# Arbeitsplan

(überarbeitete Version)

## 1 Projektvision

In den letzten Jahren vollzog sich ein rapider Wandel hinsichtlich linguistischer Ressourcen im Semantic Web. Während Informationen bereits maschinenlesbar sind, ist dies im Bereich der Morphologie weitestgehend noch nicht erreicht. Morphologische Daten sind bisher entweder nicht vorhanden oder liegen verborgen in semi-strukturierten Dokumenten. Das bedeutet, dass es derzeit keine geeignete Plattform gibt, die von Linguisten gemeinsam genutzt werden könnte, um morphologische Daten zu sammeln und zu kategorisieren.
Ziel dieses Projektes ist es, eine derartige Online-Plattform in Form einer Web-Applikation zu konzipieren und umzusetzen. Diese soll Sprachwissenschaftlern die Möglichkeit bieten, ihre Forschungsergebnisse festzuhalten, wiederzuverwenden und anderen zur Verfügung zu stellen. Gleichsam sollen auch sie von der Vorarbeit anderer profitieren können, indem sie deren gesammelte Informationen auswerten.
Die große Herausforderung liegt darin, die Oberfläche so intuitiv wie möglich zu gestalten, denn Linguisten besitzen meist keine bzw. nur wenig fundierte informatische Kenntnisse. Die Bedienung sollte ohne Einarbeitung offensichtlich sein und keine Spielräume für Verwirrung bieten.

## 2 Voraussetzungen

Die erfolgreiche Fertigstellung des Projektes nach Spezifikationen der Projektvision unterliegt bestimmten Voraussetzungen, welche im folgenden konkretisiert werden. Die Umsetzung des Projekts erfolgt vollständig in *JAVA*.
Zunächst muss ein Web-Application Server eingerichtet werden, dafür verwenden wir *Tomcat*. Desweiteren ist eine Weboberfläche zu erstellen. Diese erzeugen wir durch Nutzung des *Vaadin Frameworks*.
Es wird eine für jeden Nutzer frei zugängliche Filterfunktion eingerichtet. Auf diese Weise kann die Seite als reines virtuelles Nachschlagewerk genutzt werden. Um Datenerstellung oder Datenmanipulationen unbekannter Herkunft zu vermeiden und Partizipation der Nutzer zu ermöglichen, muss sich der Benutzer, bevor er die Rechte zur Erstellung oder Änderung von Dateneinträgen hat, einen User-Account anlegen. Die User-Verwaltung erfolgt mittels des *Shiro Frameworks*. Da auch registrierte Benutzer fehlerhafte Einträge erstellen oder falsche Korrekturen ausführen können, werden Fehlermeldungen in Form eines Kommentars realisiert. Gleichzeitig können Korrekturvorschläge gemacht werden. Diese sind von einem Administrator-Account zu bewerten und gegebenenfalls zu akzeptieren oder zu verwerfen. Ein solches Verfahren ist notwendig, um die permanente Korrektheit der Dateneinträge sicherzustellen.

Es werden zu jeder Modifikation entsprechende Provenienz-Daten erstellt und protokolliert. Die Verwaltung der User-Accounts und eine generelle Überwachung des Systems, sollen ebenfalls durch die Einrichtung eines Administrator-Account geregelt werden.
Aus technischer Sicht sind Interaktionen eines Benutzers, die auf Ebene der Weboberfläche erfolgen, mittels verschiedener Technologien letztendlich zu deskriptiven und insbesondere manipulativen Anfragen an die RDF-basierte Morphemdatenbank umzuwandeln, welche dem Benutzer verborgen bleiben und über die er selbst keinerlei Wissen verfügen muss. Zur Realisierung, Verwaltung und Abfrage der Morphemdatenbank verwenden wir das *Jena Framework.* Die genaue Vorgehensweise wird in der Designübersicht und Funktionalität explizit beschrieben und grafisch dargelegt. Die Erhaltung der Konsistenz der Datenbank, sowie die Vermeidung von der Aufnahme von Duplikaten wird vom Datenbanksystem selbst sichergestellt. Im letzteren Fall wird der Benutzer über die Existenz der schon vorhandenen Datensätze informiert, damit dieser nicht von einem Fehler bei der Informationsübertragung ausgeht.
Neben den technischen Voraussetzungen ist die intuitive Bedienung der Weboberfläche von zentraler Bedeutung und deshalb eine wichtige Voraussetzung. Zur weitestmöglich objektiven Gewährleistung haben wir im Qualitätssicherungskonzept spezielle Maßnahmen integriert, welche dort nachzulesen sind.
Da wir uns in der Gruppe wöchentlich treffen, an die Scrum-Methodik halten, alle Dokumente transparent für alle Mitglieder bearbeiten und einen Gruppen-internen Chatserver für weitere Absprachen nutzen, erreichen wir im Team eine Angleichung des Wissensstandes über den Fortschritt, die Vorgehensweisen und Methoden.

