Zürcher Hochschule
für Angewandte Wissenschaften

zh
aw School of
Management and Law

eUmzug Kanton Bern

Aufgabe 7: User Tasks & Web Stack

Modul Geschäftsprozessintegration im BSc Wirtschaftsinformatik

Autor
Björn Scheppler
scep@zhaw.ch

Herausgeber
Zürcher Hochschule für Angewandte Wissenschaften
School of Management and Law
Institut für Wirtschaftsinformatik (IWI)
Stadthausstrasse 14
CH-8041 Winterthur
www.zhaw.ch/iwi

Vorliegende Version
1.0.0 (5.11.2018)

Änderungshistorie

Version (Datum)	Wer	Was
1.0.0 (5.11.2018)	scep	Initiale Fassung

Hinweis: Aus Gründen der Lesbarkeit wurde im Text stets die männliche Form gewählt.

1 Zielsetzung

1. Am Beispiel mehrerer User Tasks und zugehöriger Formulare habt ihr **gelernt**, …
    a. … wie **Menschen** in Prozessapplikationen «**eingebunden**» werden,
    b. … wie Client-Applikationen (konkret Camunda Web Apps) **mit** der **Process Engine kommunizieren** (konkret REST via Camunda Forms SDK),
    c. … und wie bezüglich Design und Funktionalität **moderne Webapplikationen** erstellt werden, konkret unter Verwendung von AngularJS, Bootstrap und HTML 5.
2. Für die **Umzugsplattform** habt ihr dabei **alle User Tasks** mit den zugehörigen Formularen **implementiert** mit Ausnahme von ZahlungsDetailsErfassen.

2 Vorgeschlagenes Vorgehen

1. Besucht die **Kleinklasse** in der Semesterwoche 8 für die Kapitel 1 bis 3 der Starthilfe Teil 1 und schaut die dort verlinkten Videos für die Kapitel 4 und 5. Damit habt ihr …
    a. … bereits das Formular «**Identifikationsmerkale erfassen**» zu einem grossen Teil erstellt und das Formular «**Korrekturbedarf mitteilen**» komplett erstellt.
    b. … einen Weg kennen gelernt, um mit soapUI **effizient testen** zu können.
    c. … gemeinsam mit dem Stoff von Peter aus der Grossklasse der SW 8 das **praktische Rüstzeug**, für die Implementation der ersten Formulare (siehe nächster Punkt).
2. Implementiert die folgenden **User Tasks** entsprechend den Vorgaben aus der jeweiligen Element Documentation. Um die User Tasks zu testen, nutzt soapUI mit sinnvollen Testdaten.
    a. PersonErfassenUndIdentifizieren**AnSchalterVerwiesenWerden**
    b. WegzugsadresseErfassenUndPruefen**WegzugsadresseErfassen**
    c. ZuzugsinformationenErfassenUndPruefen**ZuzugsadresseErfassen** (Achtung: die politische Gemeinde noch nicht erfassen, da das Wissen hierzu noch fehlt)
    d. WeitereAngabenErfassen**WohnverhaeltnisErfassen**
    e. WeitereAngabenErfassen**KontaktangabenErfassen**
3. Besucht die **Kleinklasse** in Semesterwoche 9 und schaut die Videos für die Starthilfe Teil 2. Damit habt ihr …
    a. … das Formular «**Identifikationsmerkale erfassen**» fertig erstellt.
    b. … das Formular «**Versicherungs-Kartennummern erfassen**» fertig erstellt.
    c. … einige **Tipps zu AngularJS** erhalten, die für die weiteren Formulare relevant sind.
4. Stellt die Implementation des User Tasks ZuzugsinformationenErfassenUndPruefen\ **ZuzugsadresseErfassen** fertig.
5. Implementiert die folgenden **User Tasks** entsprechend den Vorgaben aus der jeweiligen Element Documentation. Um die User Tasks zu testen, nutzt soapUI mit sinnvollen Testdaten und für den ersten User Task auch das zur Verfügung gestellte PersonenRegister-Projekt.
    a. PersonErfassenUndIdentifizieren**MitumziehendePersonenAuswaehlen**
    b. ZuzugsinformationenErfassenUndPruefen**WohnungAnZuzugsadresseAuswaehlen**
    c. WeitereAngabenErfassen**UmzugsdatenErfassen**
    d. UmzugsmeldungErfassenUndBezahlen**AlleAngabenPruefen** (ohne Dokumente-Teil)

