderneres VCS (Version Control System) wie zum Beispiel GIT verwendet wurde, ist wegen der merklich besseren Integration in die Eclipse-Entwicklungsumgebung.

2.2.1.2 MAVEN

Als Build-Tool wurde Maven 3 eingesetzt. Es übernimmt somit das Builden der Webapplikation zu einem WAR (Web Archive) File. Maven ist aber weit mehr als nur ein simples Build-Tool. Es wurde unter anderem für nachfolgende Aufgaben verwendet:

- **Dependency Resolution**
  Maven kann transitive und Projekt-Abhängigkeiten (Libraries), welche im POM File definiert sind, automatisch auflösen und runterladen (inklusive dessen Source Files). Dies hat den grossen Vorteil, dass die benötigten Libraries nicht ins SVN eingecheckt werden müssen.

- **Entwicklungsumgebung aufsetzen**
  Anstatt Eclipse spezifische Files wie .project, .classpath oder der ganze settings-Ordner ins SVN einzuchecken, können diese Files ebenfalls via Maven generiert werden. Dies verhindert eine direkte Abhängigkeit zu einem spezifischen Entwicklungseditor und ermöglicht, den Entwicklern den Editor ihrer Wahl zu verwenden. Für Eclipse wurde ein spezielles Maven Plugin konfiguriert, welches die nötigen Files und Projekt-Facetten generiert. Dadurch wird das Projekt automatisch als JSF 2.0 Projekt erkannt und fügt den gewünschten Tooling Support hinzu (z.B. Code Completion in den .xhtml Files). Wenn jedoch ein anderer Editor wie zum Beispiel IDEA von JetBrains verwendet werden will, kann das Projekt nach dem auschecken ohne weitere Einstellungen importiert werden. IDEA erkennt anhand des POM Files, dass es sich um ein Maven-Projekt handelt und generiert daraus automatisch alle benötigten Projekt-Dateien.

- **Tests ausführen**
  Maven arbeitet nach dem Prinzip "Convention over Configuration" und führt bei jedem Build automatisch alle JUnit Tests aus, welche im Projekt unter dem Verzeichnis "src/test/java" zu finden sind.

- **Reports generieren**
  Für Maven stehen zahlreiche Plugins zur Verfügung, welche den Code analysieren und daraus einen XML und HTML Report generieren. Diese Reports werden von Jenkins ausgelesen und in die Seite integriert.

2.2.1.3 JSF

Um das Web GUI umzusetzen, wurde JSF 2.0 und die Component Library Primefaces 3.2 [Url05] verwendet. Da schon von Anfang an klar war, dass es ein Rich GUI sein soll und über viel Ajax beinhalten würde, war JSF für dieses Projekt prädestiniert. Bei der Auswahl der Component Library standen Primefaces und RichFaces zur Diskussion. Insgesamt bietet RichFaces zwar mehr Funktionalität und ist umfangreicher, die Integration von Primefaces ist aber wesentlich einfacher und bietet ebenfalls eine beträchtliche Menge an Funktionalität.

2.2.1.4 SPRING

Das Spring Framework [Url06] bietet viele Features, welche in Module untergebracht sind. Dank dieser modularen Architektur können nur einzelne Module verwendet werden. Bei Bedarf lässt es sich jedoch einfach mit anderen Modulen erweitern. Zum Beispiel könnte mit wenig Aufwand noch Spring AOP [Url07] verwendet werden, was für Logging Zwecke eingesetzt werden könnte. Nachfolgende Features wurden von Spring verwendet:

- **Dependency Injection**
  Anstelle des DI (Dependency Injection) Frameworks, welches bereits standardmässig mit JSF mitgeliefert wird, wurde das DI Framework von Spring [Url08] verwendet. Es ist weitaus mächtiger im Vergleich zu demjenigen von JSF und erlaubt zum Beispiel Bean Hierarchien abzubilden, bietet Unterstützung für Factory und Singleton Beans, unterstützt sowohl Lazy als auch Eager Loading von Beans und bietet hervorragende Unterstützung zum Testen. Die Bean-Definitionen können wie in JSF via Annotations oder XML erfolgen. Die XML-Variante hat jedoch den Vorteil, dass keine direkte Abhängigkeit zu Spring entsteht und Spring jederzeit wieder entfernt werden kann.

- **Expression Language**
  Anstelle der EL (Expression Language) von JSF wurde diejenige von Spring verwendet [Url09]. Wenn die Bean-Definitionen via Spring erfolgen, ist die EL von Spring ebenfalls erforderlich. Die Syntax zur Verwendung der EL ändert sich dabei aber nicht. Die EL von Spring hat aber dennoch weitere Vorteile. Zum Beispiel bietet Spring Support um ein Property-File auszulesen und stellt diese Properties anschliessend als Variablen zur Verfügung. Diese Variablen können nun mit der EL ausgelesen werden und erlauben es so die Bean Definitionen im XML konfigurierbar zu machen.

- **Transaction Management**
  Spring wurde auch eingesetzt um die Aufgaben des Transaction Managements [Url10] zu übernehmen. Neben wenigen Zeilen XML Konfiguration genügt es anschliessend Service Klassen mit @Transactional zu annotieren, damit alle Methoden an Spring Transactions teilnehmen. Anstelle der Annotation wäre auch eine komplette XML-Variante möglich gewesen, bedingt aber wesentlich mehr Aufwand, deshalb wurde darauf verzichtet. Ein grosser Vorteil von Spring Transactions ist, dass die Transaction beim Methodenaufruf automatisch begonnen und nach Ende der Methode automatisch commited wird. Wenn während der Methode eine RuntimeException geworfen wird, übernimmt Spring automatisch den Rollback. Um all dies zu ermöglichen, generiert Spring on-the-fly einen Proxy für die eigentliche Service Klasse. Zu beachten gilt, dass nun bei Bean-Zugriffen mit dem Proxy gearbeitet wird und nicht mit der eigentlichen Service-Klasse. Solange man aber mit den Interfaces arbeitet und keine Casts zu der konkreten Klasse versucht, gibt es keine Probleme.

- **Unit & Integration Testing**
  Wenn der Web Server gestartet wird, wird automatisch der Spring Kontext aufgebaut, welcher alle Bean-Definitionen enthält. Spring erlaubt jedoch auch diesen Kontext programmatisch zu erzeugen, was vor allem beim Testen Anwendung fand. Alles was dazu notwendig war, war ein XML File anzugeben, welches die Bean-Definitionen beinhaltet. Auf diese Weise konnte auch im Testumfeld mit den realen Klassen gearbeitet werden und man musste nicht auf die Dependency Injection verzichten.
  Bei Integration Tests mit der Datenbank hat man oft das Problem, dass nach dem Test die Datenbank nicht mehr in ihrem originalen Zustand ist, da neue Zeilen hinzugefügt oder gelöscht wurden. Aber auch für dieses Problem hat Spring eine Lösung. Spring stellt speziell für diesen Zweck eigene Test Runner Klassen für JUnit zur Verfügung. Diese stellen zum einen weitere JDBC Methoden zur Verfügung, um mit einer separaten Datenbank-Verbindung native SQL-Abfragen ausführen zu können, welche in den Assert Statements verwendet werden können. Zum anderen macht Spring nach Ablauf der Testmethode automatisch alle Datenbankänderungen wieder rückgängig, welche durch Spring Transactions ausgelöst wurden.

Nachfolgend ein Diagramm von den verschiedenen Modulen, die Spring anbietet. Während dem Projekt wurden alle Module des Core Container's und die Module Web, ORM, Transactions und Test verwendet.

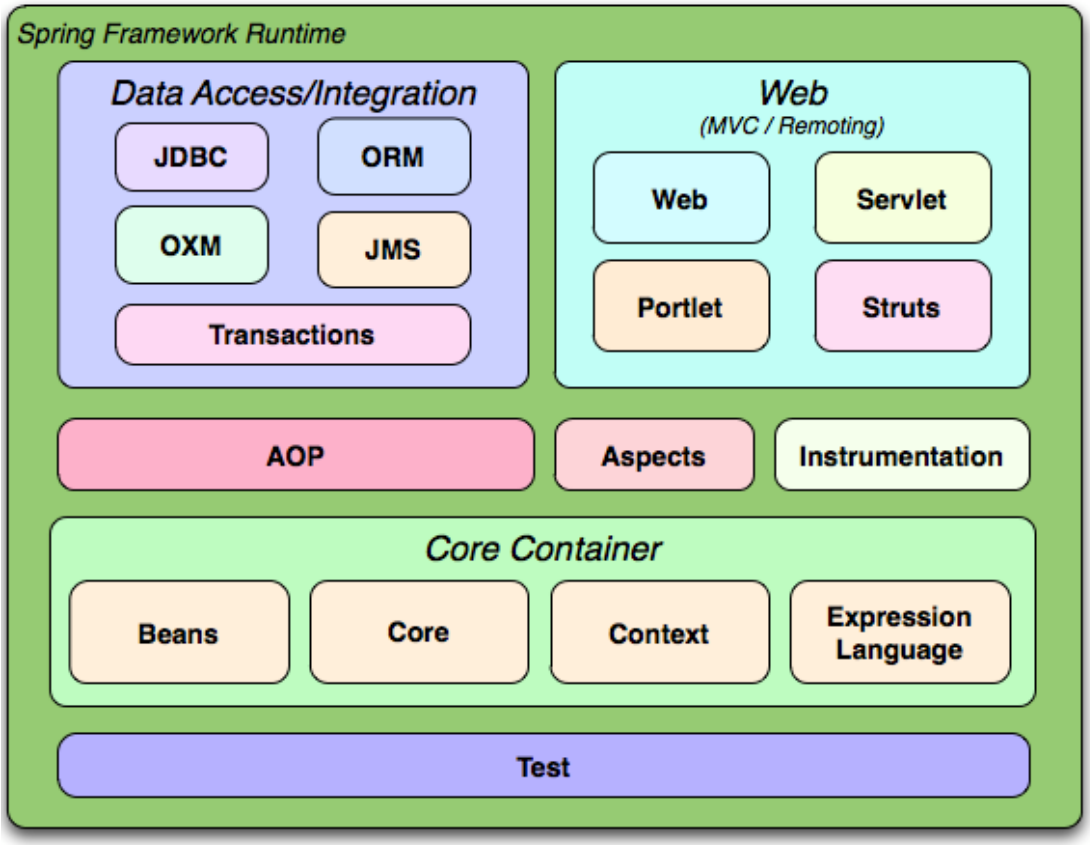


Abbildung 38: Spring Modul Diagramm

2.2.1.5 HIBERNATE

Als O/R (Objekt/Relationaler) Mapper wurde Hibernate [Url11] verwendet. Dies ermöglichte es eine relationale Datenbank in einer objektorientierten Applikation zu verwenden, ohne sich manuell um das Mapping kümmern zu müssen. Die Entitäten wurden via Annotations markiert und Abfragen erfolgten durch HQL (Hibernate Query Language) Queries. Das Laden und Speichern der Entitäten übernahm dabei vollständig Hibernate. Dadurch konnten sehr schnell Fortschritte erzielt werden. Es erlaubte zudem mehr Zeit in neue Features und die Business-Logik zu investieren.

2.2.1.6 MYSQL

Als Datenbankserver wurde Mysql ausgewählt, weil die Installation trivial ist und eine gute Verwaltungssoftware, die **Mysql Workbench** [Url12], angeboten wird. Da der virtuelle Server auf Windows läuft, hat sich der Mysql Installer für diese Plattform gut geeignet. Oracle war auch in Frage gekommen, jedoch bietet es zu viele Funktionalitäten an, welche für diese Studienarbeit überflüssig gewesen wären. Für Oracle gäbe es eine abgespeckte Version (Oracle Express Edition), jedoch benötigt man dafür einen Oracle-Account, was bei Mysql nicht der Fall ist. Mysql ist leichtgewichtiger und reicht für diese Arbeit völlig aus.

2.2.1.7 JUNIT

Um die Stabilität des Codes zu gewährleisten, wurden sowohl für Unit- als auch für Integration-Tests JUnit eingesetzt. Anstelle von JUnit hätte jedoch auch ein anderes Test-Framework wie zum Beispiel TestNG verwendet werden können. JUnit ist jedoch besser in andere Frameworks wie zum Beispiel Spring integriert und deshalb einfacher zu nutzen.

2.2.1.8 SLF4J

Als Logging-Framework wurde SLF4J (Simple Logging Facade for Java) [Url13] verwendet. SLF4J dient dabei lediglich als Facade [Gamma95]. Das darunterliegende Logging-Framework ist Logback. Ausschlaggebend für den Einsatz dieses Logging-Frameworks ist folgendes Grundproblem. Third-Part-Libraries verwenden unter Umständen andere Logging-Frameworks als die eigene Applikation und erwarten dessen Logging-Library im Klassenpfad zu finden. Spring benötigt zum Beispiel JCL (Jakarta Commons Logging) im Klassenpfad. Je mehr Third-Part-Libraries man verwendet, desto grösser wird auch die Wahrscheinlichkeit, dass mehrere unterschiedliche Logging-Frameworks verwendet werden müssen. Um dennoch nur ein Logging-Framework zu verwenden, bietet SLF4J Bridge-Libraries für fast alle anderen Logging-Frameworks an. Nachfolgendes Bild illustriert, wie SLF4J diese Situation löst.

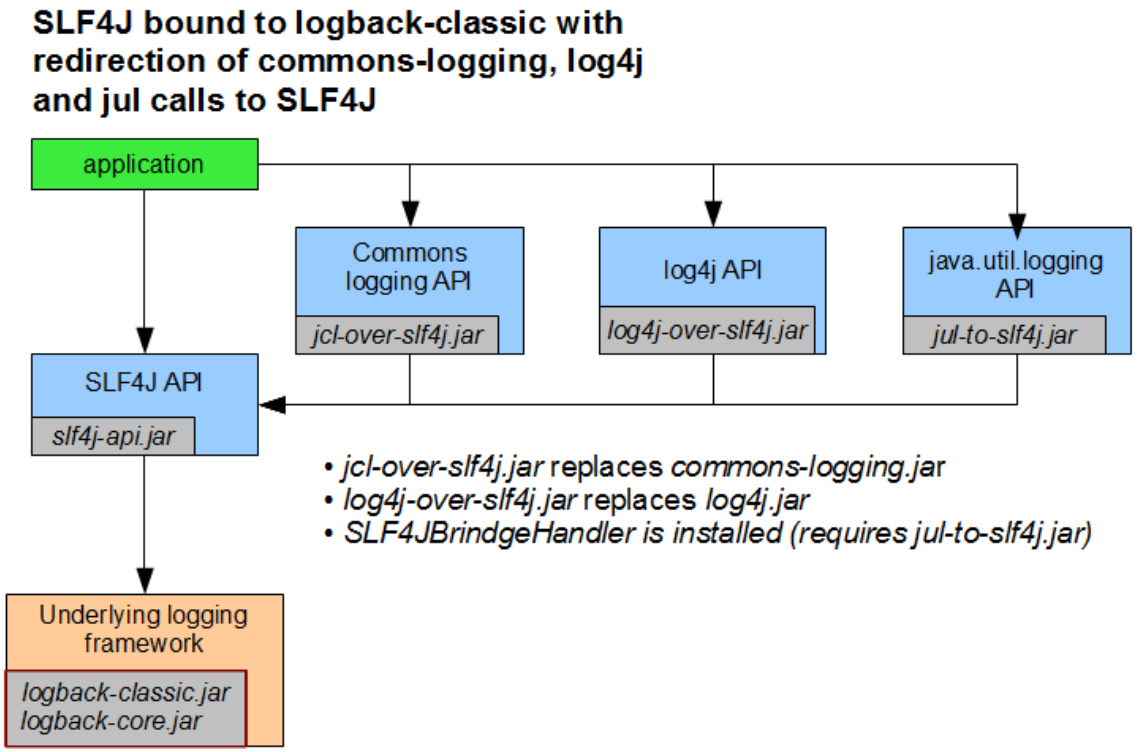


Abbildung 39: SLF4J's Lösungsansatz für mehrere Logging Frameworks

2.2.2 LOGISCHE ARCHITEKTUR

Anbei die logische Gliederung der Packages von Centralator. Die Packages **presentation** und **persistence** dienen nur zur Vervollständigung, da diese keine selbsterstellten Java-Packages sind.

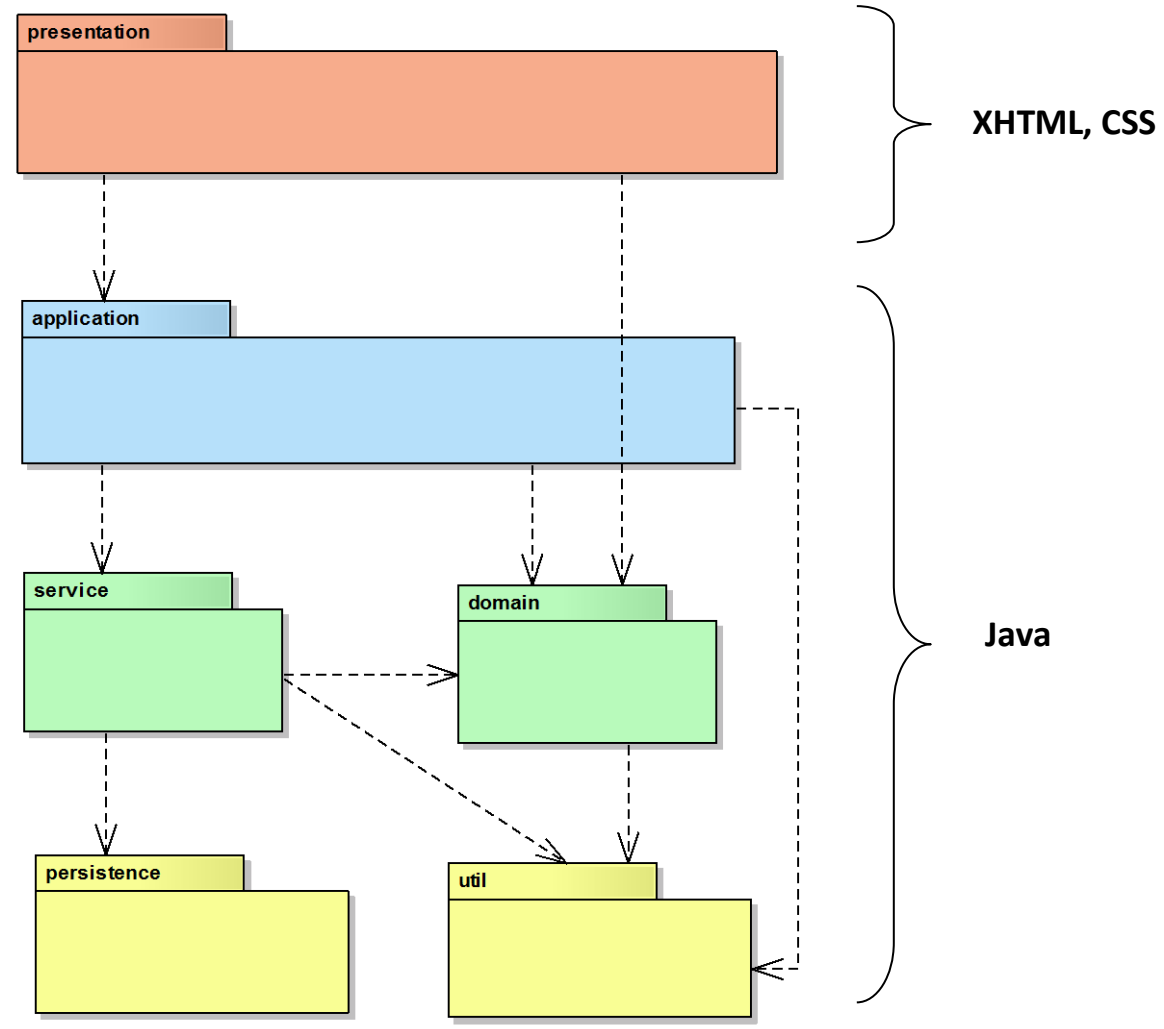


Abbildung 40: Abstraktes Package - Diagramm

Tabelle 25: Logische Architektur: Abstrakte Packages

Package	Beschreibung
presentation	Beinhaltet XHTML-Dateien sowie Ressourcen wie Bilder, Styles, Templates.
application	Beinhaltet Controller für die verschiedenen Pages sowie Exportformate.
service	Liest Daten aus der Datenbank oder schreibt in diese.
domain	Beinhaltet die Entitäten die persistiert werden sowie die Rechte der Rollen.
persistence	Hibernate O/R-Mapper. Zuständig für Entitäten-Persistierung und Session-Management.
util	Utility-Klassen für die gesamte Applikation.

2.2.2.1 Reines Package-Diagramm

Anbei ist das Java-Package-Diagramm von Centralator dargestellt. Die Packages **startpage**, **projectdashboard**, **projectinfo**, **projectstructure** und **translations** beinhalten die Controller- und Modelklassen für die entsprechenden Webseiten.

Das package **rest** beinhaltet den **RestController**, der Rest-Requests entgegennimmt und verarbeitet.

Das Package **commons** stellt gemeinsame Klassen für das **application**-Package zur Verfügung. Ein Beispiel dazu sind die Format-Klassen, welche für den Export von Übersetzungen in das jeweilige Format zuständig sind.

Das **impl**-Package im **service**-Package beinhaltet die Implementierung der Service-Interfaces. Es gibt insgesamt fünf Services: **EntryService**, **GroupService**, **KeyService**, **UserService** und **ProjectLocaleService**.

Im Subpackage **rights** des **domain**-Packages werden die Rechte für jede Rolle definiert, um den Workflow korrekt umzusetzen.