# 3 Designübersicht und Funktionalität

Das Projekt wird als Web-Applikation umgesetzt. Das Front-end besteht aus einem komponentenbasiertem User-Interface (UI), welches in englischer Sprache umgesetzt wird. Das Back-end umfasst eine Application-Server, der die Services der Web-Applikation zugänglich macht. Die Komponenten des UI setzen den Zugriff auf die Funktionalitäten um. Die Web-Applikation verarbeitet, die UI-Interaktionen des Users und aktualisiert das UI entsprechend.

## 3.1 Designübersicht

Es werden drei Rollen implementiert: Gast, User, Admin. Jede Rolle verfügt über unterschiedliche Rechte bzgl. des Zugriffs auf den Datenbestand. Dabei sind Gastrechte $\subset$ Userrechte $\subset$ Administratorrechte. Während Gäste, die Daten im Wesentlichen nur darstellen können, verfügen die User und der Administrator über die Möglichkeit die Daten zu ergänzen. Da die Umsetzung des Projektes vollständig in *JAVA* erfolgt, müssen die Komponenten des UI auf dem Server gerendert werden. Die Business Logic umfasst den Zugriff auf die Datenbank, die Abfrage der Datenbank, die Datenbank selbst und das Persistiren der Datenbank. Bei Zugriffen und Abfragen wird der Zustand von bestimmten UI Komponenten in

SPARQL-Abfragen übersetzt und die Ergebnisse an das UI zurückgegeben. Dieses nutzt diese Informationen zur Aktualisierung und zum Rendern der Client-side.

## 3.2 Funktionalität

Muss-Funktionalitäten müssen unbedingt im fertigen Produkt implementiert sein und haben eine höhere Priorität als Kann-Funktionalitäten. Bei der Implementierung der Muss-Funktionalitäten ist darauf zu achten, dass die Erweiterbarkeit des Quellcodes zur Implementierung der Kann-Funktionalitäten gewährleistet ist.

### 3.2.1 Muss-Funktionalitäten

**/LF110/** *Geschäftsprozess:* Datenansicht
*Akteur:* Gast, User, Administrator
*Beschreibung:* Die Einträge der Datenbank werden in einer übersichtlichen Darstellung präsentiert. Dabei soll die zugrundeliegende RDF weitgehend verborgen werden.

**/LF120/** *Geschäftsprozess:* Datenbrowsing
*Akteur:* Gast, User, Administrator
*Beschreibung:* Ermöglicht die Navigation durch die Datenbank durch Anklicken von Einträgen und Verfolgen von Links.

**/LF130/** *Geschäftsprozess:* Daten filtern
*Akteur:* Gast, User, Administrator
*Beschreibung:* Ermöglicht, die Einschränkung der angezeigten Daten (vgl. /LF110) durch setzen von Filtern.

**/LF140/** *Geschäftsprozess:* Datenlücken anzeigen
*Akteur:* Gast, User, Administrator
*Beschreibung:* Darstellung von noch nicht zugewiesenen Eigenschaften eines Eintrags

**/LF150/** *Geschäftsprozess:* Neuen Eintrag erstellen
*Akteur:* User, Administrator
*Beschreibung:* Ermöglicht das Erstellen eines neuen Eintrags mittels eines Webformulars

**/LF160/** *Geschäftsprozess:* Bestehenden Eintrag ergänzen
*Akteur:* User, Administrator
*Beschreibung:* Ermöglicht das einfügen noch nicht gemachter Angaben eines bestehenden Eintrags mittels eines Webformulars

**/LF170/** *Geschäftsprozess:* Bestehenden Eintrag ändern
*Akteur:* Administrator
*Beschreibung:* Ermöglicht die Aktualisierung eines bestehenden Eintrags

