Dokumentationskonzept

HOR-TINF2021

Gruppe

Tobias Hahn
Lars Holweger
Fabian Unger
Timo Zink
Frank Sadoine
Fabian Eilber

Betreuer

Prof. Dr. Phil. Antonius van Hoof

Dokumentverlauf

Version	Beschreibung/Änderung	Autor	Datum
0.5	Erstellung eines ersten Entwurfs	Fabian Unger	18.03.2023
1.0	Verbesserung auf Grundlage des Feedbacks von Betreuer	Fabian Unger	23.03.2023
1.1	Korrektur und Finalisierung	Tobias Hahn	25.03.2023

Inhaltsverzeichnis

0. Ziel des Dokuments

Die Dokumentation bei Softwareprojekten ist aus verschiedenen Gründen sehr wichtig. Neben der Dokumentation des Codes für eine bessere Verständlichkeit und Nachvollziehbarkeit, ist ein gut kommentierter Code auch einfacher zu warten. Zudem können Entwickler dadurch auch schnell sehen, wo neue Funktionen eingebaut werden können, sodass auch die Skalierbarkeit davon profitiert.

Aus diesen Gründen ist das Ziel dieses Dokuments, eine konsistente Dokumentation zu definieren. Dies beinhaltet zu Beginn, wie Dokumente im Allgemeinen erstellt werden sollen und welche Eigenschaften diese erfüllen müssen. Des Weiteren werden für das Verfassen von Code verschiedene Code Conventions festgelegt. Zudem soll entwickelter Code sowohl intern in den Dateien selbst als auch extern über andere Notationen dokumentiert werden. Durch die Coding Conventions und die zusätzliche Dokumentation soll nicht nur die Lesbarkeit verbessert werden, sondern auch die Qualität mittels einer einheitlichen Darstellung. Da bereits feststeht, welche Dokumente erstellt werden müssen, werden für diese grobe Gliederungen festgelegt. Parallel dazu werden Verantwortlichkeiten für die Erstellung unterschiedlicher Dokumente auf die Gruppenmitglieder verteilt.

1. Richtlinien für die Erstellung von Dokumenten

Dokumente, die im Verlauf des Projekts erstellt werden, müssen folgende Eigenschaften beinhalten:

Dokumentverlauf

Jedes Dokument muss eine Tabelle beinhalten, die darstellt, welche Versionen des aktuellen Dokuments von wem erstellt und welche Änderungen getätigt wurden.

Ziel eines Dokuments

In dem ersten Kapitel aller Dokumente soll das Ziel des jeweiligen Dokuments erläutert werden. Dadurch soll dem Leser klar gemacht werden, was er erwarten kann und welche Informationen im Dokument enthalten sind. Auf dieser Grundlage soll das Ziel erläutert werden.

Status

Um feststellen zu können, ob Dokumente noch nicht begonnen, in Bearbeitung oder fertig sind, wird auf Microsoft Teams eine Aufgabenseite für die Dokumente erstellt. Diesen Aufgaben soll jeweils ein Status zugeordnet werden, an dem erkannt werden kann, inwiefern das Dokument der jeweiligen Aufgabe fertiggestellt ist.

Verantwortlicher für Erstellung

Für jedes Dokument muss ein Verantwortlicher für die Erstellung zugeordnet werden. Weitere Personen dürfen ebenfalls an dem Dokument arbeiten, jedoch ist der Verantwortliche der Ansprechpartner, falls Fragen zum Dokument aufkommen.

Verantwortlicher für inhaltliche Qualitätskontrolle

Damit das Dokument den Anforderungen entspricht muss es vor der Abgabe von mindestens einer Person gelesen und überprüft werden. Diese Person ist in erster Linie der Projektleiter Tobias Hahn. Hat dieser das Dokument selbst erstellt oder hat er keine Zeit, muss er eine vom Dokument unabhängige Person bestimmen, die das Dokument kontrolliert.

Sicherung

Die erstellten Dokumente müssen sich immer in dem Datei-Verzeichnis der Gruppe in Microsoft Teams befinden. Dadurch ist sichergestellt, dass alle Mitglieder darauf zugreifen können.

2. Code Conventions

Zunächst wird festgelegt, dass Code auf Englisch verfasst wird.

Für die Entwicklung des Backends wird die Programmiersprache Java verwendet. Zudem wird das Frontend in HTML, CSS und TypeScript entwickelt. Damit der Quellcode abgesehen von Dokumentation bereits lesbar und übersichtlich ist, werden verschiedene Richtlinien zum Schreiben von Code festgelegt. In den folgenden Unterkapiteln werden die Richtlinien festgelegt. Verantwortlicher für die Einhaltung dieser Richtlinien ist der Verantwortliche für Implementierung Fabian Eilber.

2.1. Java

Im folgenden Dokument werden die allgemeingültigen Code Conventions für Java erläutert:

https://www.oracle.com/technetwork/java/codeconventions-150003.pdf

Aufgrund der Zeit, müssen nicht alle Richtlinien angewandt werden. Im Folgenden werden jedoch Richtlinien auf Grundlage des Dokuments definiert, die unbedingt eingehalten werden müssen.