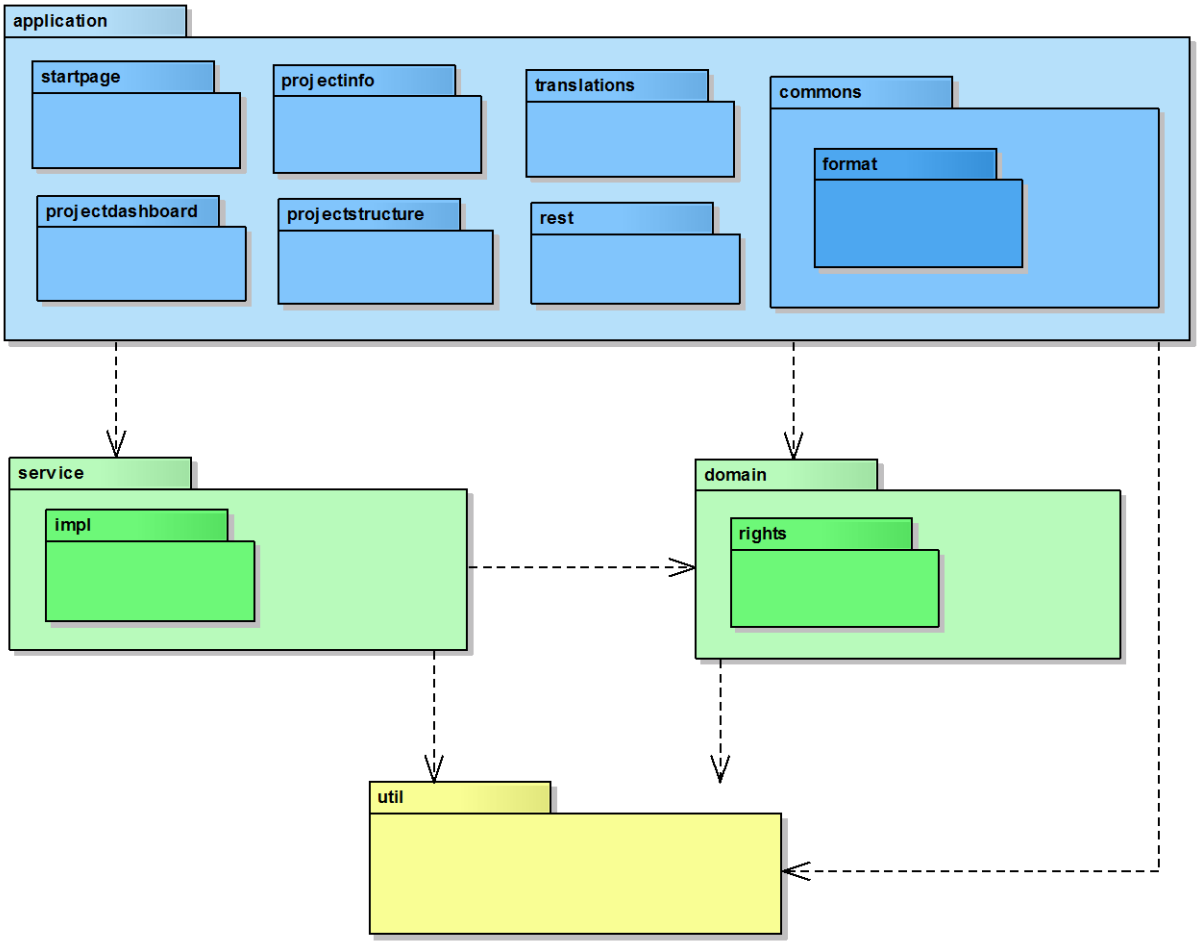


Abbildung 41: Reines Package - Diagramm

2.2.2.2 KLASSENINTERAKTION

Im Folgenden soll eine Interaktion am Beispiel des **Project Dashboards** gezeigt werden. Wenn ein Benutzer die Seite **dash.xhtml** öffnet, lädt der **ProjectDashboardController** die entsprechenden Daten über den Service-Layer (Persistenz ist im Bild absichtlich nicht angezeigt). Die Benutzereingaben auf dieser Seite werden in das **ProjectDashboardModel** gespeichert, welche später vom Controller verarbeitet werden.

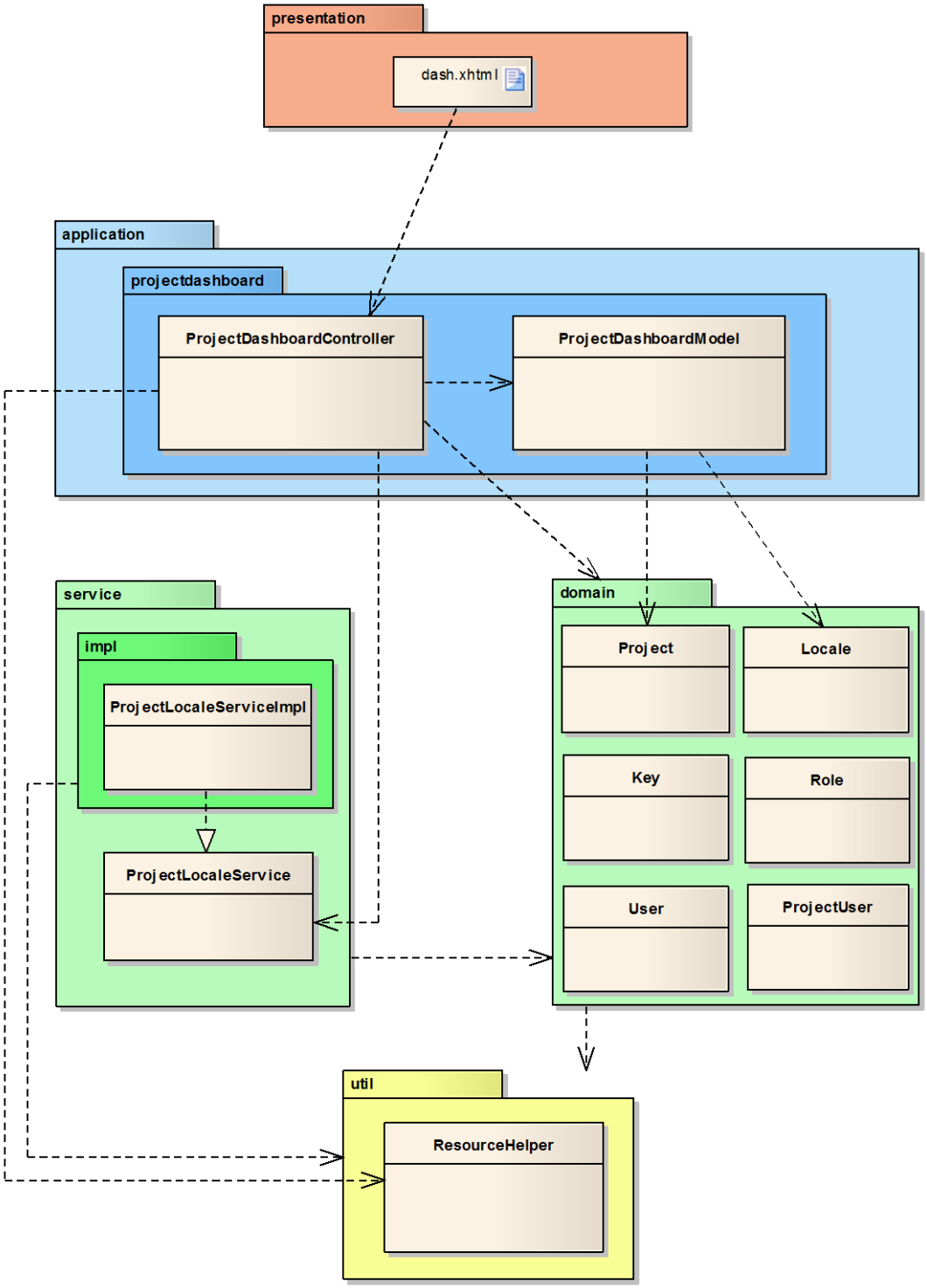


Abbildung 42: Klasseninteraktion von Project Dashboard

2.2.3 INTERAKTIONSDIAGRAMM KEY ERFASSEN

Das unten abgebildete Sequenzdiagramm beschreibt, wie ein Developer einen Key auf der Translations-Seite erfassen kann.

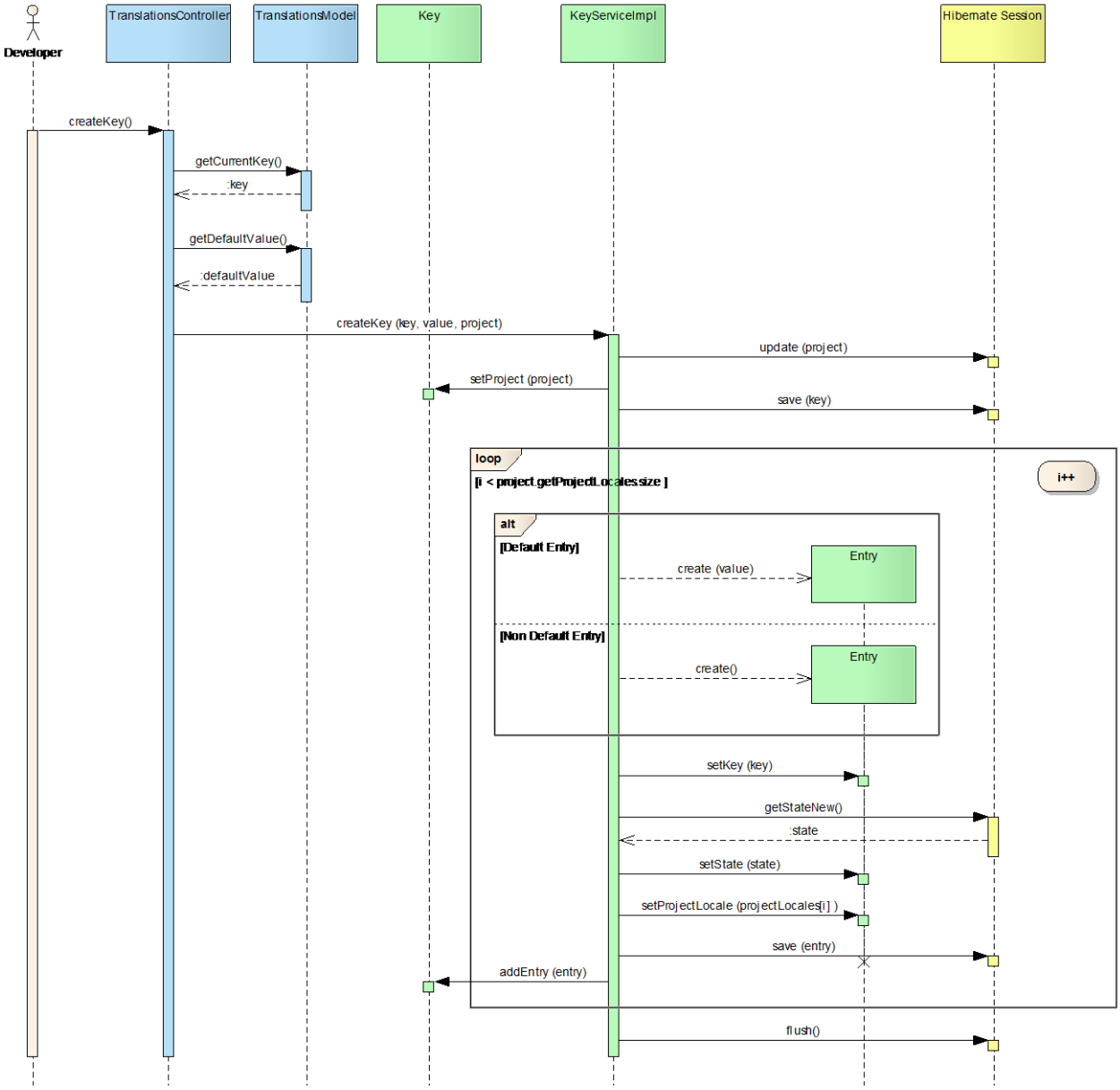


Abbildung 43: Interaktionsdiagramm Key erfassen

2.2.4 PHYSISCHE ARCHITEKTUR

Die Webapplikation wird als Webarchiv (.war-Artifakt) auf einem Servlet Container wie z.B. Tomcat deployed. Diese greift über Hibernate auf die Mysql-Datenbank zu. Ein Client kann auf zwei Arten auf die Applikation zugreifen: Entweder mittels Browser auf das Web-GUI oder mittels REST-Anfragen auf die REST-Schnittstelle.

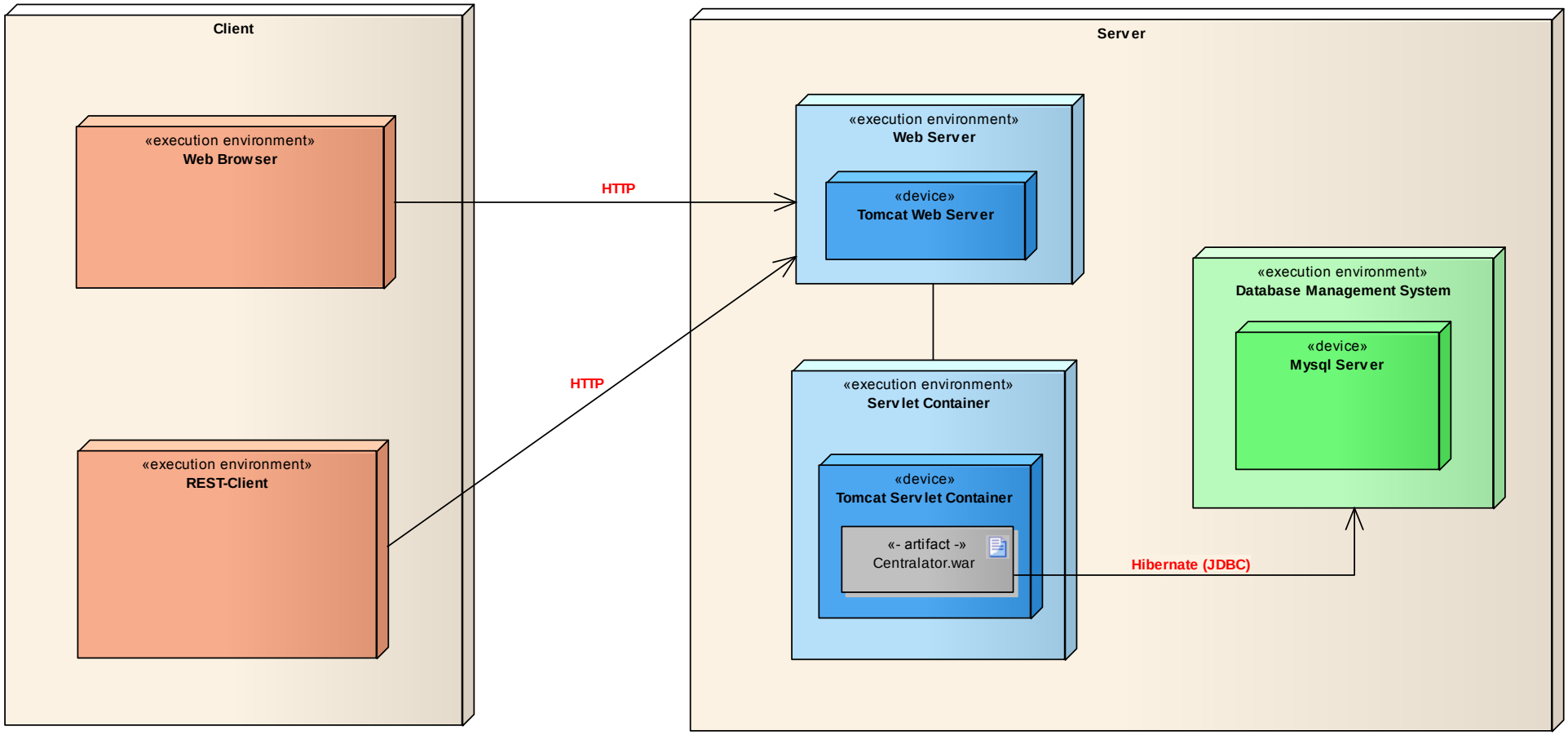


Abbildung 44: Physische Architektur

2.2.5 ARCHITEKTURKONZEPTE

Nachfolgend werden die wichtigsten Konzepte und Klassen näher beschrieben.

2.2.5.1 RIGHTS

Die abstrakte Klasse Rights stellt das zentrale Rechtesystem innerhalb der Applikation dar und ist eine Implementierung des Strategy Patterns [Gamma95]. Für jede Rolle gibt es eine konkrete Implementierung welche entscheidet, über was für Rechte diese Rolle verfügt. Nachfolgendes Klassendiagramm illustriert die Situation und veranschaulicht die verfügbaren Rechte.

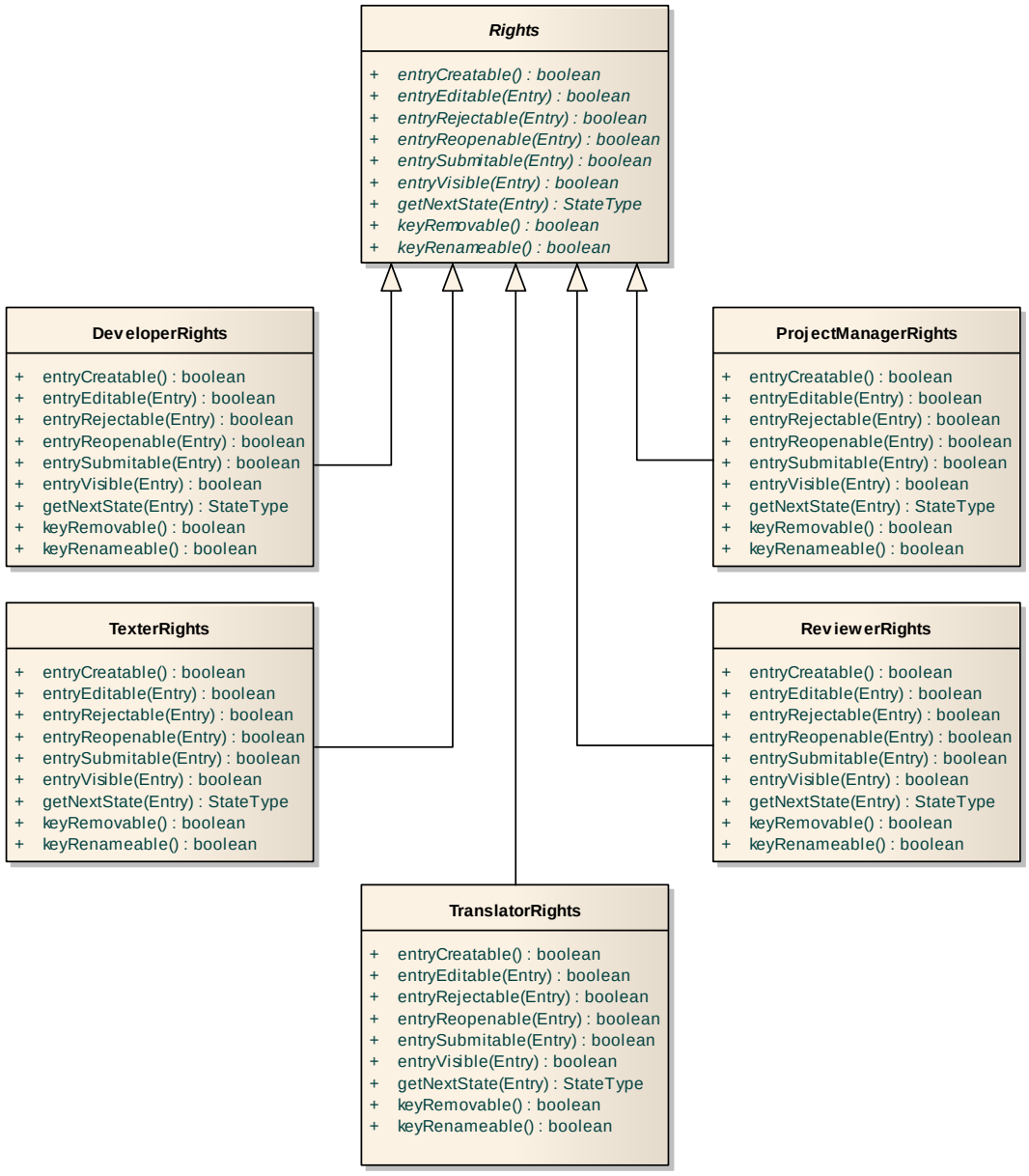


Abbildung 45: Klassendiagramm Rights

Jede Rolle verfügt über ein eigenes Rights Objekt und alle Entscheidungen ob die Rolle für etwas berechtigt ist oder nicht, finden ausschliesslich aufgrund dieser Rechten statt. Dies hat den Vorteil, dass Änderungen für eine Rolle zentral vorgenommen werden können und garantiert keine Seiteneffekte für andere Rollen zur Folge haben. Des Weiteren ist es so möglich, ohne grossen Aufwand eine bestehende Rolle zu löschen oder eine neue Rolle hinzuzufügen.

2.2.5.2 FILTER

Um Keys, Texte und Übersetzungen dynamisch zu filtern, wurde das Decorator Pattern [Gamma95] eingesetzt. Der abstrakten Klasse Rights wurde die Methode entryVisible() hinzugefügt, welche zu einem Boolean evaluiert und darüber entscheidet ob ein Entry sichtbar ist oder nicht. Es ist zu beachten, dass auch wenn ein Entry sichtbar ist, es nicht automatisch bedeutet, dass der Entry auch bearbeitbar ist. Es wurde speziell darauf geachtet, dass diese beiden Entscheidungen getrennt wurden. Die abstrakte Klasse AbstractFilterDecorator ist die Basisklasse aller Filter und delegiert standardmässig alle Methoden bis auf entryVisible() direkt an die konkrete Rights Klasse weiter.

Nachfolgende Filter Klassen wurden erstellt:

- **AllVisibleFilter**
  Zeigt jeden Entry an.

- **TextFilter**
  Zeigt nur diejenigen Entries an, welche im Text beziehungsweise in der Übersetzung ein gewisses Keyword enthalten.

- **LocaleFilter**
  Zeigt nur diejenigen Entries an, welche in einer gewissen Locale erfasst sind.

- **StateFilter**
  Zeigt nur diejenigen Entries an, welche in einem gewissen State sind.

- **KeyFilter**
  Zeigt nur diejenigen Entries an, welche einem gewissen Key oder Sub-Key davon angehören.

- **GroupFilter**
  Zeigt nur diejenigen Entries an, welche einer gewissen Gruppe angehören.

Nachfolgendes Klassendiagramm zeigt den AbstractFilterDecorator mit allen konkreten Filter-Implementierungen.