3 Abgabe

1. Die inhaltliche Abgabe ist ein Commit und Push im Github-Repository der Gruppe.
2. Die formale Abgabe erfolgt auf Moodle hier durch EINE Person im Team.
3. Dies erfolgt spätestens bis zum im Betreff der verlinkten Aufgabe aufgeführten Zeit.
4. Die Abgabe besteht darin, die Id des von uns zu korrigierenden Commits als Text einzufügen und auf «Abgabe einreichen» zu klicken

4 Bewertungskriterien

4.1 Kriterien

Maximal können **10 Punkte** erreicht werden, die sich auf die folgenden **Kriterien** aufteilen:

1. Teile aus **Kleinklassen-Unterricht-Starthilfen** korrekt implementiert (2 Punkte)
2. Implementation der User Tasks und Formulare bezüglich Anzahl, Bezeichnung, Reihenfolge und Typ der **HTML-Elemente** entsprechend den Anforderungen in der Element Documentation (3.5 Punkte)
3. Implementation bezüglich **Funktionalität inkl. Validierung** (AngularJS, JavaScript, usw.) gemäss den Anforderungen (3.5 Punkte)
4. Implementation bezüglich **Designs** (Bootstrap/CSS) und **Benutzerfreundlichkeit** (Platzhalter, Hilfetexte) gemäss den Anforderungen, respektive analog des in den Starthilfen gezeigten Formularen (1 Punkt)

4.2 Aufwandreduktion 4er-Gruppen

Die folgenden User Tasks sind nicht zu implementieren:

1. WeitereAngabenErfassen**WohnverhaeltnisErfassen**
2. WeitereAngabenErfassen**UmzugsdatenErfassen**

Im User Task UmzugsmeldungErfassenUndBezahlen**AlleAngabenPruefen** sind dementsprechend die Angaben, welche in den obigen zwei Formularen erfasst würden, ebenfalls nicht relevant.

4.3 0 Punkte

Keine Abgabe innerhalb der Abgabefrist auf Moodle und Github.

4.4 Punkteabzug

Für die folgenden formalen Fehler und fehlenden/falschen Dokumentationen gibt es die Abzüge auf die erreichte Punktzahl gemäss Kapitel 4.1. Diese Abzüge werden aufsummiert, wobei eine Gruppe nicht weniger als 0 Punkte erreichen kann.

1. **Keine fristgerechte Abgabe auf Moodle**, aber auf Github eine Lösung gepusht: 4 Punkte
2. **Auf Moodle formal eine falsche Abgabe** (z.B. keine vorhandene Commit-ID, etwas anderes als eine Commit-ID, usw.): 2 Punkte
3. HTML-Dateien sind nicht **bezeichnet** mit Id des User Tasks plus Suffix «Form»: 1 Punkt
4. HTML-Dateien haben keinen einleitenden Kommentar und JavaScript-Statements sowie sowie if-then-else- oder for-Blöcke enthlaten keinen oder keinen zutreffenden, selbst formulierten, deutschsprachigen **Zeilenkommentar**. Davon ausgenommen ist, wenn direkt untereinander mehrfach dasselbe gemacht wird (z.B. Auslesen von vier Prozessvariablen in lokale Variablen), dann kann dies in einem Kommentar zusammengefasst werden. Verletzung dieser Regel: maximal 4 Punkte.