**/LF180/** *Geschäftsprozess:* Fehler melden
*Akteur:* User, Administrator
*Beschreibung:* Ermöglicht die Meldung eines Fehlers durch erstellen eines Kommentars und informiert den Administrator über die Meldung

**/LF190/** *Geschäftsprozess:* Vorschlag erstellen
*Akteur:* User
*Beschreibung:* Ermöglicht das erstellen eines Datensatzes, der als Korrekturvorschlag an den Administrator geleitet wird

**/LF210/** *Geschäftsprozess:* Vorschlag annehmen/verwerfen
*Akteur:* Administrator
*Beschreibung:* Ermöglicht das Löschen oder die Übernahme des vorgeschlagenen Datensatzes (vgl. /LF190/) durch ändern des Datenbestandes (vgl. /LF170/)

**/LF310/** *Geschäftsprozess:* Registrierung
*Akteur:* Gast
*Beschreibung:* Ermöglicht das Erstellen eines neuen User-Accounts unter angabe der relevanten User-Daten

**/LF320/** *Geschäftsprozess:* Anmeldung
*Akteur:* User
*Beschreibung:* Ermöglicht die Authentifizierung und Anmeldung über die Weboberfläche mit Userrechten

**/LF330/** *Geschäftsprozess:* Anmeldung
*Akteur:* Administrator
*Beschreibung:* Ermöglicht die Authentifizierung und Anmeldung über die Weboberfläche mit Administratorrechten

**/LF340/** *Geschäftsprozess:* Account-Löschen
*Akteur:* User, Administrator
*Beschreibung:* Ermöglicht das Löschen von bestehenden User-Accounts durch den User selbst oder den Administrator

### 3.2.2 Kann-Funktionalitäten

**/KF110/** *Geschäftsprozess:* Freitextsuche
*Akteur:* Gast, User, Admin
*Beschreibung:* Ermöglicht das direkte Durchsuchen des Datenbestandes nach bestimmten Einträgen

**/KF120/** *Geschäftsprozess:* Erstellen neuer Rollen
*Akteur:* Administrator
*Beschreibung:* Ermöglicht das Erstellen neuer Rollen und den mit ihr assoziierten Rechten

**/KF130/** *Geschäftsprozess:* Erstellen von Inventories
*Akteur:* Admin, User
*Beschreibung:* Ermöglicht das Anlegen von Inventories.

**/KF140/** *Geschäftsprozess:* Erstellen von privaten User-Projekten
*Akteur:* User, Administrator
*Beschreibung:* Ermölgicht das Anlegen von privaten User-Inventories.

**/KF150/** *Geschäftsprozess:* Download von User-Projekt-Daten
*Akteur:* User, Administrator
*Beschreibung:* Ermöglicht das Erstellen und Speichern der User-Projekt-Daten.

**/KF160/** *Geschäftsprozess:* Protokollieren von Änderungen
*Akteur:* User, Administrator
*Beschreibung:* Bei jedem Hinzufügen oder Verändern eines Datensatzes (vgl. /LF150/ - /LF190/) werden automatisch die Art der Änderung und Provenienzdaten des Ausführenden gespeichert

**/KF170/** *Geschäftsprozess:* Provenienzdatenansicht
*Akteur:* Gast, User, Administrator
*Beschreibung:* Die in den Einträgen der Datenbank protokollierten Provenienzdaten (vgl. /KF160/) werden in einer übersichtlichen Darstellung präsentiert. Dabei soll die zugrundeliegende RDF weitgehend verborgen werden.

**/KF180/** *Geschäftsprozess:* Verbindung zu Social Media Diensten
*Akteur:* Gast, User, Admin
*Beschreibung:* Implementiert die Interaktion der Web-Applikation mit diversen Social Media Diensten

**/KF190/** *Geschäftsprozess:* Vokabularansicht
*Akteur:* Gast, User, Administrator
*Beschreibung:* Ermöglicht das Einsehen des Vokabulars

**/KF210/** *Geschäftsprozess:* Unterstützung beim Hinzufügen, Ändern und Ergänzen von Daten
*Akteur:* User, Administrator
*Beschreibung:* Beim Hinzufügen, Ändern und Ergänzen von Daten werden dem Nutzer Informationen über zulässige Eingaben gegeben

# 4 Arbeitspakete

**0. Vorprojekt: (15%):** Im Vorprojekt werden grundlegende Funktionen der Software implementiert. Eine genauere Beschreibung befindet sich im folgenden Abschnitt.