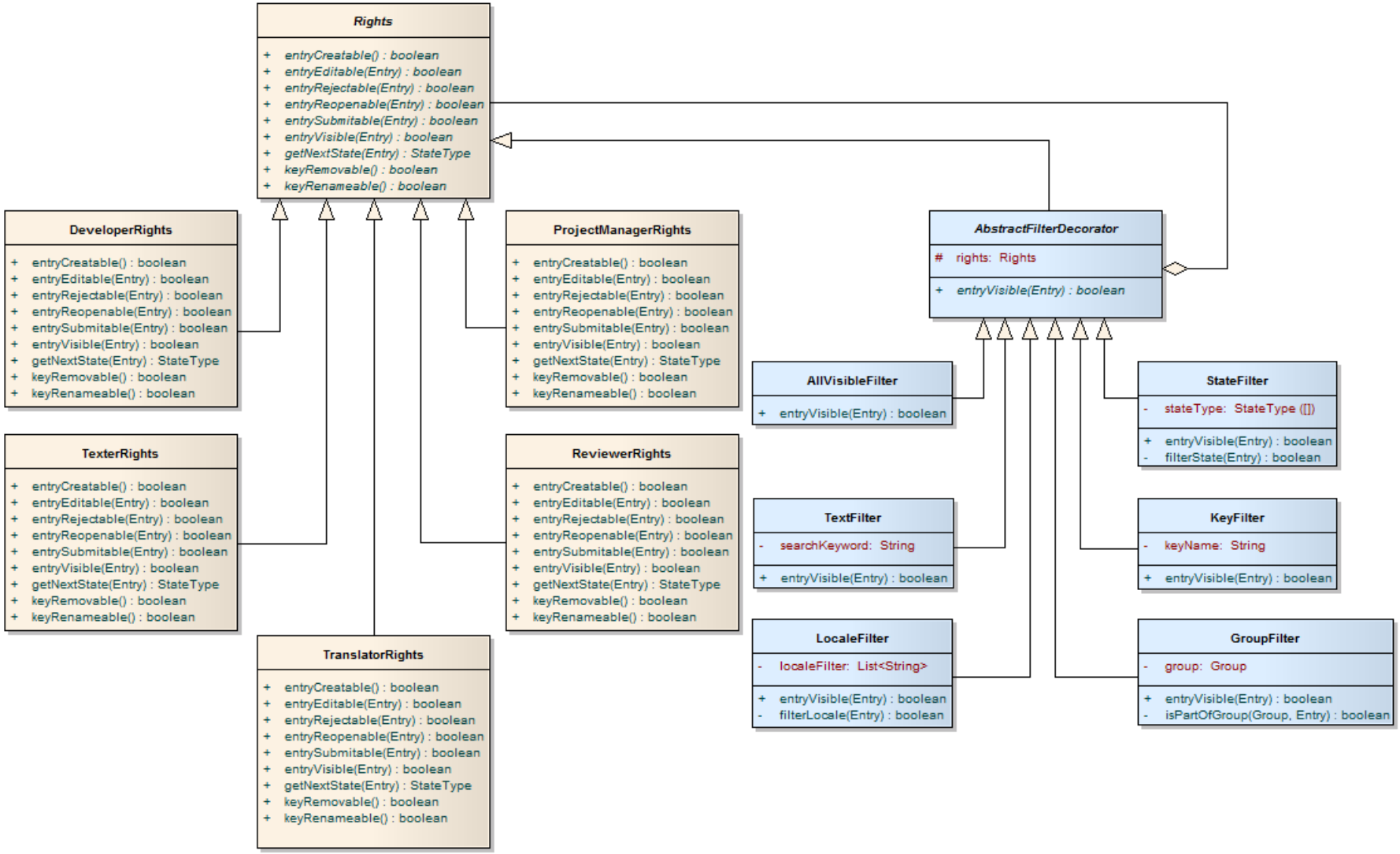


Abbildung 46: Klassendiagramm Filter Decorator

2.2.5.3 EXPORTFORMATE

Das Exportieren von Übersetzungen in ein gewünschtes Format wurde mit einer Strategy-Implementierung [Gamma95] gelöst. Die abstrakte Klasse **Format** definiert Basisattribute und Operationen für alle Formate wie z.B. das Setzen des Content-Types oder des Character-Encoding bei der HTTP-Response, welche dem Benutzer übermittelt wird.
Nicht jedem Format kann ein Printwriter übergeben werden, um in die Antwort zu schreiben, denn nicht bei allen Export-Formaten wird Text verwendet. Beim PDF-Format, welches mit Hilfe der **iText**-Bibliothek implementiert ist, muss zwingend ein Outputstream übergeben werden. Dies ist der Grund, weshalb die Basisklasse die beiden Methoden **writeTo(Outputstream)** und **writeTo(PrintWriter)** zur Verfügung stellt. Mittels **getFilePrefix()** und **getFileExtension()** kann der Name und die Endung der Exportdatei für jedes Format separat definiert werden. Für Textformate, die ein Trennzeichen zwischen Key und Value verwenden, kann mit Hilfe der Methode **getSeparator()** das gewünschte Trennzeichen angegeben werden. Beispielsweise wird bei Java das Gleichheitszeichen und bei YAML der Doppelpunkt als Trennzeichen zurückgegeben. Um die Handhabung der einzelnen Formate für die Präsentationsschicht und REST-Schnittstelle zu erleichtern, wurde die Enum-Klasse **FormatType** eingeführt.

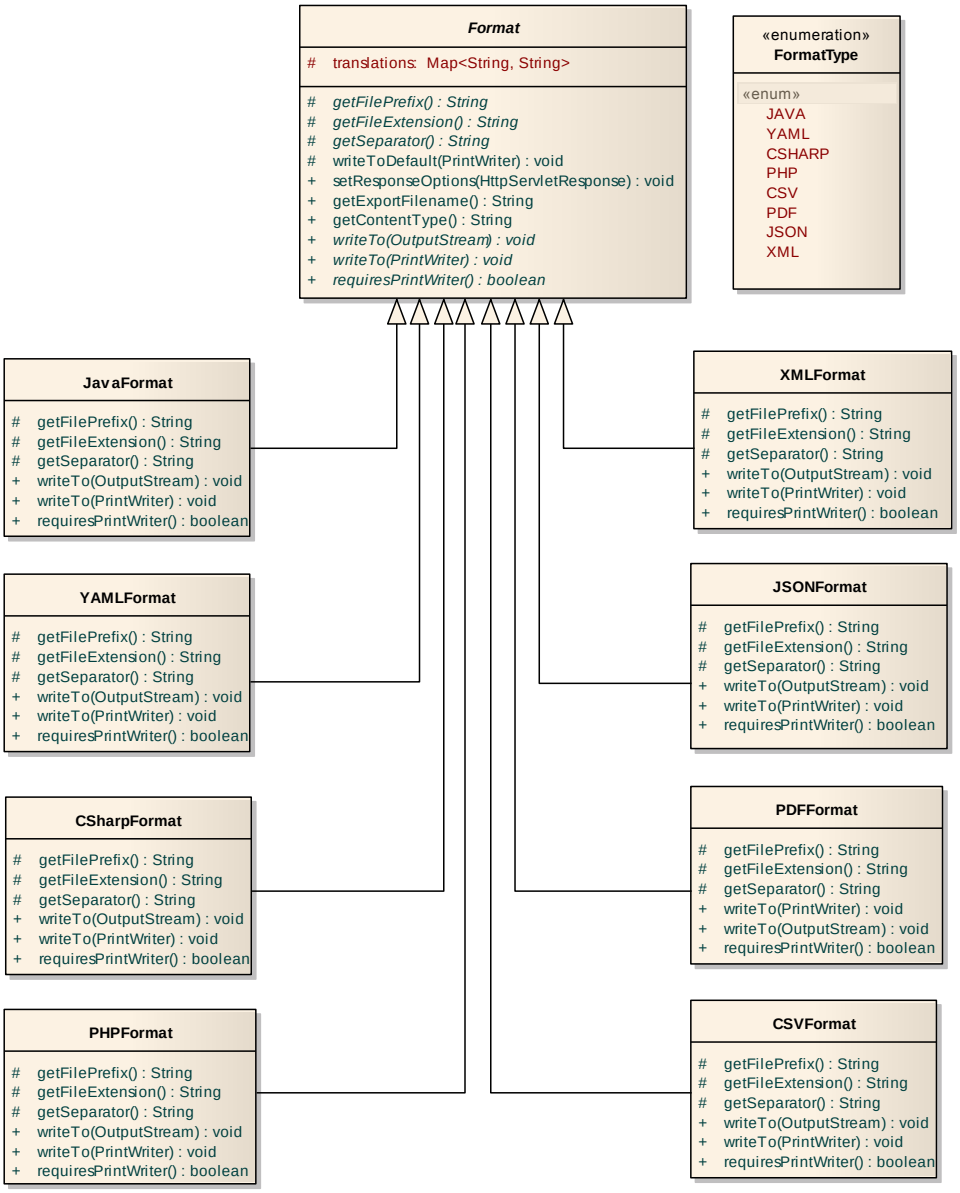


Abbildung 47: Klassendiagramm Formatklassen

2.2.6 ANBINDUNG AN REST-SCHNITTSTELLE

Um das Anbinden der Webapplikation an andere Dienste zu ermöglichen, wurde eine einfache REST-Schnittstelle entwickelt. Für Java-Webapplikationen kann die JAX-RS (Java API for RESTful Web Services) Spezifikation verwendet werden. Die Referenzimplementation davon nennt sich Jersey [Url14] und liegt als Open Source Projekt vor. Jersey erlaubt normale Java Klassen zu annotieren und sie so als REST-Services zur Verfügung zu stellen. Da diese annotierten Klassen unter einem anderen Servlet-Kontext laufen, lassen sie sich bei dessen Instanzierung nicht injecten. Dies hat zur Folge, dass standardmässig nicht auf die Spring-Beans, wie zum Beispiel die Service-Klassen, zugegriffen werden kann. Um dennoch davon Gebrauch zu machen, wird die Library jersey-spring.jar benötigt. Nachfolgender Code-Auszug zeigt eine vereinfachte Variante, wie der REST-Service implementiert wurde.

```
@Path("")
public class RestController {

  @GET
  @Produces("text/plain")
  @Path("/projects/{token}/createKey")
  public Response createKey(@PathParam("token") String token,
                            @QueryParam("key") String key,
                            @QueryParam("value") String value) {

    // validation omitted for clarity

    try {
      Project project = projectLocaleService.loadProject(token);
      keyService.createKey(key, value, project);
      return Response.ok("ok").build();
    } catch (BackendException e) {
      throw new NotFoundException("invalid values");
    }
  }
}
```

Abbildung 48: Beispiel der REST-Service Implementation

2.2.6.1 JAX-RS ANNOTATIONS

Nachfolgend ist ein Auszug der wichtigsten JAX-RS Annotations [Url15], welche für die REST-Service Schnittstelle verwendet wurden:

Tabelle 26: JAX-RS Annotations

Annotation	Beschreibung
@Path	Der Wert von @Path ist die relative URI, unter welcher die Java Klasse beziehungsweise Methode abrufbar ist.
@Get / @Post	Markiert eine Methode, um HTTP GET beziehungsweise HTTP POST Methoden entgegenzunehmen.
@PathParam	Extrahiert den Parameter von @PathParam aus der URI und stellt ihn der darauffolgenden deklarierten Variable zur Verfügung.
@QueryParam	Extrahiert den Parameter von @QueryParam aus den Request Query Parametern und stellt ihn der darauffolgenden deklarierten Variable zur Verfügung.
@Produces	Spezifiziert den MIME-Type den diese Methode als Antwort liefert.

2.2.6.2 REST API

Nachfolgend ist eine Beschreibung der kompletten REST API aufgelistet, welche die Webapplikation anbietet.

Create Key REST-Service:

Tabelle 27: Create Key REST-Service Parameter

Parameter	Optional	Beispiel	Bedeutung
token	Nein	3Ab3O49uV	Der Token ist ein einmaliger alphanumerischer Wert, welcher das Projekt identifiziert und so eine Authentisierung überflüssig macht.
key	Nein	user.title	Ein beliebiger String der den einzufügenden Key repräsentiert.
value	Nein	Anrede	Ein beliebiger String der den einzufügenden Defaulttext repräsentiert.

Aufruf Template:

- ../Centralator/rest/projects/<**token**>/createKey?key=<**key**>&value=<**value**>

Aufruf Beispiel:

- ../Centralator/rest/projects/**3Ab3O49uV**/createKey?key=**user.title**&value=**Anrede**

Export Keys REST-Service:

Tabelle 28: Export Keys REST-Service Parameter

Parameter	Optional	Beispiel	Bedeutung
token	Nein	3Ab3O49uV	Der Token ist ein einmaliger alphanumerischer Wert, welcher das Projekt identifiziert und so eine Authentisierung überflüssig macht.
locale	Nein	de_CH	Ein String der den Locale Code (ISO Code) repräsentiert, in welcher die Keys exportiert werden sollen.
format	Nein	XML	Ein String der das Export-Format angibt. Mögliche Werte sind JAVA, YAML, CSHARP, PHP, CSV, PDF, JSON und XML.

Aufruf Template:

- ../Centralator/rest/projects/<**token**>/export?locale=<**locale**>&format=<**format**>

Aufruf Beispiel:

- ../Centralator/rest/projects/**3Ab3O49uV**/export?locale=**de_CH**&format=**XML**

2.2.7 PERSISTENZ

2.2.7.1 DATENMODELL

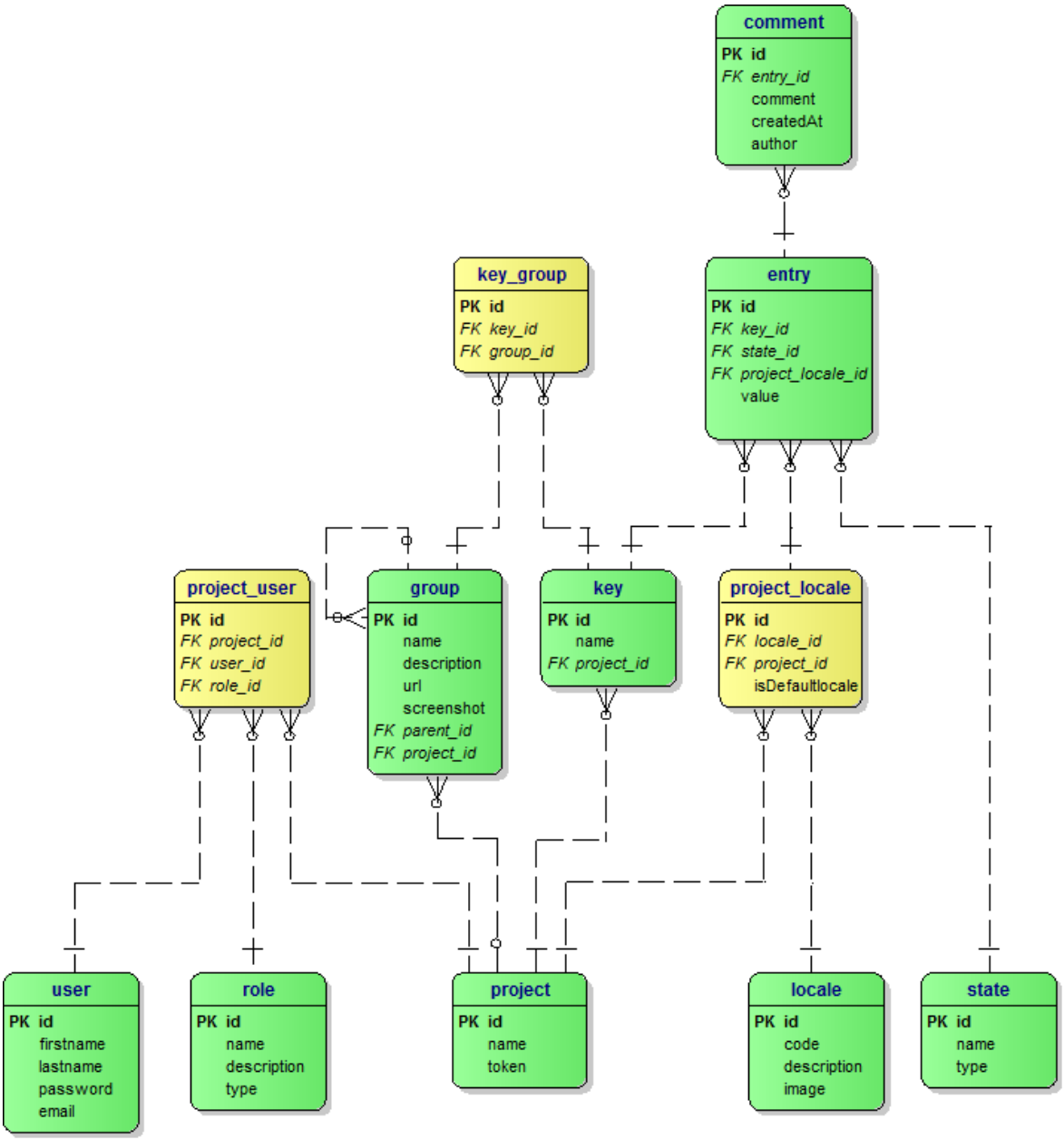


Abbildung 49: Datenmodell

2.2.7.2 BESCHREIBUNG DER WICHTIGSTEN ENTITÄTEN

Tabelle 29: Beschreibung der wichtigsten Entitäten und Beziehungen

Entität	Beschreibung
entry	Ein Entry ist ein Übersetzungseintrag, der von einem Benutzer bearbeitet wird und die zentrale Einheit des Workflows darstellt. Er gehört immer zu einem **Key** und befindet sich zu jedem Zeitpunkt in einem gewissen **Zustand**. Als Übersetzungsentität muss er immer einer **Locale** angehören. Mit dem **value**-Attribut können die eigentlichen Übersetzungen eines Keys gespeichert werden. Entries können weitergeleitet, zurückgewiesen oder wiedereröffnet werden. Für die letzten beiden Aktionen macht es Sinn, **Kommentare** zu erfassen, was mit der Entität **comment** möglich ist.
key	Ein Key ist die Identifikation einer GUI-Komponente, unterliegt meist einer hierarchischen Ordnung (z.B. global.button.cancel) und wird in verschiedene **Locales** übersetzt. Darum besitzt ein Key einen **Default-Entry** für die **Default-Locale** und beliebig viele Entries für weitere Locales. Ein Key gehört immer zu einem **Projekt** und ist für dieses eindeutig. Werden Keys zu **Gruppen** zusammengefasst, so hat ein Key beliebig viele Beziehung zu einer Gruppe.
group	Eine Gruppe ist dann nützlich, wenn mehrere **Keys** zusammengefasst sein sollten, um eine Kontextbildung zu ermöglichen (Beispiel Project Dashboard Screen). Eine Gruppe verfügt neben einem **Namen,** eine **Beschreibung**, eine **URL** und einen **Screenshot**. Sie ist in einem **Projekt** eindeutig identifizierbar und kann selber weitere **Gruppen** mit Keys beinhalten, was die Beziehung mit sich selber erklärt.
project	Ein Projekt stellt die oberste Verwaltungseinheit der Applikation dar. Es verfügt neben einem **Namen** auch noch über einen **Token**, der das Projekt identifiziert und mit dem der Zugriff auf die REST-Schnittstelle geregelt wird. Für jedes Projekt können verschiedene **Locales** definiert werden, eine davon muss die **Default-Locale** sein. Ein Projekt kann verschiedene **Keys**, **Gruppen** und **Benutzer** verwalten und Benutzer können an **Rollen** zugewiesen werden.
user	Die Entität **user** beschreibt einen Benutzer mit **Name**, **E-Mail Adresse** und **Passwort**. Für die Anmeldung eines Benutzers werden E-Mail Adresse und Passwort benötigt. Ein User kann für ein **Projekt** immer nur eine **Rolle** einnehmen.
role	Eine Rolle kann an verschiedene Benutzer in einem Projekt zugewiesen werden. Pro Projekt werden fünf Rollen definiert: **DEVELOPER** Erfasst neue Keys **TEXTER** Korrigiert Default-Entries **TRANSLATOR** Übersetzt und korrigiert Non-Default-Entries **REVIEWER** Korrigiert und weist Entries zurück **PROJECT MANAGER** Gibt Entries frei, weist diese zurück / eröffnet diese neu
locale	Die Entität **locale** beschreibt die Zielsprache, in die ein Entry übersetzt werden soll. Eine Locale besteht aus einem **Code** (z.B. de_CH), einer **Beschreibung** (German Switzerland) und einem **Bild** (z.B. german.png), das in den Projektressourcen von Centralator zu finden ist. Die Liste aller Locales, welche in die Datenbank gespeichert werden, wurde aus der Java Methode **Locale.getAvailableLocales()** entnommen, welche 152 Locales liefert. Ein Projekt verfügt über mindestens eine Locale, nämlich die Default-Locale.

state

Die Entität **state** beschreibt den **Zustand** eines Entry. Für den Workflow von Centralator wurden sieben Zustände definiert:

NEW	Entry wurde erstellt, da Developer einen neuen Key erfasst hat
READY	Entry wurde von Texter korrigiert und weitergeleitet
TRANSLATED	Non-Default-Entry wurde von Translator übersetzt
REVIEWED	Entry wurde von Reviewer korrigiert
RELEASED	Entry wurde von Project Manager freigegeben
REJECTED	Entry wurde von Project Manager oder Reviewer zurückgewiesen
REOPENED	Entry wurde von Project Manager wiedereröffnet

3 TESTING

3.1 SYSTEMTESTS

Die Systemtests wurden so durchgeführt, dass pro Webseite eine Testspezifikation sowie ein Testprotokoll erstellt wurden.

3.1.1 VORAUSSETZUNGEN

Folgende Voraussetzungen müssen erfüllt sein, damit die Systemtests für Centralator erfolgreich durchgeführt werden können:

- Eine Verbindung zur Centralator-Datenbank muss zur Verfügung stehen
- Die Centralator-Datenbank muss mit Daten (User, Locales, Projekte, Entries, Keys, Gruppen) abgefüllt sein
- Die Applikation muss auf einem Servlet-Container deployed sein

3.1.2 BEMERKUNGEN

- Bei Dialogen wird angenommen, dass nach Eingabe der Testdaten die zugehörige Primary Action (z.B. Login) ausgeführt wurde.
- Die Secondary Actions (Cancel) sind aus trivialen Gründen nicht aufgelistet, wurden aber erfolgreich getestet.
- Das Logout, welches ausser auf der Startseite von überall aus möglich ist, wurde erfolgreich getestet und wird aus Wiederholungsgründen nicht nochmals bei den entsprechenden Seiten aufgelistet. Das gleiche gilt für die Links in der Navigationsleiste.