5 Feedback

Spätestens am 29.11.2018 10:00 ist die **Musterlösung** unterhalb der Aufgabenstellung auf Moodle freigeschaltet.

Spätestens am 12.12.2018 20:00 sieht diejenige Person, welche die Aufgabe eingereicht hat, die erreichte Punktzahl pro Kriterium und insgesamt inklusive allfälliger Kommentare, wenn sie erneut auf die **Moodle-Aufgabe** klickt.

6 Fortgeschrittene Anforderungen (optional)

6.1 Mögliche zu implementierende Anforderungen

6.1.1 Dokumente hochladen – Variante cam-variable (einfach)

Ausgangslage: Falls es sich um einen Wegzug/Zuzug handelt, kann es sein, dass eine Gemeinde Dokumente für die umziehenden Personen verlangt (documentsExist ist true, wird in GetDocumentsDelegate ermittelt). Dann sollen diese Dokumente vom Benutzer im User Task «Dokumente hochladen» hochgeladen und im User Task «Alle Angaben prüfen» heruntergeladen werden können.

Anforderungen:

1. Siehe Element Documentation des User Tasks, allerdings vereinfacht.
2. Nur für den Meldepflichtigen sind die Dokumente hochzuladen und herunterzuladen (nicht für umziehende Personen).
3. Für FileUpload und FileDownload werden für jedes Dokument eine neue Prozessvariable generiert (z.B. «documentHeiratsurkunde»).
4. Die Implementation geschieht über cam-variable-type File. Dokumentation hier.

6.1.2 Dokumente hochladen – Variante documents (eher komplex)

Ausgangslage: siehe unter 6.1.1

Anforderungen:

1. Siehe Element Documentation des User Tasks «Dokumente hochladen» und «Alle Angaben prüfen».
2. Nur für den Meldepflichtigen sind die Dokumente hochzuladen und herunterzuladen (nicht für umziehende Personen).
3. Für FileUpload und FileDownload werden keine eigenen Prozessvariablen verwendet, sondern der Inhalt geht in ein MunicipalityDocumentUploadedFile-Objekt, welches eine Referenz auf ein MunicipalityDocumentRelationEntity-Objekt hat.
4. Für die Implementation wählt ihr entweder eure bevorzugte Art (wenn ihr da schon Erfahrung habt) oder verwendet die hier auf Moodle abgelegten Dateien, welche bewusst nur ein Minimum an Kommentaren haben. Eure Aufgabe ist es, diesen Code a) zu verstehen (unter Zuhilfenahme von Google & Co.) und die Zeilenkommentare entsprechend zu ergänzen und b) sinnvoll zu einem funktionierenden Ganzen zusammen zu bauen.

6.1.3 Dokumente hochladen – Variante komplex (vermutlich komplex)

Dito 6.1.2. Einziger, aber wesentlicher, Unterschied: Nicht bloss die Dokumente für den Meldepflichtigen sollen hoch- und heruntergeladen werden können, sondern auch diejenigen der mitumziehenden Personen. Implementation entweder über Multi-Instance-Activity (Vorbild: Grundversicherungsprüfung) oder von Frontend-Funktionalität wie im nächsten Kapitel «Grundversicherungsprüfung fortgeschritten» aufgeführt.

6.1.4 Grundversicherungsprüfung fortgeschritten (vermutlich eher komplex)

Ausgangslage: Momentan ist das Erfassen der Versichertenkartennummern eher umständlich, weil dies für jede einzelne Person separat durchgeführt werden muss.

Anforderung: Statt einer Multi-Instance-Activity soll ein einziger User Task "Grundversicherung prüfen" erstellt werden, welcher alle Personen und Versicherungs-Nummer-Formularfelder als Tabelle darstellt. Jedes Mal, wenn der Benutzer eine Versicherungsnummer erfasst und das Formularfeld verlässt, wird asynchron aus AngularJS heraus der REST-Service abgefragt des VeKa-Centers und das Resultat als Icon (Grüner Haken vs. Rotes Ausrufezeichen) angezeigt bei der jeweiligen Tabellenzeile als auch im Fall von Ausrufezeichen mit den Details als Text.

