UNIVERSITÄT LEIPZIG

Wintersemester 2018/19
Softwaretechnikpraktikum

App zur Inventarisierung von Unternehmenswerten

QS-Konzept

Gruppe: ak18b

Betreuer: Benjamin Lucas Friedland, Michael Fritz

Gruppenmitglieder: Alexander Zwisler, Leon Kamuf, Leon Rudolph, Maurice Eisenblätter, Maximilian Gläfcke, Robin Seidel, Sina Opitz, Steve Woywod

Inhaltsverzeichnis

1 Dokumentationskonzept

Die gesamte Dokumententation wird in der englischen Sprache verfasst.

1.1 interne Dokumentation

Bei der internen Dokumentation gilt, sie auf ein Minimum zu reduzieren und nur dann anzuwenden, wenn der Code nur schwer nachzuvollziehen ist. Die internen Kommentare, die sich auf den Code beziehen, werden darüber gesetzt.
Die Kommentare selbst sind möglichst kurz zu fassen und bei längeren Kommentaren sind sinnvolle Zeilenumbrüche zu setzen, um eine gute Übersicht zu gewährleisten.

1.2 strukturelle Dokumentation

Die Komponenten und Funktionen des Quelltextes müssen soweit wie möglich kommentiert werden, damit davon ausgehend eine besonders umfassende und leicht zu navigierende Dokumentation generiert werden kann. Bei Funktionen und Komponenten, die von uns geändert werden ist die dazugehörige Dokumentation anzupassen.
Für die strukturelle Dokumentation in Angular benutzen wir Compodoc(https://compodoc.app) und für die API tsdoc (https://github.com/Microsoft/tsdoc), damit eine Web-Dokumentation von JavaScript-Datein generiert werden kann.
HTML- und CSS-Codes werden nur intern und nicht strukturell von uns dokumentiert.

1.3 Entwurfsbeschreibung

Die Entwurfsbeschreibung ist ein umfassendes Dokument, in dem alle unsere Struktur- und Modelierungs- und Designprinzipien enthalten und begründet sind. Mit Hilfe dieses Dokuments soll sich ein externer Dritter mit ausreichendem Grundwissen problemlos einarbeiten können, damit dieser gegebenenfalls Änderungen und Anpassungen vornehmen kann.
Die Entwurfsbeschreibung entsteht im Verlauf des Praktikums, da sämtliche Entscheidungen erst nacheinander eindeutig gemacht und dementsprechende dokumentiert werden müssen.
Generell wird ein Release alle relevanten Entwurfsbeschreibungen des Releases beinhalten.

2 Coding Standard

Mit dem Coding Standard wird festgehalten, an welche Style Guidelines es sich beim Coden zu halten gilt. Dies fördert zum Einen die Übersichtlichkeit und zu Anderen die Einheitlichkeit des Codes und sorgt damit für gute Lesbarkeit.
TSLint (https://palantir.github.io/tslint/) benutzen wir als Linter und Angular speziell benutzt codelyzer (http://codelyzer.com)(Set von Regeln für TSLint),da wir für das gesamte Projekt TypeScript (https://www.typescriptlang.org) benutzen.

3 Testkonzept

3.1 Durchzuführende Tests

Um eine generelle Funktionalität der Software zu gewährleisten, werden eine Reihe von Tests durchgeführt, die aufeinander aufbauen und somit bei fehlerhaften Anpassungen die Fehlersuche vereinfachen.
Gegliedert sind die Tests in Komponententests, Integrationstests und Systemtests.

3.1.1 Komponententests

Komponententests (siehe 3.2.1, 3.2.3), oder auch Unittests, prüfen die einzelnen Module auf ihre Funktionalität. Dafür sind je nach Anforderung an die entsprechende Komponente verschiedene Testreihen durchzuführen, um u.a. Genauigkeit, Geschwindigkeit sowie deren Grenzen zu überprüfen

3.1.2 Integrations- und Systemtests

Nachdem die einzelnen Module die Komponententests bestanden haben, werden sie im Integrationstest auf ihr korrektes Zusammenspiel als Einheit geprüft.
Ein Systemtest (siehe 2.3.2), welcher einer ganzen Reihe von Integrationstests mit immer wachsenden Einheiten zu Grunde liegt, simuliert die Nutzung der Software durch einen Anwender. Um einen erfolgreichen Test zu garantieren, werden verschiedene Situationen mit erwarteten Ergebnissen verglichen und eventuell auftretende Fehler oder Missstände dokumentiert und behoben

3.2 Tools

Wir verwenden das Continuous Integration Tool von unserer GitLab-Instanz. Tests werden jedes Mal automatisch ausgeführt, wenn Änderungen commited werden. Für die einzelnen Tests werden die Tools Karma. Protractor und Mocha benutzt.

3.2.1 Karma

Zum Testen unser JavaScript-Clients benutzen wir Karma (http://karma-runner.github.io/latest/index.html), welches standardmäßig von Angular bereitgestellt wird.

3.2.2 Protractor

Um bei Angular E2E-Tests durchzuführen, wird von uns das Angular-Framework Protractor (https://github.com/angular/protractor) verwendet.

3.2.3 Mocha

Wir verwenden zum Testen vom JavaScript das Mocha-Framework (https://mochajs.org/) in Kombination mit Chai (https://www.chaijs.com/), welche eine Assertion Library ist. Um die API zu testen, benutzen wir supertest (https://github.com/visionmedia/supertest#readme.

4 Organisatorische Festlegungen

4.1 Treffen

Zur Klärung aufkommender Fragen sowie zur Überprüfung des Fortschritts des Projekts gibt es jede Woche ein internes Gruppentreffen, wo nur das Team anwesend ist, und ein allgemeines Gruppentreffen zusammen mit den Tutoren und Betreuern.
Das interne Treffen findet immer vor und auch manchmal nach dem allgemeinen Treffen statt und dient einerseits zur Absprache, was mit den Tutoren besprochen werden soll und andererseits zur Auswertung und weiterer Zielsetzung.
Das Treffen mit den Tutoren findet stets freitags um 14Uhr in Raum P901 statt und dient zur Überprüfung des Projektfortschritts und zur Klärung aufgekommener Fragen.

4.2 Protokolle

Beide Gruppentreffen werden in separaten Protokollen festgehalten, die für jeden zugänglich auf Git bereitgestellt werden. Dort wird folgendes festgehalten:

- Datum
- Fragen und die dazugehörigen Antworten
- Verbesserungsvorschläge
- Zielsetzungen für die nächste Woche
- Gruppenentscheidungen

4.3 Abgaben und Termineinhaltung

Intern wurde festgelegt, Abgaben vor jedem Gruppentreffen fertigzustellen, damit genug Zeit bleibt, diese anzupassen und auch zu testen. Gemeinsam werden die Abgaben auf Mängel geprüft, Fragen geklärt und Feedback gegeben.
So wird sichergestellt, dass die vorgebenden Termine eingehalten und Abgaben fehlerfrei abgegeben werden.

4.4 GitFlow

Genrell wird bei uns nicht direkt in den Master gepusht, sondern Feature-Branches gepusht, die anschließend in den Master gemerged werden. Dafür brauch jede Feature-Branch mindestens 2 Approvels (also mindestens zwei Zustimmungen von anderen Team-Mitgliedern).
Ein Feature-Branche bezieht sich auf ein oder mehrere Issues, die abarbeitet werden. Außerdem ist beim Mergen darauf zu achten, dass Konflikte gelöst werden.