3.1.3 START PAGE

3.1.3.1 TESTSPEZIFIKATION

Tabelle 30: Testspezifikation Start Page

Nr.	Beschreibung	Testdaten	Erwartetes Resultat
T100	Der Benutzer will sich beim System anmelden. Der **Login**-Dialog wird angezeigt.	Keine bzw. nur Benutzername oder Passwort wurden eingegeben	System zeigt für die leeren Felder eine Fehlermeldung an.
T101	"	Syntaktisch ungültige E-Mail-Adresse	System zeigt eine Fehlermeldung an, dass die eingegebene E-Mail-Adresse ungültig ist.
T102	"	Noch nicht im System vorhandene E-Mail-Adresse	System zeigt eine Fehlermeldung an, dass für die eingegebene E-Mail-Adresse kein User registriert ist.
T103	"	Gültige und bereits im System vorhandene E-Mail-Adresse, falsches Passwort	System zeigt eine Fehlermeldung an, dass das Login fehlgeschlagen ist.
T104	"	Gültige und bereits im System vorhandene E-Mail-Adresse, korrektes Passwort	Benutzer wird auf die Project Dashboard Seite weitergeleitet.
T105	"	Passwort 3x falsch eingegeben	System sperrt den Benutzer für zehn Minuten.
T106	Der Benutzer will sich beim System registrieren. Der **Register**-Dialog wird angezeigt.	Keine bzw. nicht alle Felder wurden ausgefüllt	System zeigt für die leeren Felder eine Fehlermeldung an.
T107	"	Syntaktisch ungültige E-Mail-Adresse	System zeigt eine Fehlermeldung an, dass die eingegebene E-Mail-Adresse ungültig ist.
T108	"	Im System vorhandene E-Mail-Adresse	System zeigt eine Fehlermeldung an, dass die eingegebene E-Mail-Adresse besetzt ist.
T109	"	Neue E-Mail-Adresse, Vorname, Nachname, Passwort,	Neuer Benutzer wird im System erfasst, an Benutzer wird erfolgreiche Meldung angezeigt.
T110	Benutzer hat sein Passwort vergessen.	E-Mail-Adresse	System schickt an die eingegebene E-Mail-Adresse eine E-Mail mit Aktivierungslink.

3.1.3.2 TESTPROTOKOLL VOM 21.05.2012

Tabelle 31: Testprotokoll Start Page

Nr.	Resultat	Kommentar
T100	OK	
T101	OK	
T102	OK	
T103	OK	
T104	OK	
T105	NOK	Nicht implementiert
T106	OK	
T107	OK	
T108	OK	
T109	OK	
T110	NOK	Nicht implementiert

3.1.4 PROJECT DASHBOARD

3.1.4.1 TESTSPEZIFIKATION

Tabelle 32: Testspezifikation Project Dashboard

Nr.	Beschreibung	Testdaten	Erwartetes Resultat
T200	Benutzer hat sich erfolgreich beim System angemeldet.	Keine	System zeigt die Projekte an, für die er Mitglieder ist, falls welche existieren. Die aktuelle Benutzerrolle, die Locales für das Projekt sowie die Key-Statistik wird angezeigt.
T201	"	Keine	Default-Locale wird immer als erste Locale angezeigt.
T202	Benutzer möchte neues Projekt erstellen. Er klickt auf den Add-Button. Der **Add Project**-Dialog wird angezeigt.	Keine	System zeigt für die leeren Felder eine Fehlermeldung an.
T203	"	Projektname, Default-Locale, Supported Locale. Default und Supported Locale sind nicht identisch	Neues Projekt mit den gewählten Locales wird im System erfasst, dem Benutzer wird erfolgreiche Meldung angezeigt.
T204	"	Projektname, Default-Locale, Supported Locale. Default und Supported Locale sind identisch	System zeigt eine Fehlermeldung an, dass Default und Supported Locale sich unterscheiden sollten.
T205	Benutzer möchte ein Projekt löschen. Er klickt auf den Delete-Button des zugehörigen Projekts.	Keine	System löscht Projekt. An Benutzer wird erfolgreiche Meldung angezeigt.
T206	Benutzer klickt auf die Project Info Action.	Keine	System zeigt die zum selektierten Projekt zugehörige Project Info Seite an.

3.1.4.2 TESTPROTOKOLL VOM 21.05.2012

Tabelle 33: Testprotokoll Project Dashboard

Nr.	Resultat	Kommentar
T200	OK	
T201	NOK	Nicht immer wird die Default-Locale als erste Locale gezeigt. Das System verhält sich nicht deterministisch.
T202	OK	
T203	OK	
T204	NOK	Nicht impementiert
T205	OK	
T206	OK	

3.1.5 PROJECT INFO

3.1.5.1 TESTSPEZIFIKATION

Tabelle 34: Testspezifikation Project Info

Nr.	Beschreibung	Testdaten	Erwartetes Resultat
T300	Benutzer ist ein Project Manager.	Keine	Die Add- und Delete-Buttons, um Benutzer und Locales dem Projekt hinzuzufügen bzw. vom Projekt zu löschen, sind aktiviert.
T301	Benutzer ist kein Project Manager.	Keine	Die Add- und Delete-Buttons, um Benutzer und Locales dem Projekt hinzuzufügen bzw. vom Projekt zu löschen, sind deaktiviert.
T302	Project Manager möchte einen Benutzer am Projekt zuweisen. Er klickt auf den Add-Button einer Rollensektion. Der **Invite User**-Dialog wird angezeigt.	Keine	System zeigt für das leere Feld **E-Mail** eine Fehlermeldung an.
T303	"	Falsche E-Mail-Adresse	System zeigt eine Fehlermeldung an, dass kein Benutzer mit der eingegebenen E-Mail-Adresse registriert ist.
T304	"	E-Mail-Adresse eines Benutzers, der bereits an Projekt zugewiesen ist	System zeigt eine Fehlermeldung an, dass ein Benutzer mit der eingegebenen E-Mail-Adresse bereits am Projekt zugewiesen ist.
T305	"	E-Mail-Adresse eines im System registrierten Benutzers, der noch nicht an Projekt zugewiesen ist	System weist den Benutzer an Projekt zu. Das System sendet dem Benutzer eine E-Mail. An Project Manager wird erfolgreiche Meldung angezeigt. Name und Vorname des Benutzers wird in Liste der gewählten Rolle eingetragen.
T306	Project Manager möchte Benutzer vom Projekt löschen. Er klickt auf den Delete-Button eines Projektmitglieds.	Keine	System löscht Benutzer vom Projekt. An Project Manager wird erfolgreiche Meldung angezeigt.
T307	Project Manager möchte dem Projekt eine neue Locale hinzufügen. Er klickt auf den Add-Button in der **Manage Locales** Sektion. Der **Add Locale**-Dialog wird angezeigt.	Noch nicht im Projekt verfügbare Locale ausgewählt	System fügt neue Locale dem Projekt hinzu. An Project Manager wird erfolgreiche Meldung angezeigt. Beschreibung und Code der neuen Locale wird in die **Manage Locales** Sektion eingetragen.
T308	"	Bereits im Projekt verfügbare Locale ausgewählt	System zeigt eine Fehlermeldung an, dass die neue Locale sich von den bestehenden unterscheiden sollte.

T309	Project Manager möchte bestehende Non-Default-Locale vom Projekt löschen. Er klickt auf den Delete-Button der gewünschten Locale.	Keine	System löscht Locale vom Projekt. An Project Manager wird erfolgreiche Meldung angezeigt.
T310	Benutzer hat keine bestimmte Rolle.	Keine	Default-Locale wird in der **Manage Locale** Sektion immer zuerst angezeigt und kann nicht gelöscht werden.
T311	Benutzer hat keine bestimmte Rolle.	Keine	In der **Rest Information** Sektion werden die korrekten Locale-Codes, der Projekt-Token und die verfügbaren Exportformate angezeigt.

3.1.5.2 TESTPROTOKOLL VOM 23.05.2012

Tabelle 35: Testprotokoll Project Info

Nr.	Resultat	Kommentar
T300	OK	
T301	NOK	Nicht implementiert
T302	OK	
T303	OK	
T304	OK	
T305	(OK)	Emailversand nicht implementiert
T306	OK	
T307	OK	
T308	NOK	Nicht implementiert
T309	OK	
T310	OK	
T311	OK	

3.1.6 PROJECT STRUCTURE

3.1.6.1 TESTSPEZIFIKATION

Tabelle 36: Testspezifikation Project Structure

Nr.	Beschreibung	Testdaten	Erwartetes Resultat
T400	Projekt hat noch keine Gruppen definiert, Keys jedoch schon.	Keine	Auf linker Seite wird Root-Node mit Projektname anzeigt. Auf rechter Seite sind die zum Projekt gehörenden Keys zu sehen.
T401	Benutzer macht Rechtsklick auf Root-Node.	Keine	Kontextmenü zeigt nur die Items **Add Group** und **Export** an.
T402	Benutzer möchte eine neue Gruppe erstellen. Er klickt auf den Kontextmenüpunkt **Add Group**. Der **Add Group**-Dialog wird angezeigt.	Keine	System zeigt für das leere Feld **Name** eine Fehlermeldung an.
T403	"	Gruppename, der noch nicht existiert	System fügt neue Gruppe dem Projekt hinzu. Dem Benutzer wird erfolgreiche Meldung angezeigt. Neue Gruppe wird im Gruppenbaum eingetragen.
T404	"	Gruppename, der im gleichen Baumlevel bereits existiert	System zeigt eine Fehlermeldung an, dass sich im aktuellen Baumlevel bereits eine Gruppe mit gleichem Namen befindet.
T405	"	Gruppename, der nicht im aktuellen, jedoch in anderen Baumleveln bereits existiert	System fügt neue Gruppe dem Projekt hinzu. Dem Benutzer wird erfolgreiche Meldung angezeigt. Neue Gruppe wird im Gruppenbaum eingetragen.
T406	Benutzer macht Rechtsklick auf Gruppe (nicht Root-Node).	Keine	Kontextmenü zeigt die Items **Add Group**, **Remove Group** und **Export** an.
T407	Benutzer macht Rechtsklick auf Gruppe (nicht Root-Node), nachdem min. ein Key selektiert wurde.	Keine	Kontextmenü zeigt die Items **Add Keys**, **Add Group**, **Remove Group** und **Export** an.
T408	Benutzer hat neue Gruppe erstellt. Er wählt min. einen Key aus, macht Rechtsklick auf die erstellte Gruppe und klickt auf **Add Keys**.	Keine	System assoziert selektierte Keys mit der Gruppe. Keys werden im Gruppenbaum in der erstellten Gruppe eingetragen.
T409	Benutzer möchte eine Gruppe löschen. Er selektiert die Gruppe, macht Rechtsklick und klickt auf **Remove Group**.	Keine	System entfernt die Gruppe mitsamt den assoziierten Keys vom Projekt. Dem Benutzer wird erfolgreiche Meldung angezeigt. Gelöschte Gruppe wird vom Gruppenbaum gelöscht.

T410	Benutzer selektiert einen Key in einer Gruppe, macht Rechtsklick und klickt auf **Remove Key**	Keine	System entfernt den Key von der Gruppe. Gelöschter Key wird im Gruppenbaum entfernt.
T411	Der Gruppenbaum umfasst bereits mehrere Gruppen inkl. Subgruppen und Keys. Benutzer möchte den gesamten Gruppenbaum exportieren. Er macht Rechtsklick auf Root-Node und selektiert **Export**.	Keine	System generiert eine PDF-Datei mit allen Gruppen und Keys. Benutzer kann die Datei herunterladen.
T412	Der Gruppenbaum umfasst bereits mehrere Gruppen inkl. Subgruppen und Keys. Benutzer möchte eine Gruppe exportieren. Er macht Rechtsklick auf die Gruppe selektiert **Export**.	Keine	System generiert eine PDF-Datei mit der selektierten Gruppe inkl. Keys und Subgruppen. Benutzer kann die Datei herunterladen.

3.1.6.2 TESTPROTOKOLL VOM 23.05.2012

Tabelle 37: Testprotokoll Project Structure

Nr.	Resultat	Kommentar
T400	OK	
T401	OK	
T402	OK	
T403	(OK)	Anzeige der Meldung nicht implementiert
T404	NOK	Nicht implemenetiert
T405	(OK)	Anzeige der Meldung nicht implementiert
T406	NOK	Der Kontextmenüeintrag **Add Keys** wird auch angezeigt, obwohl keine Keys selektiert wurden. Nicht Implementiert.
T407	OK	
T408	OK	
T409	(OK)	Anzeige der Meldung nicht implementiert
T410	OK	
T411	OK	
T412	OK	

3.1.7 PROJECT TRANSLATIONS

Die Systemtests für die Project Translations Seite werden in zwei Teile aufgeteilt.
Im ersten Teil werden allgemeine Funktionalitäten (Übersetzungen exportieren, Filterung usw.) getestet. Im zweiten Teil werden Tests zum Workflow durchgeführt.

3.1.7.1 TESTSPEZIFIKATION ALLGEMEIN

Tabelle 38: Testspezifikation Project Translations: Allgemein

Nr.	Beschreibung	Testdaten	Erwartetes Resultat
T500	Benutzer möchte alle Übersetzungen exportieren. Er klickt auf den Export-Button. Der **Export Translations**-Dialog wird angezeigt.	Keine	System exportiert alle Übersetzungen der Default-Locale in das Java-Format. Benutzer kann Datei herunterladen.
T501	"	Gewünschte Locale und Format	System exportiert alle Übersetzungen der selektierten Locale in das selektierte Format. Benutzer kann Datei herunterladen.
T502	Benutzer möchte Entries nach Locales filtern.	Eine Locale selektiert	System zeigt alle Entries der selektierten Locale an.
T503	"	Mehrere Locales selektiert	System zeigt alle Entries der selektierten Locales an.
T504	Benutzer möchte Entries hierarchisch filtern. Er drückt auf den **Hierarchical**-Button und selektiert einen Sub-Key im Key-Baum.	Keine	System zeigt alle Entries an, die zum selektierten Sub-Key gehören.
T505	Benutzer möchte Entries nach Gruppen filtern. Er drückt auf den **Group**-Button und selektiert eine Gruppe.	Keine	System zeigt alle Entries an, die zur selektierten Gruppe gehören.
T506	Benutzer möchte Entries nach Stati filtern.	Ein Status selektiert	System zeigt alle Entries an, welche sich im selektierten Status befinden.
T507	"	Mehrere Stati selektiert	System zeigt alle Entries an, welche sich in den selektierten Stati befinden.
T508	Benutzer möchte nach Texten filtern. Nach Eingabe des Wertes wird der Fokus des Textfeldes verlassen.	Wert, der in den Texten nicht vorkommt	System zeigt keine Entries an.
T509	"	Wert, der in den Texten vorkommt	System zeigt Entries an, welche den eingegebenen Wert besitzen.
T510	"	Es wird kein Wert eingegeben, Textfeld verliert aber Fokus	System zeigt alle Entries an.

3.1.7.2 TESTPROTOKOLL ALLGEMEIN VOM 24.05.2012

Tabelle 39: Testprotokoll Project Translations: Allgemein

Nr.	Resultat	Kommentar
T500	OK	
T501	OK	
T502	OK	
T503	OK	
T504	OK	
T505	OK	
T506	OK	
T507	OK	
T508	OK	
T509	OK	
T510	OK	

3.1.7.3 TESTSPEZIFIKATION WORKFLOW

Für Entries wird angenommen, dass jede Rolle immer alle Entries sehen kann (Filter = All).

Tabelle 40: Testspezifikation Project Translations: Workflow

Nr.	Beschreibung	Testdaten	Erwartetes Resultat
T550	Benutzer ist Project Manager oder Developer.	Keine	**Add Key**-Button ist aktiviert.
T551	" Er klickt auf den Add Key-Button. Der **Add Key**-Dialog wird angezeigt.	Keine	System zeigt für die leeren Felder eine Fehlermeldung an.
T552	"	Key, der noch nicht im Projekt existiert, Defaulttext	System erfasst den Key und erstellt für alle Locales einen Entry im status **new**. Der Defaulttext wird im Default-Entry eingetragen. System zeigt Benutzer erfolgreiche Meldung an.
T553	"	Key, der bereits im Projekt existiert, Defaulttext	System zeigt Fehlermeldung an, dass der Key bereits im Projekt existiert.

T554	Benutzer ist weder Project Manager noch Developer.	Keine	Add Key-Button ist deaktiviert.
T555	Benutzer ist Project Manager oder Developer und möchte einen Key umbenennen.	Keine	Das Umbenennen eines Keys ist möglich.
T556	" Er klickt auf den Key und nach Eingabe des Werts klickt er auf **Save**.	Neuer Keywert	System speichert den neuen Keywert im System ab. Am Benutzer wird erfolgreiche Meldung angezeigt.
T557	"	Leerer Keywert	System zeigt Fehlermeldung an, dass ein Wert erforderlich ist.
T558	Benutzer ist weder Project Manager noch Developer.	Keine	Das Umbenennen eines Keys ist nicht möglich.
T559	Benutzer ist Project Manager oder Developer.	Keine	Delete Key-Button ist für jeden erfassten Key aktiviert.
T560	Benutzer ist weder Project Manager noch Developer.	Keine	Delete Key-Button ist für jeden erfassten Key deaktiviert.
T561	Benutzer ist Project Manager oder Developer und möchte einen Key löschen. Der Delete Key-Button beim gewünschten Key wird betätigt.	Keine	System löscht Key vom Projekt mitsamt den zugehörigen Entries. Am Benutzer wird erfolgreiche Meldung angezeigt. Key und Entries sind im GUI nicht ersichtlich.
T562	Benutzer ist Developer.	Keine	Developer kann Keys erfassen, löschen und umbenennen. Alle Entries sind read-only.
T563	Benutzer ist Texter.	Keine	Texter kann Default-Entries im Status **new**, **reopened** oder **rejected** bearbeiten und weiterleiten.
T564	Benutzer ist Translator.	Keine	Translator kann Non-Default-Entries im Status **ready**, **reopened** oder **rejected** bearbeiten und weiterleiten.
T565	Benutzer ist Reviewer.	Keine	Reviewer kann Entries im Status **translated** bearbeiten, weiterleiten oder zurückweisen. Er kann auch Default-Entries im Status **ready** bearbeiten, weiterleiten oder zurückweisen.
T566	Benutzer ist Project Owner.	Keine	Project Owner kann Keys erfassen, löschen und umbenennen. Er kann Entries im Status **ready**, **translated** oder **reviewed** zurückweisen. Er kann alle Entries ausser die im Status **released** bearbeiten oder weiterleiten. Entries im Status **released** können wiedereröffnet werden.

T567	Benutzer ist Project Owner oder Reviewer. Ein Entry wird rejected. Der **Reject**-Dialog erscheint.	Keine	System erfasst keinen Kommentar. Entry wechselt in den Status **rejected**.
T568	"	Text wird im Kommentarfeld eingegeben	System erfasst einen Kommentar im Entry. Entry wechselt in den Status **rejected**.
T569	Benutzer ist Project Owner. Ein Entry wird wiedereröffnet. Der **Reopen**-Dialog erscheint.	Keine	System erfasst keinen Kommentar. Entry wechselt in den Status **reopened**.
T570	"	Text wird im Kommentarfeld eingegeben	System erfasst einen Kommentar im Entry. Entry wechselt in den Status **reopened**.
T571	Benutzer ist kein Developer und möchte einen Entry bearbeiten. Nach der Bearbeitung wird Entry weitergeleitet.	Text ist leer	System zeigt Fehlermeldung an, dass der Text leer ist.
T572	"	Text ist nicht leer bzw. er wurde bearbeitet	System speichert den neuen Text im Entry ab und zeigt dem Benutzer eine erfolgreiche Meldung an.
T573	Benutzer ist Texter. Default - Entry ist im Status **new**, **reopened** oder **rejected**. Default-Entry wird bearbeitet und weitergeleitet.	Keine	Default-Entry wechselt in den Zustand **ready**.
T574	Benutzer ist Translator. Non-Default-Entry ist im Status **ready**, **reopened** oder **rejected**. Non-Default-Entry wird bearbeitet und weitergeleitet.	Keine	Non-Default-Entry wechselt in den Zustand **translated**.
T575	Benutzer ist Reviewer. Default-Entry ist im Status **ready**. Default-Entry wird bearbeitet und weitergeleitet.	Keine	Default-Entry wechselt in den Zustand **reviewed**.
T576	Benutzer ist Reviewer. Default-Entry ist im Status **ready**. Default-Entry wird zurückgewiesen.	Keine	Default-Entry wechselt in den Zustand **rejected**.
T577	Benutzer ist Reviewer. Non-Default-Entry ist im Status **translated**. Non-Default-Entry wird bearbeitet und weitergeleitet.	Keine	Non-Default-Entry wechselt in den Zustand **reviewed**.

T578	Benutzer ist Reviewer. Non-Default-Entry ist im Status **translated**. Non-Default-Entry wird zurückgewiesen.	Keine	Non-Default-Entry wechselt in den Zustand **rejected**.
T579	Benutzer ist Project Manager. Entry ist im Status **reviewed**. Entry wird zurückgewiesen.	Keine	Entry wechselt in den Zustand **rejected**.
T580	Benutzer ist Project Manager. Entry ist im Status **reviewed**. Entry wird weitergeleitet.	Keine	Entry wechselt in den Zustand **released**.
T581	Benutzer ist Project Manager. Entry ist im Status **released**. Entry wird wiedereröffnet.	Keine	Entry wechselt in den Zustand **reopened**.

3.1.7.1 TESTPROTOKOLL WORKFLOW VOM 24.05.2012

Tabelle 41: Testprotokoll Project Translations: Workflow

Nr.	Resultat	Kommentar
T550	OK	
T551	OK	
T552	OK	
T553	OK	
T554	OK	
T555	OK	
T556	OK	
T557	OK	
T558	OK	
T559	OK	
T560	OK	
T561	OK	
T562	OK	
T563	OK	
T564	OK	
T565	OK	
T566	OK	

T567	OK	
T568	OK	
T569	OK	
T570	OK	
T571	NOK	Nicht implementiert. Leere Entries können trotzdem weitergeleitet werden.
T572	OK	
T573	OK	
T574	OK	
T575	OK	
T576	OK	
T577	OK	
T578	OK	
T579	OK	
T580	OK	
T581	OK	

3.1.8 REST-SCHNITTSTELLE

Für die REST-Schnittstelle gibt es zwei URI' s:

Key erstellen:

URI 1: <server:port>/Centralator/rest/projects/<token>/createKey?key=<key>&value=<value>

Beispiel:
localhost:8080/Centralator/rest/projects/X4ASFxasd2/createKey?key=global.button.save&value=Speichern

Übersetzungen exportieren:

URI 2: <server:port>/Centralator/rest/projects/<token>/export?locale=<locale-code>&format=<format>

Beispiel:
localhost:8080/Centralator/rest/projects/X4ASFxasd2/export?locale=de&format=XML

Die Testspezifikation bezieht sich auf die Platzhalter und URI-Nummern.