Implementations-Tipps: Für diese Anforderung werden Kenntnisse in REST vorausgesetzt und dass der VeKa-Auskunftsdienst bereits über REST zugänglich ist, was erst später im Semester der Fall ist. Daher kann diese Anforderung sicher erst später vollständig implementiert werden.

6.1.5 WebApplikation für VeKa-Center (vermutlich komplex)

Ausgangslage: Dies ist die Fortsetzung der fortgeschrittenen Anforderung 6.1.3 aus der Aufgabe 5. Dort hast Du im Idealfall die Persistierungs-Komponenten implementiert für die übergreifende Zielsetzung. Diese zur Wiederholung:

Aufgaben-übergreifende Zielsetzung: Eleganter wäre es, wenn Mitarbeitende des Kantons (und der Gemeinden) über eine separate Webapplikation (nicht Camunda) sowohl Stammdaten verwalten als auch den Status zu einer bestimmten Person abfragen könnten. Hierzu braucht es entsprechende Persistierungskomponenten und Controller-Klassen (Aufgabe 5), eine separate Webapplikation (diese Aufgabe) und noch eleganter REST-Controller (Aufgabe 9).

Anforderung: Für die folgenden im Idealfall in Aufgabe 5 implementierten Controller soll nun die zugehörige Webapplikation bereitgestellt werden, also eine Benutzeroberfläche, welche diese Controller-Funktionalität frontendseitig bereitstellt:

1. Verwaltung von **Dokumenten**: Hinzufügen eines neuen Dokuments, Suchen eines Dokuments über seinen Namen, Ausgeben einer Liste aller Dokumente, Umbenennen eines Dokuments, Löschen eines Dokuments sofern in keiner Beziehung mehr genutzt.
2. Dasselbe gilt für **Gemeinden** plus Verändern der drei Gebührenwerte für eine Gemeinde.
3. **Transaktions-Logbuch**: Den aktuellsten Status-Eintrag (*status* und *logTimeStamp*) für eine bestimmte Person ausgeben; Alle möglichen Status-Einträge zurückgeben alphabetisch sortiert; Liste aller Personen mit einem bestimmten Status zurückgeben, sortiert nach Zeit.

Implementations-Tipps: Bezüglich Wahl der Technologie für das Frontend seid ihr völlig offen. Da wir uns im Spring (Boot)-Kontext befinden und mit den Camunda Webapps schon etwas AngularJS kennen gelernt haben, können wir grundsätzlich zwei Varianten empfehlen:

1. Verwendung von **Thymeleaf** in Kombination mit Bootstrap. Tutorials: Spring Tutorial und o7planning Tutorial. Vorteile: Etwas einfacher zum Implementieren, kann sofort durchgeführt werden (noch ohne REST). Nachteile: weniger dynamisch und modern.
2. Verwendung von **AngularJS** (oder einem anderen Framework der Wahl wie z.B. vue.js). Tutorial z.B. Dzone-Artikel. Vorteile: Moderner, sauberere Trennung von Backend und Frontend (via REST). Nachteile: siehe Vorteile bei 1.

6.2 Abgabe

Die Regeln in Kapitel 5.2 der Hauptaufgabestellung einhalten. Falls die Abgabe nicht erst am 24.12.2018, sondern gleichzeitig mit den Basis-Anforderungen erfolgt, dann in der Abgabe auf Moodle (siehe Kapitel 3 oben) nicht bloss die Commit Id (oder Ids, falls auch VeKa betroffen) angeben, sondern auf einer zweiten Zeile auch der Hinweis: «Fortgeschrittene Anforderungen implementiert».