**1. User-Interface: (20%):** Das UI bildet als Benutzerschnittstelle einen zentralen Bestandteil der Software. Es muss leicht verständlich und intuitiv bedienbar sein. Auch sollte es ein ansprechendes Design besitzen um das Arbeiten mit der Software angenehmer zu machen. Das UI wird durch *Vaadin* implementiert.

**2. Schnittstelle User Interface - Buisiness Logic: (15%):** Die Implementierung der Schnittstelle zwischen dem UI und der Buisiness Logic erfolgt, wenn keine einfache Integration möglich ist, durch selbstgeschriebene Klassen.

**3. Buisiness Logic: (25%):** Die Buisiness Logic bildet den eigentlichen funktionalen Kern der Software. Hier werden serverseitig die Verwalung und der User-Accounts und -Rechte, sowie alle weiteren Funktionen (deskriptiv, manipulativ) implementiert. Wir nutzen hierfür das *Jena Framework.* Als Webserver benutzen wir *Tomcat*. Die User-Accounts und Rechteverwaltung werden mithilfe des pache-Shiro-*ShiroFramework* implementiert.

**4. Schnittstelle Web-Applikation - Datenbank: (15%):** Um die Daten letztendlich als RDF-Daten zu Laden und zu Speichern verwenden wir das *Jena Framework*. Funktionen, die Integrität der Daten sicherstellen müssen hier implementiert werden. Auch ein Mechanismus, welcher über schon vorhandene Daten informiert muss implementiert werden.

**5. Übergabe der Software: (5%):** Um den Einstieg in das Arbeiten mit der Software zu erleichtern planen wir den Endbenutzern die Software kurz vorzustellen und etwaige Fragen zu klären.

**6. Handbuch: (5%):** Die Erstellung eines kleinen Benutzerhandbuchs, welches die Handhabe und die grundlegenden Funktionen der Software beschreibt ist geplant.

# 5 Vorprojekt

## Aufgabe

Im Vorprojekt sollen Kernfunktionalitäten der Software implementiert werden. Es soll ein reduziertes UI erstellt werden, über das mittels der Web-Applikation Daten angezeigt und gefiltert werden können.. Weniger Wert wird vorerst auf die clientseitige Oberfläche der

Web-Applikation gelegt, sowie auf die Rechtevergabe der verschiedenen Useraccounts. Auch weitere Funktionen wie Einfügen, Datenlücken finden usw. werden vorerst nicht implementiert.

### Begründung

Das erfolgreiche Einrichten der Datenbank, sowie die Integration der verschiedenen Frameworks sind grundlegende Wegweiser für den weiteren Verlauf und Erfolg des gesamten Projekts. Dafür scheint uns eine Beschäftigung mit den Frameworks *Jena* und *Vaadin* als äußerst wichtig. Durch das Sammeln praktischer Erfahrung über das Zusammenspiel aller Komponenten in kleinem Maßstab, wird sich die Gruppe das Know-How für das Hauptprojekt erschließen.

# 6 Glossar

## 6.1 Informatische Begriffe

#### Apache Tomcat [1]

Apache Tomcat ist ein Webserver und Webcontainer. Tomcat erlaubt es in java geschriebene Webanwendungen mittels Servlets bzw. Java Server Pages (jsp) auszuführen. Tomcat kann als eigenständiger Webserver betrieben werden oder in andere Webserver integriert werden.

#### Apache Shiro [2] [3]

Apache Shiro ist ein Java open-source Sicherheits-Framework. Mit Shiro können folgende Funktionen realisiert werden: Authentifizierung, Autorisation, Kryptographie und session management.

## 6.2 weitere informatische und linguistische Begriffe

siehe externes Dokument

## Quellen

[1] https://en.wikipedia.org/wiki/Apache_Tomcat (abgerufen am 31.01.2016)
[2] http://shiro.apache.org/ (abgerufen am 31.01.2016)
[3] https://en.wikipedia.org/wiki/Apache_Shiro (abgerufen am 31.01.2016)