3.1.8.1 TESTSPEZIFIKATION

Tabelle 42: Testspezifikation REST-Schnittstelle

Nr.	Beschreibung	Testdaten	Erwartetes Resultat
T600	URI 1 oder URI 2	**<token>** leer	System findet Ressource nicht. Es meldet HTTP Status 404.
T601	URI 1 oder URI 2	Falscher **<token>**, gehört zu keinem Projekt	System meldet, dass der Token ungültig ist.
T602	URI 1 oder URI 2	**<token>** gültig, **<key>**, **<value>**, **<locale-code>** oder **<format>** leer	System meldet, dass Werte fehlen.
T603	URI 1	**<token>** gültig, **<key>** und **<value>**	System erfasst für das entsprechende Projekt einen neuen Key **<key>** mit Defaulttext **<value>**. System meldet OK.
T604	URI 1	**<token>** gültig, **<key>** und **<value>**, **<key>** existiert bereits.	System meldet, dass **<key>** bereits existiert.
T605	URI 2	**<token>**, **<locale-code>** und **<format>** gültig	System exportiert alle vorhandenen Übersetzungen der Locale **<locale-code>** in das Format **<format>**.
T606	URI 2	**<token>** und **<format>** gültig, **<locale-code>** ungültig	System meldet, dass die Locale **<locale-code>** ungültig ist.
T607	URI 2	**<token>** und **<format>** gültig, **<locale-code>** ungültig für Projekt	System meldet, dass die Locale **<locale-code>** für das Projekt ungültig ist.
T608	URI 2	**<token>** und **<locale-code>** gültig, **<format>** ungültig	System meldet, dass das Format **<format>** ungültig ist.
T609	URI 1 und URI 2	Alle Platzhalter gültig, an URI wird ein weiterer Parameter angehängt: z.B. **&myVar=20**	System meldet, dass zu viele Parameter angegeben wurden.

3.1.8.2 TESTPROTOKOLL VOM 23.05.2012

Tabelle 43: Testprotokoll REST-Schnittstelle

Nr.	Resultat	Kommentar
T600	OK	
T601	OK	
T602	OK	
T603	OK	
T604	NOK	Nicht implementiert, es können gleiche Keys erfasst werden.
T605	OK	
T606	OK	
T607	NOK	Nicht implementiert. Für gültige Locales, die nicht am gewählten Projekt zugewiesen sind, werden leere Übersetzungen exportiert.
T608	OK	
T609	NOK	Nicht implementiert. Für korrekte URI' s mit zusätzlich angehängtem Parameter-String wird der String ignoriert und die gewünschte Aktion trotzdem ausgeführt.

3.2 UNIT TESTS / INTEGRATION TESTS

Für die wesentlichsten Klassen wurden Unit und Integration Tests erstellt. Die Testabdeckung und die getesteten Packages sind im Kapitel 5.4 beschrieben.

3.3 USABILITY-FEEDBACK

Im Rahmen dieses Projekts wurden aus zeitlichen Gründen keine Usability-Tests durchgeführt. Um trotzdem eine Meinung zum User Interface einzuholen, wurde am 18. April 2012 eine Kurzpräsentation der Applikation beim Kunden foryouandyourcustomers in Pfäffikon ZH gehalten. Anwesend war auch Johann Richard als Experte für User Experience und Design. Herr Richard hat den Studierenden im Anschluss an die Präsentation folgenden Input gegeben:

- **Andere Icons verwenden**
  Auf der Translations-Seite sollten andere Icons für das Weiterleiten oder Ablehnen von Übersetzungen verwendet werden, welche für den Benutzer intuitiver zu verstehen sind. Die Translations-Seite ist die wichtigste Seite, da hier die meisten Benutzer ihre Arbeit effizient erledigen müssen.
  ☑ Das GUI wurde gemäss Feedback angepasst

- **In der Navigation Links statt Buttons verwenden**
  Da viele Buttons bereits in den Dialogen benutzt werden, sollten Buttons nicht auch noch in der Navigation verwendet werden, dafür aber Links.
  ☑ Das GUI wurde gemäss Feedback angepasst

- **Weisser Text auf orangem Hintergrund ist schlecht zu sehen**
  Auf der Translations-Seite sollte für das Anzeigen der Keys eine andere Farbe eingesetzt werden, da weiss auf orangem Hintergrund schlecht zu sehen ist.
  ☑ Das GUI wurde gemäss Feedback angepasst

- **Primary + Secondary Actions besser unterscheiden**
  Bei Formularen bzw. Dialogen ist es typisch, dass ein Benutzer sowohl primäre Aktionen (z.B. Eingaben speichern) als auch sekundäre Aktionen (z.B. Formular schliessen, Eingaben zurücksetzen) ausführt. Weil die Primary Actions für den Benutzer wichtiger sind, sollten sich diese im User Interface von den Secondary Actions unterscheiden. Herr Richard hat den Studierenden zu diesem Thema die URL [Url16] empfohlen.
  In der Applikation sollten daher für Formulare nicht gleiche Buttons für Primary und Secondary Actions verwendet werden. Wenn Buttons eingesetzt werden, sollten sich diese farblich unterscheiden. Für Secondary Actions sind Links statt Buttons auch eine Option.
  ☒ Das GUI wurde aus zeitlichen Gründen nicht angepasst

- **Labeling der Filteroptionen nachführen**
  Der Status jedes Übersetzungseintrags auf der Translations-Seite wird mit einer Farbe dargestellt. Da Übersetzungen nach Stati gefiltert werden können, wäre es für das Auge optimal, wenn die Farbe jedes Statusfilters gleich wie die Farbe des Status im Eintrag einer Übersetzung wäre, sprich eine Farblegende der Stati zu haben.
  ☒ Das GUI wurde aus zeitlichen Gründen nicht angepasst

4 ERGEBNISSE UND SCHLUSSFOLGERUNG

4.1 WAS WURDE ERREICHT?

In den vergangenen 15 Wochen wurde eine beinahe produktreife Applikation entwickelt, welche bereits über eine beträchtliche Menge an Funktionalität beinhaltet. In der ersten Phase wurde der heutige Übersetzungsprozess analysiert und Probleme aufgezeigt. Anschliessend wurden verschiedene Webapplikationen, welche bereits versuchen diese Probleme zu adressieren, auf Schwachstellen untersucht. Hauptsächlich wurden die beiden Webapplikationen LocaleApp und WebTranslateIt untersucht.

Beide Applikationen verfügen über nachfolgende Funktionalitäten:

- Zentrale Verwaltung
- Unterstützung für mindestens eine Programmiersprache

Nachfolgende Mängel wurden in der einen oder anderen Applikation gefunden:

- Kein klarer Workflow
- Keine Unterstützung für mehrere Programmiersprachen
- Keine Anbindung an andere Dienste möglich
- Schlechte Usability

Nach dieser Analyse-Phase wurde ein Architekturprototyp entwickelt und anschliessend wurde die Applikation programmiert. Während der Entwicklung wurde von Anfang an viel Zeit in die Architektur investiert, um alle erwähnten Mängel so gut wie möglich zu beheben. Centralator verfügt über nachfolgende Funktionalitäten:

- **Zentrale Verwaltung**
  Die Applikation erlaubt die zentrale Verwaltung von Benutzern, Rollen, Projekten und Übersetzungen. Allein durch diese Zentralisierung der Übersetzungen wurde der Übersetzungsprozess auf einen Schlag vereinfacht.

- **Klarer Workflow**
  Zugrunde liegt ein einfaches und intuitives Zustandsmodell, welches durch die Applikation abgebildet wurde. Die Applikation garantiert dadurch einen sequentiellen Ablauf. Zudem ist zu jedem Zeitpunkt klar ersichtlich, in welchem Zustand sich ein Text beziehungsweise Übersetzung befindet.

- **Unterstützung für mehrere Programmiersprachen**
  Die Applikation erlaubt, die erfassten Texte und Übersetzungen in die Formate YAML, PHP, RESX (C#), PDF, CSV, XML, JSON und als Properties (Java) zu exportieren. Dies erlaubt es, die Applikation in möglichst vielen Softwareprojekten einzusetzen.

- **REST API für andere Dienste**
  Die Applikation stellt ein einfaches REST API zur Verfügung, das es erlaubt, neue Keys mit einem Defaulttext zu erfassen und bestehende Keys mitsamt Übersetzung in eines der obenstehenden Formate zu exportieren. Dies erlaubt anderen Entwicklern eine massgeschneiderte Anbindung an ihre Applikationen. Zum Beispiel könnte mit wenig Aufwand ein Skript geschrieben werden, welches alle paar Minuten den neusten Stand der Übersetzungen runterlädt und in die Produktionsumgebung einer Applikation einspielt.

- **Verbesserte Usability**
  Die Applikation hat die Anzahl Benutzerinteraktionen auf ein Minimum reduziert, indem auf unnötige Dialoge und Bestätigungen verzichtet wurde und automatisch jede Änderung gespeichert wurde. Somit verhält sich die Applikation ähnlich wie ein Spreadsheet mit Auto-Save-Funktionalität.

- **Live Filter**
  Die Applikation erlaubt es vorhandene Übersetzungen auf eine Vielzahl von Arten live zu filtern. Unterstützt wird das Filtern nach Sprache, Text, Key, Gruppe und nach einem oder mehreren Zuständen. All diese Arten können zudem in einer beliebigen Reihenfolge kombiniert werden.

- **Logisches Gruppieren**
  Centralator erlaubt es als bisher einzige Applikation, Keys logisch und nicht nur hierarchisch zu Gruppieren. Typischerweise würde man pro Screen eine eigene Gruppe erstellen, wodurch einem Benutzer sofort ersichtlich wird, auf welchem Screen und somit in welchem Kontext der Key verwendet wird. Darüber hinaus ermöglicht die Applikation auch das Exportieren dieser Gruppen mitsamt den Keys und allen Übersetzungen.

4.2 WAS WURDE NICHT ERREICHT

Im Grossen und Ganzen wurden alle angestrebten Ziele erreicht. Eines der nichtfunktionalen Ziele, welches nicht erreicht wurde, ist die geforderte Performance. Dadurch das Hibernate praktisch die ganzen Abhängigkeiten auf einmal ladet, werden zum Teil auch nichtbenötigte Abhängigkeiten aufgelöst, was unnötig Zeit kostet.

4.3 SCHLUSSFOLGERUNGEN

Obschon die Integration aller Frameworks wie JSF, Primefaces, Jersey, Spring und Hibernate viel Zeit in Anspruch genommen hat, hat sich deren Einsatz auf lange Sicht definitiv bewährt. Gerade diesen Frameworks ist es zu verdanken, dass sehr schnell Fortschritte erzielt werden konnten. Spring stellt dabei das Bindeglied zwischen allen Frameworks dar und war durch seine unglaubliche Flexibilität und Erweiterbarkeit eine der besten Architekturentscheide, die getroffen wurden. Wenn die Zeit nochmals zurückgedreht werden könnte, hätte schon von Anfang an mehr Zeit in das Technologiestudium von Hibernate investiert werden sollen, da dieses Framework recht komplex ist und ein gutes Verständnis fordert, um es korrekt einzusetzen.

Nachfolgende Punkte stehen einer Kommerzialisierung noch im Weg:

- **Performance Verbesserungen**
  Am meisten Zeit müsste für Performance Verbesserungen investiert werden. Zurzeit werden praktisch alle Objektabhängigkeiten aufgelöst, auch wenn dies gar nicht zwingend gefordert wäre.

- **Concurrency**
  Obschon die Applikation paralleles Bearbeiten der gleichen Daten erlaubt, wurden die daraus resultierenden Effekte nicht näher untersucht. Hibernate ist zwar in der Lage parallele Aufrufe korrekt zu managen, aber eventuell müsste die Applikation dafür noch weiter angepasst werden.

- **Erweitern der REST API**
  Die REST API birgt ein extrem grosses Potential und lässt sich mit wenig Aufwand leicht erweitern. Die API könnte noch mehr Services zur Verfügung stellen, wie zum Beispiel das Hinzufügen von Übersetzungen zu einem Defaulttext. Durch ein Skript könnten dadurch alle Defaulttexte mitsamt den Übersetzungen in die Applikation importiert werden.

- **I18N**
  Um die Glaubwürdigkeit der eigenen Applikation zu erhöhen, sollte die eigene Applikation selbst auch mehrsprachig sein und die eigene Applikation zur Verwaltung der Daten nutzen.

- **GUI Redesign**
  Obschon die Applikation optisch ansprechend wirkt, sollte die grafische Benutzeroberfläche nochmals überarbeitet werden, damit es einen professionelleren Eindruck erweckt.

- **Projektstand publizieren**
  Nach einem Projektrelease sollte es möglich sein, einen Snapshot vom aktuellen Stand zu erstellen und diesen Stand zu publizieren. Die REST API könnte dann so erweitert werden, dass man die Wahl hätte, ob der aktuelle Stand oder der letzte publizierte Stand exportiert werden soll.

Anhang

5 QUALITÄTSMASSNAHMEN

5.1 REDMINE

Als Unterstützung für das Projekt-Management wurde die Webapplikation **Redmine** eingesetzt. Mit Redmine kann eine Projektplanung mit Hilfe eines Gantt-Diagramms erstellt werden, sodass man nie den Überblick über das Projekt verliert. Für die Projektplanung können Tasks erfasst werden, welche bis zu einem bestimmten Termin erledigt sein müssen. Für diese Arbeit wurden vor allem User Stories als Tasks erfasst. Ein weiteres Feature von Redmine ist das Wiki, auf dem beispielsweise Protokolle oder Anleitungen erfasst und publiziert werden können.

5.2 JENKINS

Als Continuous Integration Buildsystem wurde **Jenkins** eingesetzt. Jenkins erlaubt sowohl Ant- als auch Maven-Builds zu konfigurieren und auszuführen. Durch die Flexibilität von Jenkins können Plugins installiert und konfiguriert werden. Für diese Arbeit wurden die Plugins **PMD**, **Findbugs** und **Cobertura** installiert, mit denen sowohl der Code (PMD, Findbugs) als auch die Testabdeckung (Cobertura) analysiert werden kann.

5.3 STATIC CODE ANALYSIS

Für die Code-Analyse wurden die Tool **PMD** und **Findbugs** eingesetzt. Beide Tools wurden als Plugin auf den Jenkins-Build-Server installiert, sodass immer ersichtlich war, wo überall Bugs vorhanden waren. Für dieses Projekt wurden die Bugs regelmässig analysiert und gleich behoben. Gegen Ende des Projekts wurden für die letzten Bugs noch Trends erstellt.

5.3.1 PMD

Eine Auflistung aller Smells kann bei [Url17] nachvollzogen werden.

Gegen Ende des Projekts war nur noch ein Smell vorhanden und zwar ein **UnusedPrivateField**-Smell. Dieser wurde ebenfalls behoben, sodass am Ende keine Warnungen mehr im PMD-Trend zu sehen waren:

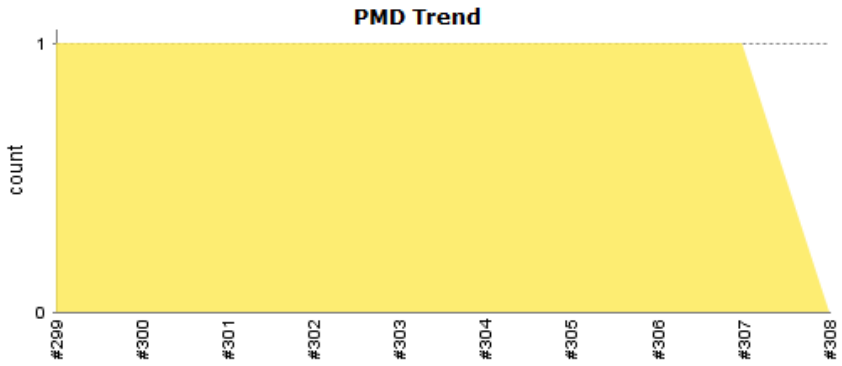
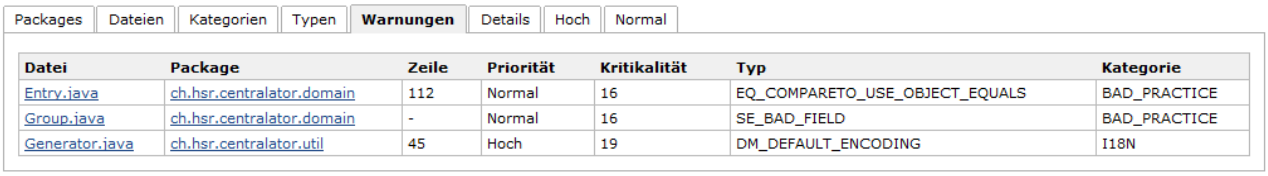


Abbildung 50: PMD Trend

5.3.2 FINDBUGS

Eine Auflistung aller Bugs kann kann bei [Url18] nachvollzogen werden.
Gegen Ende des Projekts waren 16 Bugs offen, welche bis auf die folgenden drei behoben wurden:



Packages | Dateien | Kategorien | Typen | **Warnungen** | Details | Hoch | Normal

Datei	Package	Zeile	Priorität	Kritikalität	Typ	Kategorie
Entry.java	ch.hsr.centralator.domain	112	Normal	16	EQ_COMPARETO_USE_OBJECT_EQUALS	BAD_PRACTICE
Group.java	ch.hsr.centralator.domain	-	Normal	16	SE_BAD_FIELD	BAD_PRACTICE
Generator.java	ch.hsr.centralator.util	45	Hoch	19	DM_DEFAULT_ENCODING	I18N

Abbildung 51: PMD Trend

Nachfolgend die Findbugs-Erklärungen und die Gründe für die Nicht-Behebung:

Klasse Entry

ch.hsr.centralator.domain.Entry defines compareTo(Entry) and uses Object.equals()
This class defines a compareTo(...) method but inherits its equals() method from java.lang.Object. Generally, the value of compareTo should return zero if and only if equals returns true. If this is violated, weird and unpredictable failures will occur in classes such as PriorityQueue. In Java 5 the PriorityQueue.remove method uses the compareTo method, while in Java 6 it uses the equals method. From the JavaDoc for the compareTo method in the Comparable interface: It is strongly recommended, but not strictly required that (x.compareTo(y)==0) == (x.equals(y)). Generally speaking, any class that implements the Comparable interface and violates this condition should clearly indicate this fact. The recommended language is "Note: this class has a natural ordering that is inconsistent with equals.

Wie in der Erklärung zu lesen, ist die es nicht zwingend notwendig, für die compareTo-Methode eine eigene equals-Methode zu implementieren, aber empfohlen wird es. In unserem Fall bezieht sich die compareTo-Methode aber nicht auf die Attribute des übergebenen Objekts (other), darum kann diese Meldung ignoriert werden.

```
110   @Override
111   public int compareTo(Entry other) {
112     if(this.getProjectLocale().isDefaultLocale()){
113       return -1;
114     }
115
116     if(!this.getProjectLocale().isDefaultLocale()){
117       return 1;
118     }
119     return 0;
```

Abbildung 52: comparteTo-Methode von Entry

Klasse Group

ch.hsr.centralator.domain.Group defines non-transient non-serializable instance field screenshot
This Serializable class defines a non-primitive instance field which is neither transient, Serializable, or java.lang.Object, and does not appear to implement the Externalizable interface or the readObject() and writeObject() methods. Objects of this class will not be deserialized correctly if a non-Serializable object is stored in this field.

Das Attribut **screenshot** vom Typ java.sql.Blob implementiert in der Tat nicht das Serializable Interface. Als transient markieren würde das O/R-Mapping aufbrechen, darum wird dieses Attribut so beibehalten.

Klasse Generator

Found reliance on default encoding in ch.hsr.centralator.util.Generator.generateToken(String): String.getBytes()
Found a call to a method which will perform a byte to String (or String to byte) conversion, and will assume that the default platform encoding is suitable. This will cause the application behaviour to vary between platforms. Use an alternative API and specify a charset name or Charset object explicitly.

Das unterschiedliche Encoding für verschiedene Plattformen kann in unserem Fall ignoriert werden. Für das Generieren des Tokens spielt es keine Rolle.

```
40   public static String generateToken(String projectName) {
41
42     if (messageDigest == null)
43       return null;
44
45     messageDigest.update(projectName.getBytes());
```

Abbildung 53: generateToken-Methode von Generator

Wie im Findbugs Trend zu erkennen ist, wurden die 16 Bugs bis auf 3 behoben:

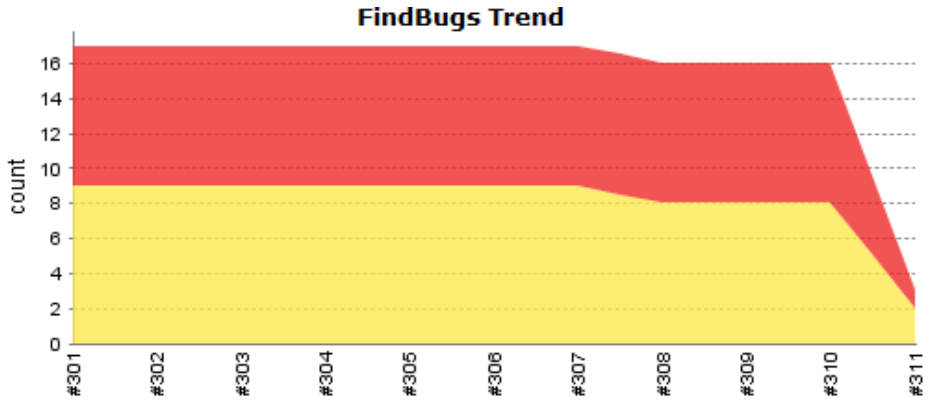


Abbildung 54: Findbugs Trend

5.4 TESTABDECKUNG UND CODESTATISTIK

Wie in den nachfolgenden beiden Abbildungen zu entnehmen ist, wurden insgesamt 16 Tests geschrieben. Trotz allem liegt die totale Line Coverage gemäss Cobertura [Url19] bei 45%. Dies liegt hauptsächlich daran, dass Spring komplette Integration Tests ermöglichte, wodurch der gesamte Java Code mitsamt der Datenbank Anbindung getestet werden konnte.

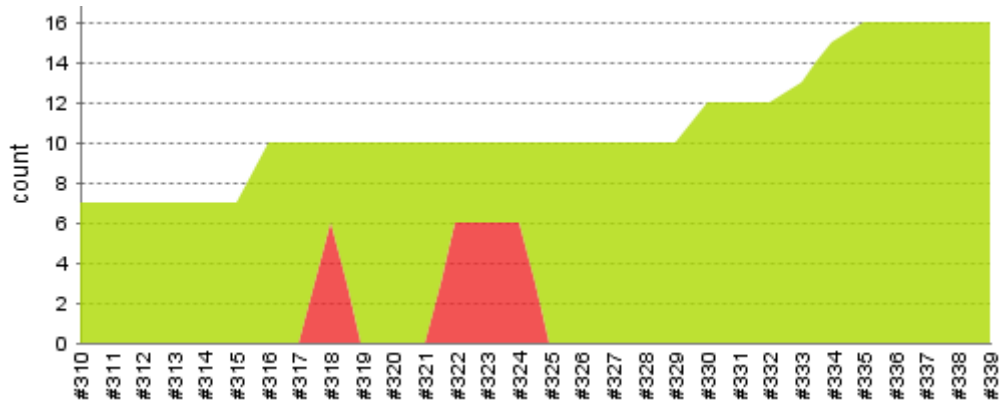


Abbildung 55: Trend der JUnit Testergebnisse

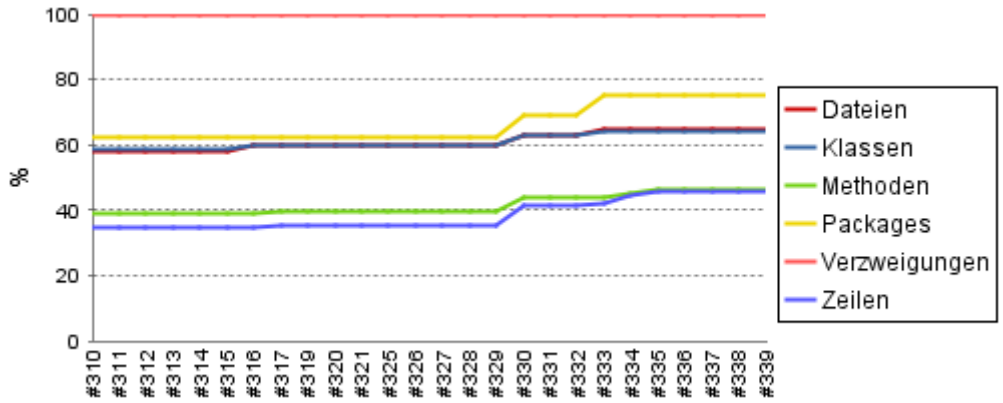


Abbildung 56: Trend der Testabdeckung

Nachfolgende Cobertura Diagramme zeigen sowohl die Testabdeckung des ganzen Projekts als auch über die einzelnen Packages. Wie aus den Abbildungen zu entnehmen ist, lag der Fokus klar auf dem Service und Application Layer.

Name	Klassen	Verzweigungen	Dateien	Zeilen	Methoden	Packages
Cobertura Testabdeckung	64% 45/70	100% 0/0	65% 40/62	45% 886/1950	46% 334/725	75% 12/16

Abbildung 57: Testabdeckung des ganzen Projekts

Name	Klassen	Verzweigungen	Dateien	Zeilen ↑	Methoden
ch.hsr.centralator.service	100% 1/1	-	100% 1/1	100% 2/2	100% 1/1
ch.hsr.centralator.domain	100% 15/15	-	100% 12/12	82% 276/337	87% 152/175
ch.hsr.centralator.service.impl	100% 5/5	-	100% 5/5	73% 191/260	80% 36/45
ch.hsr.centralator.util	75% 3/4	-	75% 3/4	66% 23/35	64% 7/11
ch.hsr.centralator.application.projectinfo	100% 2/2	-	100% 2/2	65% 105/161	54% 32/59
ch.hsr.centralator.application.projectdashboard	100% 2/2	-	100% 2/2	53% 53/100	54% 19/35
ch.hsr.centralator.application.projectstructure	100% 2/2	-	100% 2/2	50% 69/138	50% 23/46
ch.hsr.centralator.application	67% 2/3	-	67% 2/3	45% 18/40	55% 12/22
ch.hsr.centralator.application.startpage	100% 2/2	-	100% 2/2	39% 33/84	34% 11/32
ch.hsr.centralator.application.translations	80% 4/5	-	100% 2/2	35% 103/294	30% 31/102
ch.hsr.centralator.domain.rights	46% 6/13	-	46% 6/13	7% 11/151	12% 9/78
ch.hsr.centralator.application.commons.format	13% 1/8	-	14% 1/7	1% 2/139	2% 1/62
ch.hsr.centralator.application.commons	0% 0/2	-	0% 0/1	0% 0/13	0% 0/3
ch.hsr.centralator.application.commons.format.pdf	0% 0/3	-	0% 0/3	0% 0/108	0% 0/30
ch.hsr.centralator.application.commons.format.xml	0% 0/2	-	0% 0/2	0% 0/27	0% 0/10
ch.hsr.centralator.application.rest	0% 0/1	-	0% 0/1	0% 0/61	0% 0/14

Abbildung 58: Testabdeckung der einzelnen Packages

5.5 ARCHITEKTUR-ANALYSE MIT STRUCTURE 101

Structure 101 [Url20] ist ein Analyse-Tool zur Struktur- und Komplexitätsanalyse. Es ist vor allem nützlich bei Architektur-Designs, um Package-Strukturen mit den Architektur-Schichten zu vergleichen. Das Tool erlaubt Archive zu integrieren und dessen Source Code zu analysieren. Für Centralator wurde das Webarchiv **Centralator.war** analysiert. Structure 101 gibt es für verschiedene Programmiersprachen, für dieses Projekt wurde mit der Java-Version gearbeitet.

Structure 101 kann für jede Package-Stufe Analysen zu Kohäsion und Tangles (= Design-Unschönheiten wie zyklische Abhängigkeiten) berechnen. Für Centralator wurden die folgenden Levels erfasst:

Level	#	C	T
Design level 3	4	100 %	0 %
Design level 2	11	100 %	0 %
Design level 1	12	100 %	0 %
Leaf package	16	100 %	12 %
Outer class	67	100 %	24 %

Abbildung 59: Erfasste Design Levels in Structure 101

Die Spalten **#**, **C** und **T** bedeuten, wie viele Objekte (Packages / Klassen) es in einem Level gibt, welche Kohäsion der Level aufweist und welcher Grad an Tangles vorhanden ist.

Nachfolgend werden die ersten vier Levels genauer unter die Lupe genommen. Auf den Level **Outer class** wird aus Layoutgründen nicht weiter eingegangen, da dort alle Klassenabhängigkeiten gezeigt werden.

5.5.1 DESIGN LEVEL 3

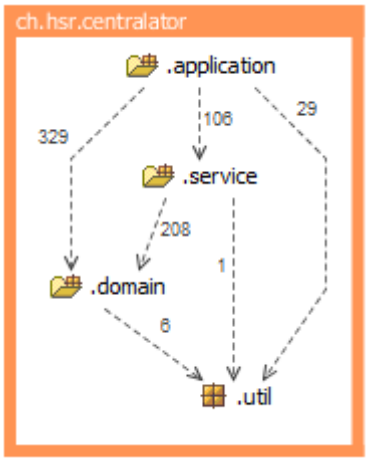


Abbildung 60: Design Level 3

Dieser Level zeigt den abstraktesten Package-Layer, sprich die Schichtenarchitektur. Die Zahlen neben den Abhängigkeitspfeilen bedeuten die Anzahl Abhängigkeiten zwischen den Objekten. In diesem Level sind keine zyklischen Abhängigkeiten erkennbar.

Structure Resultat: 4 Items, 6 Dependencies, 0 Tangles

5.5.2 DESIGN LEVEL 2

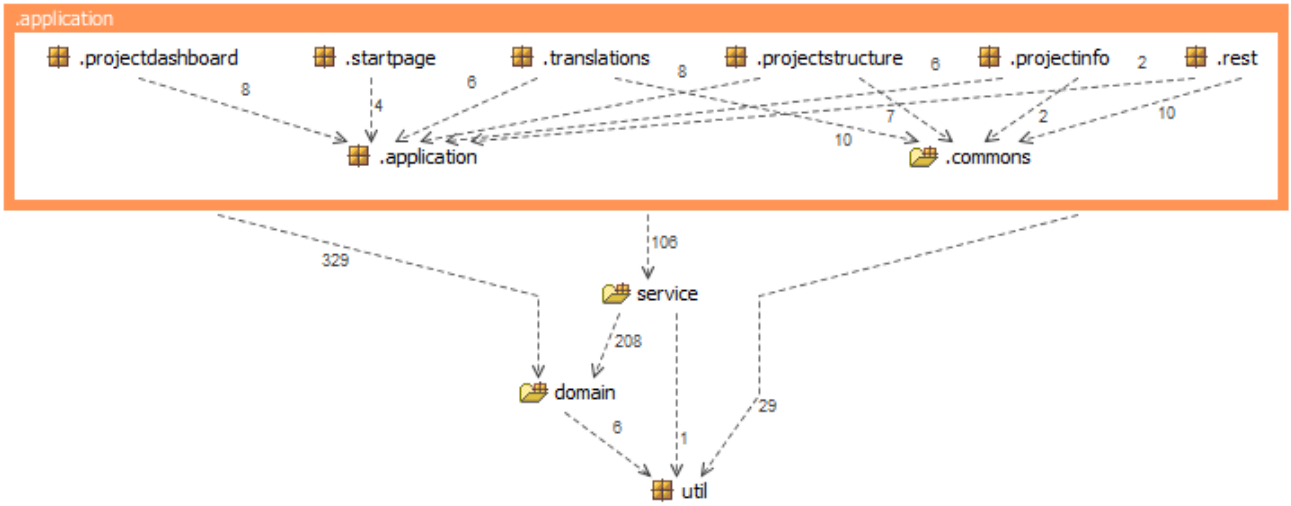


Abbildung 61: Design Level 2

Dieser Level geht in das **application**-Package detaillierter ein. Auch in diesem Level sind keine zyklischen Abhängigkeiten erkennbar.

Structure Resultat: 11 Items, 33 Dependencies, 0 Tangles

5.5.3 DESIGN LEVEL 1

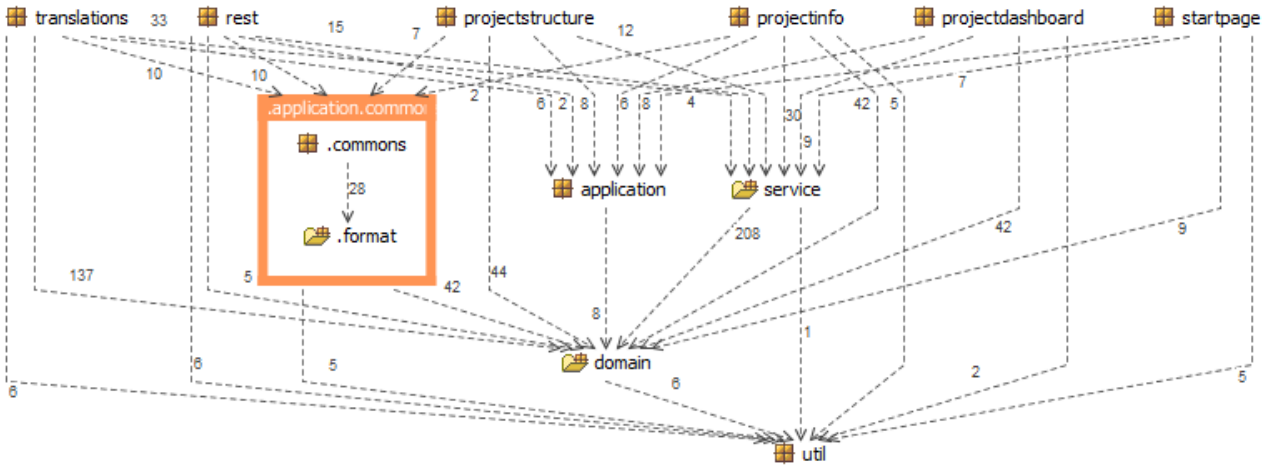


Abbildung 62: Design Level 1

Dieser Level stellt die Abhängigkeiten im **application**-Package detaillierter dar. Zusätzlich wird das Subpackage **format** im **commons**-Package gezeigt. Auch in diesem Level sind keine zyklischen Abhängigkeiten erkennbar.

Structure Resultat: 12 Items, 38 Dependencies, 0 Tangles

5.5.4 LEAF PACKAGE

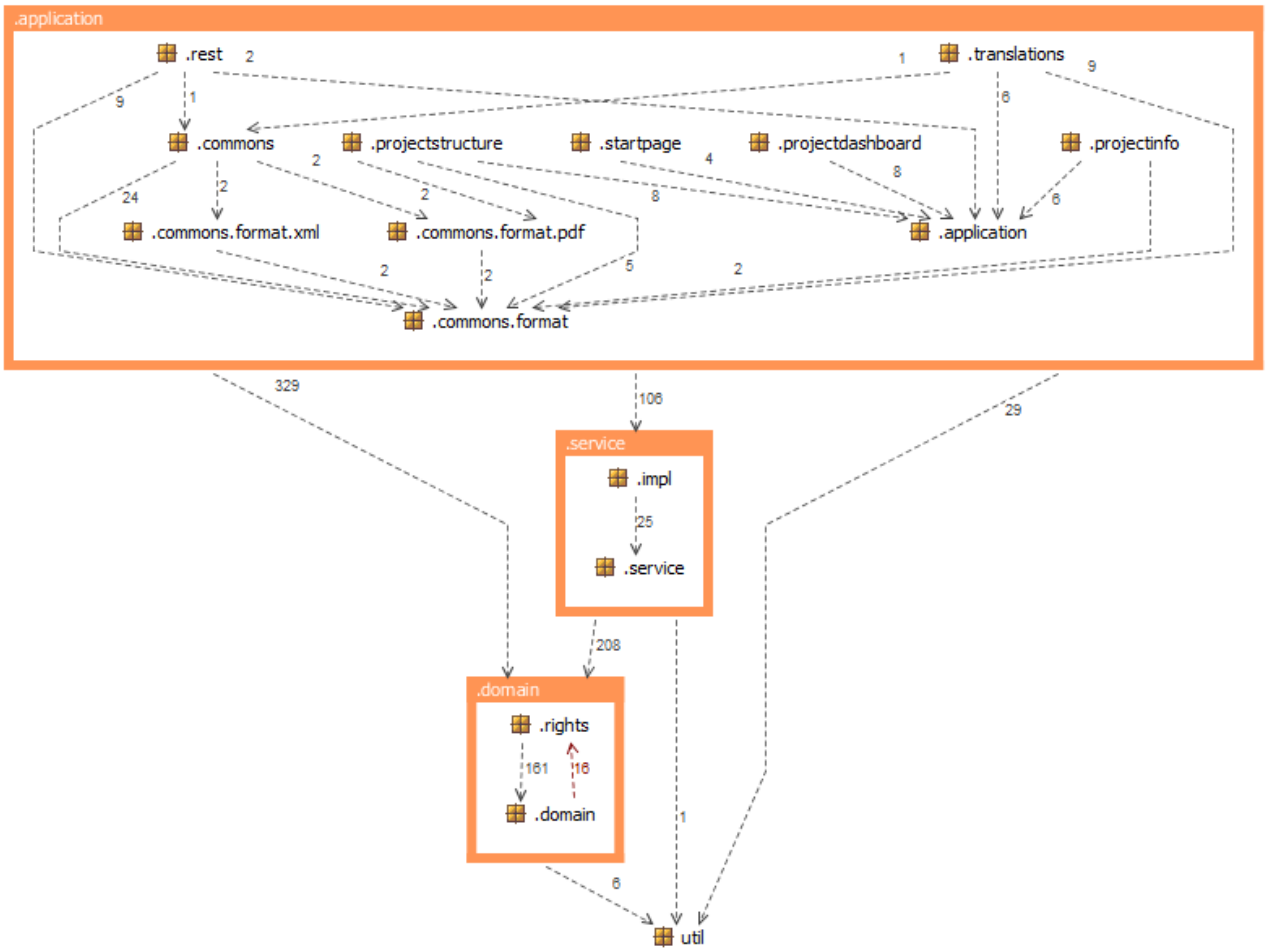


Abbildung 63: Leave Package Level

Dieser Level zeigt alle Packages des Projekts. Wie im **domain**-Package zu erkennen ist, gibt es eine zyklische Abhängigkeit zwischen dem Subpackage **rights** und dem Parent-Package **domain**. Grund für diesen Tangle ist die Implementation der Rollenrechte. Im **rights**-Package wurde für jede Rolle eine Rechte-Klasse erzeugt, welche die abstrakten Methoden der Oberklasse **Rights** implementiert. Als Beispiel kann die Klasse **DeveloperRights** genommen werden, welche die Methoden **entryEditable**, **entryRejectable**, **entryReopenable**, **entrySubmitable**, **entryVisible**, **getNextState** implementiert. Als Parameter für diese Methoden wird die Domainklasse **Entry** verwendet, welche für die Rechtevergabe bzw. den ganzen Workflow von zentraler Bedeutung ist. Als Gegenstück wird die Domain-Klasse **Role** genommen, welche eine Referenz auf die **Rights**-Klassen haben muss, damit die Rechte einer Rolle gelesen bzw. gesetzt werden können.

Abbildung 64: Tangle Packages rights / domain

Structure Resultat:16 Items, 51 Dependencies, 1 Tangle

6 INSTALLATION UND DEPLOYMENT

6.1 ENTWICKLUNGSUMGEBUNG AUFSETZEN

Um das Projekt in die Eclipse-Entwicklungsumgebung einzubinden, sind nachfolgende Schritte nötig. Vorausgesetzt wird Maven und SVN. Die nachfolgenden Schritte wurden ausschliesslich mit Maven 3 und SVN 1.7.4 durchgeführt, es sollte jedoch auch mit Maven 2 und früheren SVN Versionen möglich sein.

1. In Eclipse eine Classpath-Variable mit dem Namen **M2_REPO** definieren, welche auf das lokale Maven Repository zeigt: Bsp. **/Users/Sandro/.m2/repository**.

2. Mit der Konsole in das Workspace-Verzeichnis wechseln und nachfolgenden SVN-Befehl ausführen um das Projekt auszuchecken:

```
cd ~/workspace
svn checkout https://svns.hsr.ch/Centralator/Centralator
```

Abbildung 65: Shell – SVN command um ein Repository auszuchecken

3. Ins ausgecheckte Verzeichnis wechseln und nachfolgenden Maven-Befehl ausführen.

```
cd ~/workspace/Centralator
mvn eclipse:eclipse
```

Abbildung 66: Shell – Maven command um Eclipse Projekt Files zu generieren

   Dies generiert unter anderem die Dateien .project und .classpath, welche von Eclipse benötigt werden um das Projekt in Eclipse importieren zu können. Darüber hinaus fügt es dem Projekt automatisch Projekt Facetten hinzu, damit es als JSF 2.0 Projekt erkannt wird und somit Tooling-Support bietet.

4. Das Projekt in Eclipse importieren:
   File > Import… > Existing Projects into Workspace

5. Alle Maven Dependencies markieren, damit sie beim Deployment ins WEB-INF/lib Verzeichnis kopiert werden:
   Rechts-Klick auf das Eclipse Projekt > Properties > Deployment Assembly > Add… > Java Build Path Entries > Selektiere alle M2_REPO/*.jars > Finish

6. In Eclipse ein Tomcat 7 Server hinzufügen und die VM-Argumente anpassen, um den Perm Space zu erhöhen. Dies wird benötigt, da JSF, Spring und Hibernate recht viele Ressourcen beanspruchen und der Tomcat-Server sonst oft Memory-Probleme kriegt.
   Doppel-Klick auf den Tomcat-Server > Open launch configuration > Arguments >
   -XX:MaxPermSize=512m -Xms512M -Xmx1024M

6.1.1 MAVEN DEPENDENCIES ÄNDERN

Wenn eine Java-Library hinzugefügt oder gelöscht werden soll, muss das pom.xml File angepasst werden. Damit die Änderungen auch in Eclipse wirksam werden, muss die Projektstruktur neu generiert werden. Dazu sind folgende Schritte nötig:

1. Damit Eclipse die generierten Dateien neu ladet, muss entweder Eclipse oder das Projekt temporär geschlossen werden.

2. Um die generierten Dateien wie .project und .classpath zu löschen kann nachfolgender Maven-Befehl verwendet werden.

```
mvn eclipse:clean
```

Abbildung 67: Shell – Maven command um Eclipse Projekt Files zu lsöchen

3. Nun müssen diese Dateien neu generiert werden.

```
mvn eclipse:eclipse
```

Abbildung 68: Shell – Maven command um Eclipse Projekt Files zu generieren

4. Je nachdem was bei Schritt 1. gewählt wurde kann jetzt Eclipse gestartet oder das Projekt wieder geöffnet werden.

5. Alle Maven-Dependencies müssen erneut markiert werden, damit sie beim Deployment ins WEB-INF/lib Verzeichnis kopiert werden:
   Rechts-Klick auf das Eclipse-Projekt > Properties > Deployment Assembly > Add… > Java Build Path Entries > Selektiere alle M2_REPO/*.jars > Finish

6.2 DEPLOYMENTARTIFAKT GENERIEREN

Maven übernimmt vollständig das generieren eines WAR Files, welches auf einem Servlet-Container deployed werden kann. Der nachfolgende Maven-Befehl generiert im Projekt unter dem **target** Verzeichnis ein lauffähiges WAR File.

```
mvn clean install
```

Abbildung 69: Shell – Maven command um Projekt zu builden

6.3 DEPLOYMENT

6.3.1 DATENBANK AUFSETZEN

Für Centralator wurde als Datenbankserver ein **Mysql Server 5.5** eingesetzt. Dieser kann sehr einfach mittels **Mysql Installer für Windows** installiert werden [Url21].
Wählt man beim Installer alle Komponenten aus, so wird auch automatisch der **Mysql Workbench Client** installiert, mit welchem Datenbanken bequem verwaltet werden können.

6.3.2 MYSQL JDBC DRIVER VERWENDEN

Um über Java auf eine Mysql Datenbank mittels JDBC zugreifen zu können, muss der Mysql JDBC-Treiber als Library im Projekt existieren. Für Centralator wurde der Treiber **mysql-connector-java-5.1.18.jar** verwendet.

6.3.3 DATA DEFINITION LANGUAGE

Für Centralator stehen vier sql-Dumps zur Verfügung, mit welchem die Datenbank aufgesetzt werden kann:

Tabelle 44: SQL-Dumps für Centralator

Datei	Zweck
centralator-skeleton.sql	Erstellung der Centralator-DB ohne Daten
centralator-data.sql	Erstellung der Centralator-DB mit Demo Daten
centralator-test-skeleton.sql	Erstellung der Centralator-Test-DB ohne Daten
centralator-test-data.sql	Erstellung der Centralator-Test-DB mit Test Daten

Diese Dumps können sehr einfach mittels Mysql Workbench importiert werden. Die Abbildung unten soll dies verdeutlichen. Mit **Import from Self-Contained File** kann ein Dump angegeben werden. Mittels **Start Import** wird die Datenbank neu erzeugt.

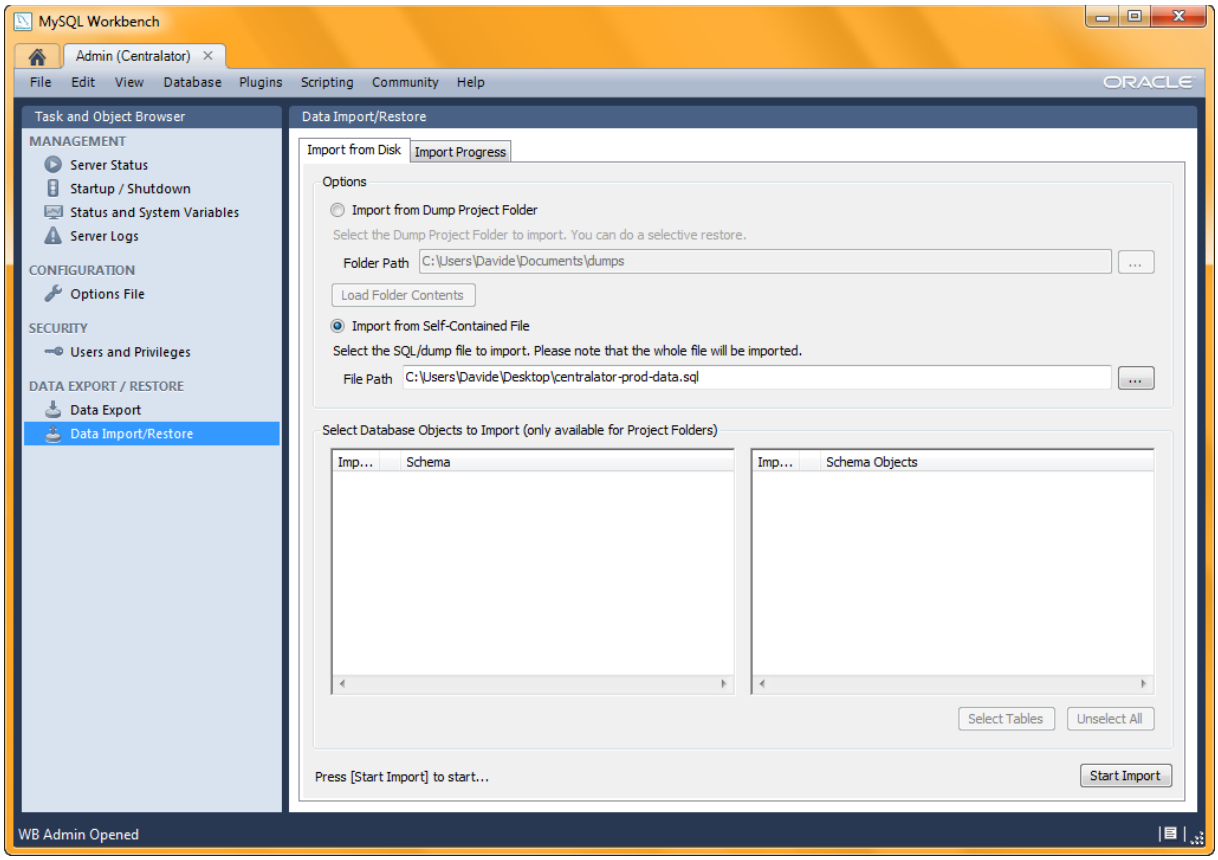


Abbildung 70: Import Dump mit Mysql Workbench

6.3.4 SPRING-CONFIGURATION FÜR DATENBANK-VERBINDUNG

Die Datenbank-Verbindung wird mittels eines Property-Files sichergestellt, welches von Spring und schlussendlich auch von Hibernate genutzt wird. In der Datei **db.properties**, welches sich im Code unter **src/main/resources** befindet, werden alle relevanten JDBC-Parameter definiert.
Für diese Studienarbeit wurde der Mysql-Server auf dem virtuellen Host **sinv-56027.edu.hsr.ch** verwendet.

db.properties

```
jdbc.driverClassName=com.mysql.jdbc.Driver
jdbc.url=jdbc:mysql://sinv-56027.edu.hsr.ch:22/centralator
jdbc.username=root
jdbc.password=**********
```

test-db.properties für Integrationstests

```
jdbc.driverClassName=com.mysql.jdbc.Driver
jdbc.url=jdbc:mysql://sinv-56027.edu.hsr.ch:22/centralator-test
jdbc.username=root
jdbc.password=**********
```

6.3.5 APPLIKATION DEPLOYEN

Sobald das **Centralator.war** Artefakt generiert wurde, kann dieses auf einem Servlet-Container deployed werden. Für diese Studienarbeit wurde die Applikation auf einem **Apache Tomcat Servlet Version 7.0** deployed.

7 PROJEKTMANAGEMENT

7.1 PROJEKTORGANISATION

- Das Projekt besteht aus zwei Mitgliedern. Die Mitglieder sind einander gleichgestellt.
- Das Projekt dauert 15 Wochen.
- Der Name der Applikation ist **Centralator**, eine Mischung aus **Centra**lization und Trans**lator**.

7.1.1 ORGANISATIONSSTRUKTUR

Abbildung 71: Portrait von Davide Giampa

Abbildung 72: Portrait von Sandro Felicioni

7.1.2 VERANTWORTUNG UND AUFGABEN

Tabelle 45: Verantwortung und Aufgaben

Name	Aufgaben
Sandro Felicioni	Entwicklung, Projektautomation, Framework Integration
Davide Giampa	Entwicklung, Infrastruktur

7.1.3 EXTERNE SCHNITTSTELLEN

Prof. Daniel Keller ist für die Beratung und Benotung zuständig. Der Kunde ist die Firma **foryouandyourcustomer** aus Pfäffikon ZH.

7.1 PROJEKTPLAN

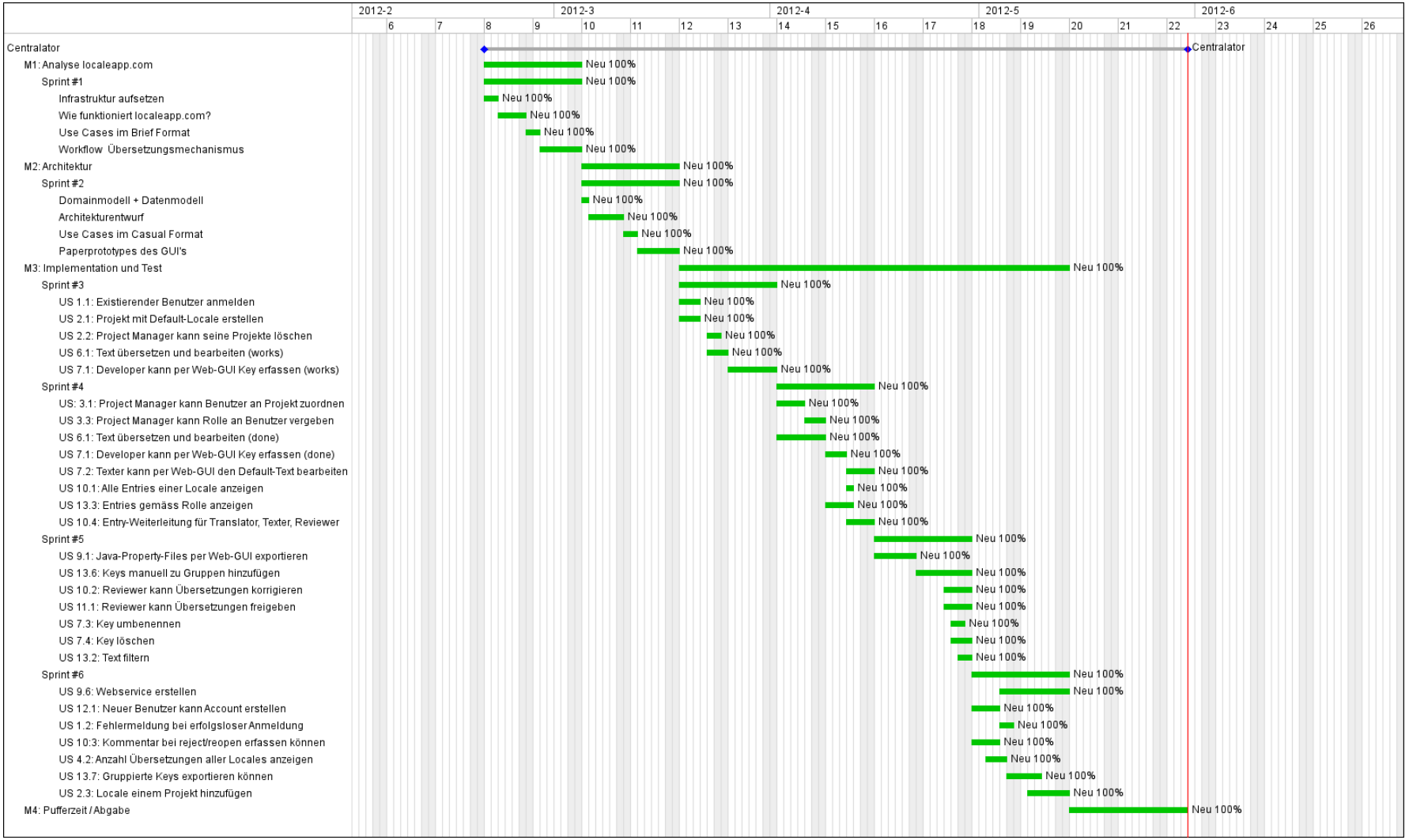


Abbildung 73: Projektplan

7.2 SOFTWARENTWICKLUNGSPROZESS

Als Softwareentwicklungsprozess wurde eine Mischung zwischen Scrum und Rup verwendet.
In den ersten zwei Wochen wurde als Inception-Phase LocaleApp analysiert. In den Wochen 3 und 4 wurde ein Architekturprototyp entwickelt und die Risiken minimiert (=Elaboration-Phase). In den Wochen 5-12 wurden total vier Sprints durchlaufen, pro Sprint zwei Entwicklungswochen.

Auf der nächsten Seite ist der SOLL-/IST-Zeitaufwand der wichtigsten User Stories dargestellt.

7.2.1 SOLL-/IST-AUFWAND DER WICHTIGSTEN USER STORIES

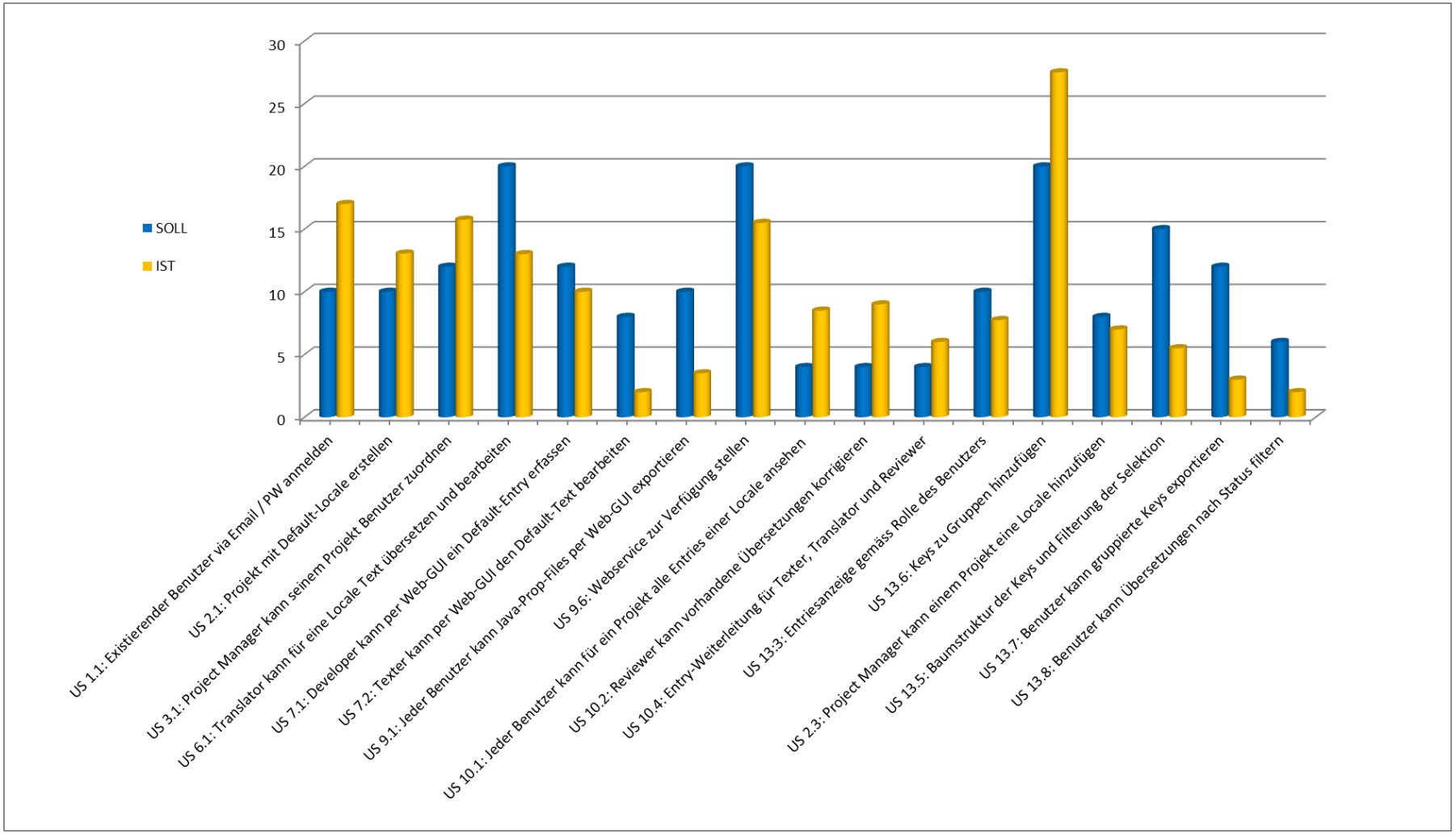


Abbildung 74: Soll-/Ist-Aufwand der wichtigsten User Stories

7.2.2 PRODUCT BACKLOG

Der Product Backlog ist im Kapitel 1.5.2.4 vorzufinden.

7.2.3 SPRINT BACKLOGS UND AUFWAND

Tabelle 46: Sprint #3 Aufwand (19.03.12-01.04.12)

User Stories	Soll-Zeit (h)	Ist-Zeit (h)
US 1.1: Existierender Benutzer anmelden	10	
US 2.1: Projekt mit Default-Locale erstellen	10	
US 2.2: Project Manager kann seine Projekte löschen	3	
US 6.1: Text übersetzen und bearbeiten (works)	10 (total 20)	
US 7.1: Developer kann per Web-GUI einen Key erfassen (works)	6 (total 12)	

Tabelle 47: Sprint #4 Aufwand (02.04.12-15.04.12)

User Stories	Soll-Zeit (h)	Ist-Zeit (h)
US 3.1: Project Manager kann Benutzer an Projekt zuordnen	12	
US 3.3: Project Manager kann Rolle an Benutzer vergeben	4	
US 6.1: Text übersetzen und bearbeiten (done)	10 (total 20)	
US 7.1: Developer kann per Web-GUI einen Key erfassen (done)	6 (total 12)	
US 7.2: Texter kann per Web-GUI den Defaulttext bearbeiten	8	
US 10.1: Alle Entries einer Locale anzeigen	4	
US 10.4: Entry-Weiterleitung für Translator, Texter, Reviewer	4	
US 13.3: Entries gemäss Rolle anzeigen	10	

Tabelle 48: Sprint #5 Aufwand (16.04.12-29.04.12)

User Stories	Soll-Zeit (h)	Ist-Zeit (h)
US 7.3: Key umbenennen	8	
US 7.4: Key löschen	5	
US 9.1: Java-Property-Files per Web-GUI exportieren	10	
US 10.2: Reviewer kann Übersetzungen korrigieren	4	
US 11.1: Reviewer kann Übersetzungen freigeben	4	
US 13.2: Text filtern	3	
US 13.6: Keys manuell zu Gruppen hinzufügen	20	

Tabelle 49: Sprint #6 Aufwand (30.04.12-13.05.12)

User Stories	Soll-Zeit (h)	Ist-Zeit (h)
US 1.2: Fehlermeldung bei erfolgsloser Anmeldung anzeigen	3	
US 2.3: Locale einem Projekt hinzufügen	8	
US 4.2: Anzahl Übersetzungen aller Locales anzeigen	5	
US 9.6: Rest-Webservice erstellen	20	
US 10.3: Kommentar bei reject/reopen erfassen können	15	
US 12.1: Neuer Benutzer kann Account erstellen	6	
US 13.7: Gruppierte Keys exportieren können	12	

7.3 ZEITAUFWAND

7.3.1 GELEISTETER AUFWAND

Gemäss Vorschrift sollten 30 Stunden pro ECTS Punkt investiert werden. Da diese Studienarbeit acht ECTS Punkte gibt, müssten folglich pro Person 240 Stunden geleistet werden. Nachfolgende Diagramme zeigen den geforderten und effektiven Stundenaufwand.

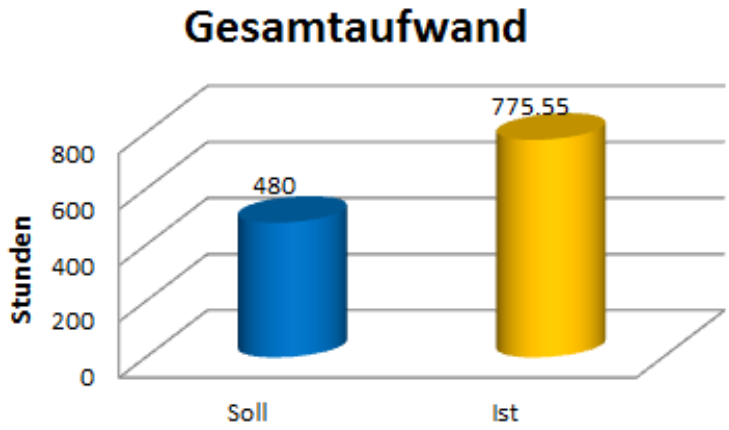


Abbildung 75: Gesamtaufwand

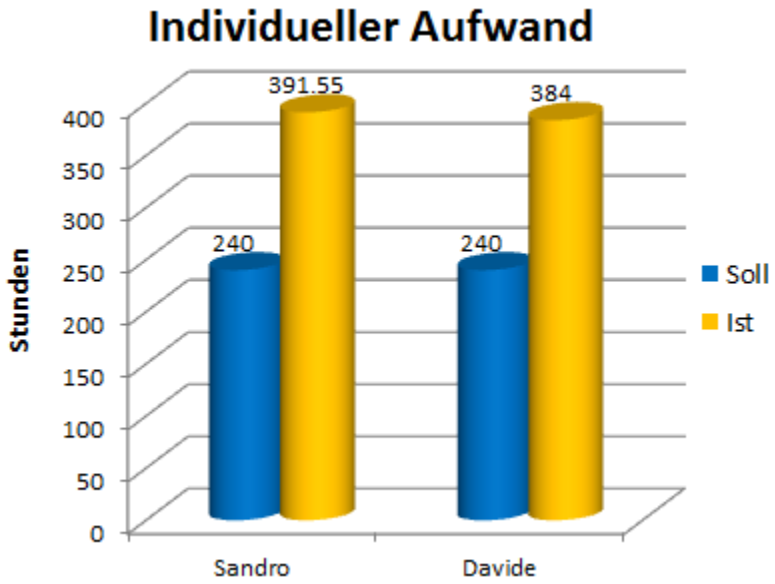


Abbildung 76: Individueller Aufwand

7.3.2 ARBEITSVERTEILUNG

Die nachfolgenden beiden Diagramme zeigen zum einen in welche Tätigkeiten Zeit investiert wurde und zum anderen auch zu welchem Zeitpunkt.

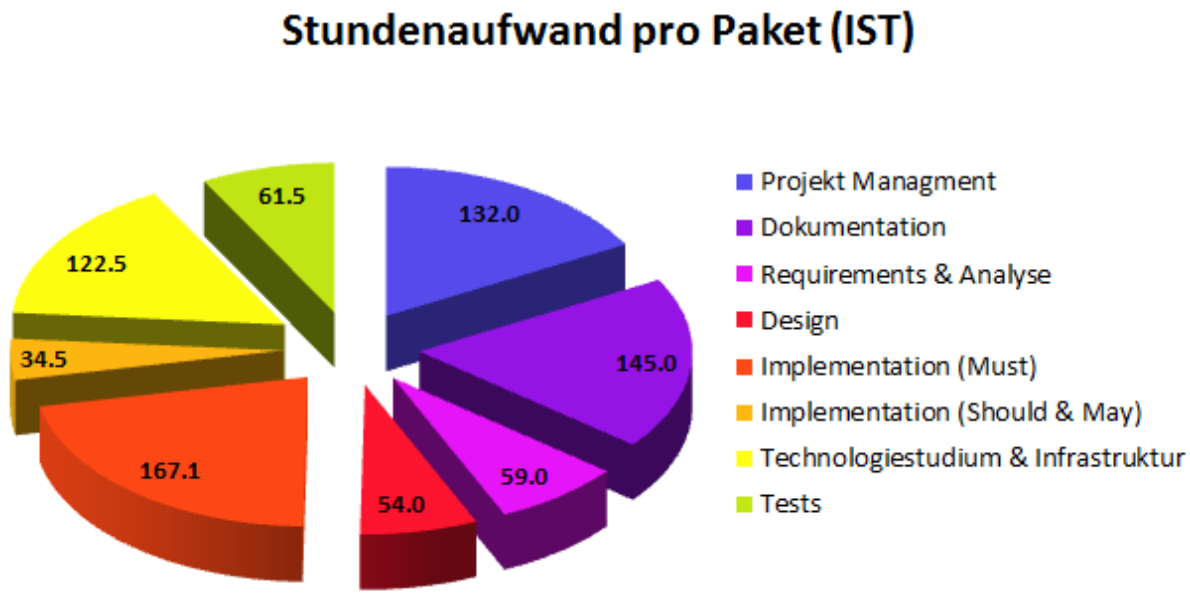


Abbildung 77: Stundenaufwand pro Paket (IST)

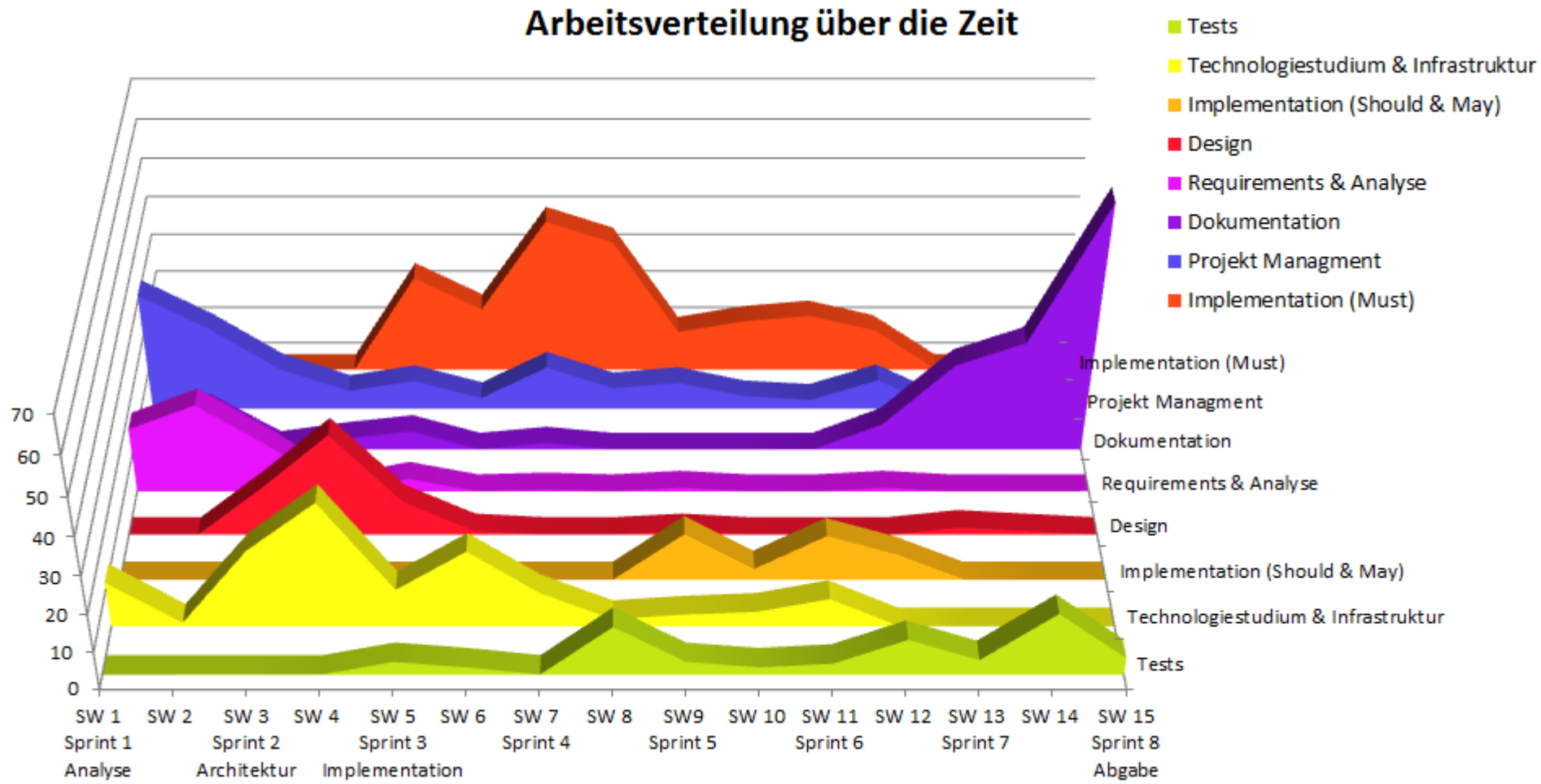


Abbildung 78: Arbeitsverteilung über die Zeit

8 PERSÖNLICHE BERICHTE

8.1 DAVIDE GIAMPA

Für mich persönlich war diese Studienarbeit eine Supererfahrung. Einerseits konnte ich viele neue Technologien kennenlernen und direkt für die Arbeit einsetzen, andererseits konnte ich erste Erfahrung mit echten Industriepartnern sammeln. Ich habe noch nie eine Webapplikation durch alle Tiers entwickelt, umso mehr Spass hat es gemacht.

Zu Beginn der Arbeit waren mir die Begriffe Spring, Maven und Hibernate nicht geläufig. Sandro Felicioni und ich haben uns entschieden, diese Technologien genauer unter die Lupe zu nehmen und zu schauen, ob wir einen Mehrwert gewinnen konnten. Für Hibernate mussten wir am Anfang viel Zeit investieren, bis wir lauffähige Queries hatten. Die Features von Spring wie z.B. Dependency Injection und Unterstützung für Testing haben vieles erleichtert. Mit Maven konnten wir automatische Abhängigkeiten zu Libraries verwalten und so ein Mitschleppen von Libraries verhindern.
Betreffend Datenbank waren wir uns am Anfang uneinig, ob wir Oracle oder Mysql einsetzen sollten. Da Oracle aber für dieses Projekt fast zu umfangreich hätte sein können, einigten wir uns auf Mysql. Mit der Mysl Workbench, ein Managing-Tool von Oracle, konnten wir sehr effizient Datenbank-Tasks erledigen (Queries testen, Tabellen bearbeiten, Datenmodell generieren, Dumps importieren und exportieren usw.). Der Aufwand in die erwähnten Technologien hat sich im Grossen und Ganzen bewährt, da wir eleganten und knappen Code schreiben konnten.

Die Zusammenarbeit mit dem Kunden verlief reibungslos. Die gewünschten Features konnten wir problemlos umsetzen und den Kunden so zufrieden stellen.

Die Reviews mit dem Betreuer waren sehr lehrreich, Herr Keller gab uns viele Tipps, die in das Projekt miteingeflossen sind. Was ich das nächste Mal anders machen würde, ist schon von Anfang an mehr Doku zu schreiben, da wir in den letzten zwei Wochen auf das Gaspedal drücken mussten.
Zum Schluss möchte ich mich noch bei Sandro Felicioni und Herrn Keller für die tolle Zusammenarbeit bedanken.

8.2 SANDRO FELICIONI

15 Wochen Studienarbeit sind nun vorbei und es ist ein guter Augenblick um nochmals auf das geleistete zurückzuschauen. Da die Aufgabenstellung ziemlich offen gestellt war, wussten wir anfangs nicht recht was uns erwarten würde und ob wir den Anforderungen gerecht werden konnten. Es stellte sich aber schnell heraus, dass die Ziele zwar hoch waren, jedoch mit den richtigen Technologien und dem nötigen Einsatz durchaus zu erreichen waren.

Obschon mir einige der eingesetzten Technologien wie Java, JSF und MySQL bekannt waren, hatte ich bis anhin nur geringe Erfahrungen mit Primefaces, Spring und Hibernate. Ich habe mich im Verlauf der Arbeit oft gefragt ob wir damit die richtigen Technologieentscheidungen getroffen haben, denn die Integration dieser Frameworks hat uns viel Zeit gekostet und insbesondere die Lernkurve bei Hibernate war ziemlich steil. Rückblickend kann ich aber sagen, dass ich froh bin über die getroffenen Technologieentscheidungen und würde sogar behaupten, dass wir gerade deswegen so viel in so kurzer Zeit erreicht haben. Vor allem Spring hat sich als extrem wertvoll erwiesen, und hat uns hervorragend in den Bereichen Dependency Injection, Testing und Anbindung an die Datenbank unterstützt. Die gesammelten Erfahrungen mit Spring erachte ich als sehr wertvoll und bin überzeugt, dass ich sie auch bei meinen nächsten Web Projekten wieder gewinnbringend einsetzen kann.

Abschliessend möchte ich mich noch bei unserem Betreuer Prof. Daniel Keller für die gute Zusammenarbeit bedanken. Er stand uns während der gesamten Arbeit zur Seite und hat uns bei Fragen und Unklarheiten konstruktiv unterstützt. Ebenfalls möchte ich mich bei meinem Team Kollegen Davide Giampa für seinen grossen Einsatz bedanken. Ich hätte mir für diese Arbeit keinen besseren Partner vorstellen können.

9 VERZEICHNISSE

9.1 ABBILDUNGSVERZEICHNIS

Abbildung 1: Ist- und Soll-Situation des Übersetzungsprozesses IV
Abbildung 2: Use Case Diagramm 5
Abbildung 3: Workflow Aktivitätsdiagramm 13
Abbildung 4: Zustandsdiagramm 14
Abbildung 5: Paperprototyp Startpage 15
Abbildung 6: Paperprototyp Startpage – Login Dialog 15
Abbildung 7: Paperprototyp Project Dashboard 16
Abbildung 8: Paperprototyp Project Dashboard – Add Project Dialog 16
Abbildung 9: Paperprototyp Project Info 17
Abbildung 10: Paperprototyp Project Info – Invite User Dialog 17
Abbildung 11: Paperprototyp Project Translations 18
Abbildung 12: Startpage 19
Abbildung 13: Startpage - Login Dialog 19
Abbildung 14: Startpage - Register Dialog 20
Abbildung 15: Project Dashboard 21
Abbildung 16: Project Dashboard - Add Project Dialog 22
Abbildung 17: Project Info 23
Abbildung 18: Project Info – Benutzer verwalten 23
Abbildung 19: Project Info – Invite User Dialog 24
Abbildung 20: Project Info – Locales verwalten 24
Abbildung 21: Project Info – Add Locale Dialog 24
Abbildung 22: Project Info – REST Service Information 25
Abbildung 23: Project Structure 26
Abbildung 24: Project Structure – Kontextmenü Root-Gruppe 26
Abbildung 25: Project Structure – Kontextmenü für Gruppe / Subgruppe 27
Abbildung 26: Project Structure – Kontextmenü für Key 27
Abbildung 27: Project Structure – Add Group Dialog 27
Abbildung 28: Project Translations 28
Abbildung 29: Project Translations – Filter-Sektion 28
Abbildung 30: Project Translations – Text-Filter 29
Abbildung 31: Project Translations - Hierarchische Gruppierung 29
Abbildung 32: Project Translations - Logische Gruppierung 29
Abbildung 33: Project Translations – Translations Sektion 29
Abbildung 34: Project Translations – Add Key Dialog 30
Abbildung 35: Project Translations – Export Translations Dialog 31
Abbildung 36: Project Translations – Reopen Dialog 31
Abbildung 37: Project Translations – Reject Dialog 31
Abbildung 38: Spring Modul Diagramm 34
Abbildung 39: SLF4J's Lösungsansatz für mehrere Logging Frameworks 36
Abbildung 40: Abstraktes Package - Diagramm 37
Abbildung 41: Reines Package - Diagramm 38
Abbildung 42: Klasseninteraktion von Project Dashboard 39
Abbildung 43: Interaktionsdiagramm Key erfassen 40
Abbildung 44: Physische Architektur 41
Abbildung 45: Klassendiagramm Rights 42

Abbildung 46: Klassendiagramm Filter Decorator 44
Abbildung 47: Klassendiagramm Formatklassen 45
Abbildung 48: Beispiel der REST-Service Implementation 46
Abbildung 49: Datenmodell 49
Abbildung 50: PMD Trend 76
Abbildung 51: PMD Trend 77
Abbildung 52: comparteTo-Methode von Entry 77
Abbildung 53: generateToken-Methode von Generator 78
Abbildung 54: Findbugs Trend 78
Abbildung 55: Trend der JUnit Testergebnisse 79
Abbildung 56: Trend der Testabdeckung 79
Abbildung 57: Testabdeckung des ganzen Projekts 80
Abbildung 58: Testabdeckung der einzelnen Packages 80
Abbildung 59: Erfasste Design Levels in Structure 101 81
Abbildung 60: Design Level 3 81
Abbildung 61: Design Level 2 82
Abbildung 62: Design Level 1 82
Abbildung 63: Leave Package Level 83
Abbildung 64: Tangle Packages rights / domain 84
Abbildung 65: Shell – SVN command um ein Repository auszuchecken 85
Abbildung 66: Shell – Maven command um Eclipse Projekt Files zu generieren 85
Abbildung 67: Shell – Maven command um Eclipse Projekt Files zu lsöchen 86
Abbildung 68: Shell – Maven command um Eclipse Projekt Files zu generieren 86
Abbildung 69: Shell – Maven command um Projekt zu builden 86
Abbildung 70: Import Dump mit Mysql Workbench 88
Abbildung 71: Portrait von Davide Giampa 90
Abbildung 72: Portrait von Sandro Felicioni 90
Abbildung 73: Projektplan 91
Abbildung 74: Soll-/Ist-Aufwand der wichtigsten User Stories 93
Abbildung 75: Gesamtaufwand 95
Abbildung 76: Individueller Aufwand 95
Abbildung 77: Stundenaufwand pro Paket (IST) 96
Abbildung 78: Arbeitsverteilung über die Zeit 97

9.2 TABELLENVERZEICHNIS

Tabelle 1: Bestehende Akteure 4
Tabelle 2: Neue Akteure 4
Tabelle 3: Farb-Legende zu den Use Cases 6
Tabelle 4: Allgemeine Use Cases 6
Tabelle 5: Use Cases für Project Manager 6
Tabelle 6: Use Cases für Texter und Translator 6
Tabelle 7: Use Cases für Reviewer 7
Tabelle 8: Use Cases für Developer 8
Tabelle 9: Legende zum Product Backlog 9
Tabelle 10: User Stories zu Anmelden 9
Tabelle 11: User Stories zu Projekt verwalten 9
Tabelle 12: User Stories zu Benutzer verwalten 9

Tabelle 13: User Stories zu Projektstatus überprüfen 9
Tabelle 14: User Stories zu Projektstand publizieren 10
Tabelle 15: User Stories zu Text übersetzen 10
Tabelle 16: User Stories zu Key erfassen 10
Tabelle 17: User Stories zu Übersetzung importieren 10
Tabelle 18: User Stories zu Übersetzung exportieren 11
Tabelle 19: User Stories zu Übersetzung überprüfen 11
Tabelle 20: User Stories zu Übersetzung freigeben 11
Tabelle 21: User Stories zu Account verwalten 11
Tabelle 22: User Stories zu allgemeinen Features 12
Tabelle 23: Project Dashboard Aktionen 21
Tabelle 24: Project Translations Aktionen 30
Tabelle 25: Logische Architektur: Abstrakte Packages 37
Tabelle 26: JAX-RS Annotations 47
Tabelle 27: Create Key REST-Service Parameter 48
Tabelle 28: Export Keys REST-Service Parameter 48
Tabelle 29: Beschreibung der wichtigsten Entitäten und Beziehungen 50
Tabelle 30: Testspezifikation Start Page 53
Tabelle 31: Testprotokoll Start Page 54
Tabelle 32: Testspezifikation Project Dashboard 55
Tabelle 33: Testprotokoll Project Dashboard 56
Tabelle 34: Testspezifikation Project Info 57
Tabelle 35: Testprotokoll Project Info 58
Tabelle 36: Testspezifikation Project Structure 59
Tabelle 37: Testprotokoll Project Structure 60
Tabelle 38: Testspezifikation Project Translations: Allgemein 61
Tabelle 39: Testprotokoll Project Translations: Allgemein 62
Tabelle 40: Testspezifikation Project Translations: Workflow 62
Tabelle 41: Testprotokoll Project Translations: Workflow 65
Tabelle 42: Testspezifikation REST-Schnittstelle 68
Tabelle 43: Testprotokoll REST-Schnittstelle 69
Tabelle 44: SQL-Dumps für Centralator 87
Tabelle 45: Verantwortung und Aufgaben 90
Tabelle 46: Sprint #3 Aufwand (19.03.12-01.04.12) 94
Tabelle 47: Sprint #4 Aufwand (02.04.12-15.04.12) 94
Tabelle 48: Sprint #5 Aufwand (16.04.12-29.04.12) 94
Tabelle 49: Sprint #6 Aufwand (30.04.12-13.05.12) 94

9.3 LITERATURVERZEICHNIS

[Gamma95] Gamma, E.; Helm, R.; Johnson, R.; Vlissides, J. 1995 Design Patterns: Elements of Reusable Object-Oriented Software. Addison-Wesley Longman Publishing Co., Inc.

[Larman11] Larman Craig, "Applying UML and Patterns", 3. Auflage, 2011

[Url01] Referenzplattform LocaleApp
http://www.localeapp.com
letzter Zugriff 31.05.2012

[Url02] Scrum Product Backlog
http://de.wikipedia.org/wiki/Scrum#Product_Backlog
letzter Zugriff 10.05.2012

[Url03] Google Translate
http://translate.google.com/
letzter Zugriff 20.04.2012

[Url04] Wireframe Software Axure
http://www.axure.com
letzter Zugriff 15.03.2012

[Url05] JSF Komponentbibliothek Primefaces
http://www.primefaces.org
letzter Zugriff 25.05.2012

[Url06] Spring Framework
http://www.springsource.org/
letzter Zugriff 25.05.2012

[Url07] Spring Aspected Oriented Programming
http://static.springsource.org/spring/docs/3.1.x/spring-framework-reference/html/aop.html#aop-introduction
letzter Zugriff 25.05.2012

[Url08] Spring Depedency Injection
http://static.springsource.org/spring/docs/3.1.x/spring-framework-reference/html/beans.html#beans-factory-collaborators
letzter Zugriff 25.05.2012

[Url09] Spring Expression Language
http://static.springsource.org/spring/docs/3.1.x/spring-framework-reference/html/expressions.html,
letzter Zugriff 26.05.2012

[Url10] Spring Transaction Management
http://static.springsource.org/spring/docs/3.1.x/spring-framework-reference/html/testing.html#testing-tx
letzter Zugriff 31.05.2012

[Url11] Hibernate Documentation
http://docs.jboss.org/hibernate/orm/4.1/manual/en-US/html/,
letzter Zugriff 05.05.2012

[Url12] Mysql Workbench
http://www.mysql.de/products/workbench/
letzter Zugriff 31.05.2012

[Url13] Simple Logging Facade for Java
http://www.slf4j.org/
letzter Zugriff 31.05.2012

[Url14] Jersey JaX-RS Reference Implementation User Guide
http://jersey.java.net/nonav/documentation/latest/user-guide.html#d4e8
letzter Zugriff 31.05.2012

[Url15] JAX-RS Annotations
http://docs.oracle.com/javaee/6/tutorial/doc/gilik.html
letzter Zugriff 30.05.2012

[Url16] Primary & Secondary Actions
http://www.lukew.com/ff/entry.asp?571
letzter Zugriff 20.05.2012

[Url17] PMD Rules
http://pmd.sourceforge.net/rules/index.html
letzter Zugriff 30.05.2012

[Url18] Findbugs Description
Fehler! Hyperlink-Referenz ungültig.http://findbugs.sourceforge.net/bugDescriptions.html
letzter Zugriff 30.05.2012

[Url19] Cobertura Test Coverage
http://cobertura.sourceforge.net/
letzter Zugriff 24.05.2012

[Url20] Structure 101
http://www.headwaysoftware.com/
letzter Zugriff 24.05.2012

[Url21] Mysql Installer für Windows
http://dev.mysql.com/tech-resources/articles/mysql-installer-for-windows.html,
letzter Zugriff 23.03.2012

10 INHALT CD

Ordner	Inhalt
Bericht	Bericht in Word- und PDF-Format
Centralator-Eclipse-Projekt	Eclipse-Projekt als Zip-Datei
Centralator-WAR	Webarchiv des Projekts
DB-Dumps	Datenbank-Dumps für die Erstellung dieser

11 AUFGABENSTELLUNG

Die Internationalisierung von Programmen läuft immer etwa nach dem folgenden Muster ab: Die Programmierer nutzen den Mechanismus ihrer Programmierumgebung, um Anzeige-Texte vor dem Ausgeben zu übersetzen, z.B. display_error(translate("server_unreachable")); wobei je nach gewählter Sprache die entsprechend übersetzte Fehlermeldung ausgegeben wird. Dazu gibt es irgendwo Dateien oder DB-Einträge, die aus dem 'key' mit dem Wert "server_unreachable" die Texte "Der gewählte Server ist zur Zeit nicht erreichbar.", "The selected server is not reachable at the moment.", "Votre serveur n'est pas accessible au moment." anzeigen - oder was auch immer die Übersetzer/innen liefern.

Normalerweise werden zu Beginn die 'key' Strings und die zugehörigen Texte in der sogenannten Originalsprache (meist Englisch) festgelegt. Dann werden die Originaltexte zu Übersetzungsbüros geschickt, wo sie übersetzt und wieder zurückgeschickt werden. Anschliessend werden die Texte von Software Engineers geprüft, ob deren Textlänge in der Anzeige der Masken und Reports auch Platz hat (häufig ist die deutsche Übersetzung 20-30% länger als der englische Text). Danach werden die Texte durch firmeninterne Personen (oft Personal in den jeweiligen Ländervertretungen) geprüft und abgenommen. Erst dann kann die Applikation auf den Markt gebracht werden.

Für die Entwicklungsumgebung Ruby on Rails gibt es eine sehr einfach zu bedienende Web-App, die das Übersetzen von Texten, die durch ein Ruby-Programm angezeigt werden, erleichtert: http://www.localeapp.com

Aufgabenstellung: es soll eine Web-Applikation geschrieben werden, die sich an die oben erwähnte LocaleApp anlehnt, die aber unabhängig von der Computer-Sprache den Übersetzungs-Workflow erleichtert. Es sollen insbesondere die Sprachen Java, C# und PHP durch die Applikation unterstützt werden. Dabei sollen die Texte in einer Datenbank gespeichert werden, damit ganz einfach die verschiedenen Sprachen unterstützt werden können. Ebenso soll durch Endnutzer ein einfacher Workflow definiert werden können, der durch die Applikation umgesetzt wird.

Die Applikation selbst sollte in Java/JSF oder C# .net programmiert werden. Falls überzeugende Gründe vorliegen, kann die Applikation auch anders implementiert werden. Die Sprachen, die bei Anzeige und Übersetzung unterstützt werden sollen sind: europäische Sprachen, inkl. solcher mit kyrillischer Schrift (unicode; aber nicht Chinesisch, Japanisch, auch keine rechts-links Sprachen wie Arabisch).

Daniel Keller

12 EIGENSTÄNDIGKEITSERKLÄRUNG

Wir, Sandro Felicioni und Davide Giampa erklären hiermit, dass die vorliegende Arbeit selber und ohne fremde Hilfe durchgeführt wurde, ausser derjenigen, welche explizit in der Aufgabenstellung erwähnt ist oder mit dem Betreuer schriftlich vereinbart wurde. Wir haben sämtliche verwendeten Quellen erwähnt und gemäss gängigen wissenschaftlichen Zitierregeln korrekt angegeben.

Rapperswil-Jona, 1. Juni 2012

Sandro Felicioni

Davide Giampa