Namensgebung:

- Klassennamen müssen aus Substantiven bestehen. Besteht ein Klassenname selbst aus mehreren Worten, muss dieser in PascalCase angegeben werden.
    - Beispiel: „EventHandler“
- Namen von Interfaces werden wie Klassennamen angegeben.
- Methodennamen beinhalten zu Beginn ein Verb und können optional aus weiteren Wörtern bestehen, wobei dann der Methodenname in camel case angegeben werden muss.
    - Beispiel: „showAllUsers“
- Variablennamen müssen kurz, aber trotzdem aussagekräftig sein. Sie werden wie Methoden in lower camel case angegeben. Temporäre Variablen, wie Zählvariablen für Schleifen, können trotzdem aus einem Buchstaben bestehen.

- Konstanten müssen in Großbuchstaben und „_“ zwischen verschiedenen Wörtern angegeben werden.
    - Beispiel: „MAX_WIDTH“

Deklarationen:

- Es darf nur eine Deklaration pro Zeile erstellt werden.
- Deklarationen sollten direkt am Anfang eines Blocks gesetzt werden. Variablen sollten also direkt am Anfang eines Blocks deklariert werden und nicht erst dann, wenn sie verwendet werden.
- Sobald Variablen deklariert werden, sollten diese auch initialisiert werden. Es sei denn, sie sind an verschiedene Bedingungen des Programms geknüpft.
- Bei der Methodendeklaration darf zwischen der Methode und der Parameterliste kein Leerzeichen stehen.
- Die öffnende geschweifte Klammer sollte am Ende der Zeile, in der die Deklaration steht, gesetzt werden. Die Schließende bildet am Ende des Blocks eine eigene Zeile.
- Zwischen verschiedenen Methoden muss sich eine leere Zeile befinden.

Statements:

- Eine Zeile darf maximal ein Statement beinhalten.
- Der Aufbau verschiedener Statements wird für dieses Projekt nicht vorgegeben, kann jedoch in dem Dokument nachgeschlagen werden.

2.2. HTML

Unter der folgenden Webseite sind viele Richtlinien festgelegt:

https://www.w3schools.com/html/html5_syntax.asp

Diese müssen nicht alle eingehalten werden, bieten jedoch eine bessere Lesbarkeit. Im Folgenden werden die Richtlinien definiert, die unbedingt eingehalten werden müssen.

Document Type

- Zu Beginn jedes HTML-Dokuments muss folgender Dokumenttyp deklariert werden: <!DOCTYPE html>

Namensgebung

- Elementnamen / Tagnames müssen in Kleinbuchstaben verwendet werden.
- Attributnamen werden auch in Kleinbuchstaben angegeben. Die Werte von Attributen müssen in Anführungszeichen angegeben werden.

2.3. CSS

Die Grundlage für die Richtlinien für CSS bildet der Inhalt folgender Webseite:

https://docs.ckan.org/en/2.9/contributing/css.html

Auch hier gilt, dass nicht alle eingehalten werden müssen, im Folgenden jedoch Richtlinien festgelegt werden, die eingehalten werden müssen.

Namensgebung

- Alle IDs, Klassen und Attribute müssen kleingeschrieben sein. Sollten sie aus mehreren Wörtern bestehen, werden diese durch einen Bindestrich voneinander getrennt.

Properties

- Nach einem Doppelpunkt der Deklaration einer Property muss ein Leerzeichen gesetzt werden.
- Pro Zeile darf nur eine Deklaration einer Property erstellt werden.

2.4. TypeScript

Für die Code Conventions zu TypeScript bildet folgende Webseite die Grundlage:

https://www.itwinjs.org/learning/guidelines/typescript-coding-guidelines/

Wie bei Java, HTML und CSS gilt auch hier, dass diese Richtlinien lediglich Empfehlungen sind, und nicht zwingend im Projekt angewandt werden müssen. Folgende Richtlinien müssen von den Entwicklern jedoch angewendet werden.

Namensgebung

- Funktionen, Properties und Variablen werden in camel case definiert.
- Für Strings müssen Anführungszeichen verwendet werden.

3. Inline Dokumentation

Teil des Java Development Kits (JDK) ist auch das Software-Dokumentationswerkzeug Javadoc. Dieses erstellt aus bestimmten Kommentaren im Quellcode HTML-Dokumente, in der die angegeben Informationen übersichtlich dargestellt werden. Die Kommentare müssen mit „/**“ beginnen und mit „*/“ enden, Zeilen dazwischen beginnen immer mit einem „*“. Zudem müssen sie sich entweder vor einer Klassen-, Feld-, Konstruktor oder Methoden-Deklaration befinden. Der Inhalt der Kommentare wird in HTML verfasst. Dabei besteht dieser zunächst aus einer Beschreibung, gefolgt von verschiedenen optionalen Tags.

Die Liste aller Tags ist unter folgendem Link abrufbar:

https://docs.oracle.com/javase/8/docs/technotes/tools/windows/javadoc.html#CHDBEFIF

Im Folgenden wird der Aufbau eines solchen Kommentars für eine Klasse (siehe Quellcode 1) und eine Methode (siehe Quellcode 2) dargestellt.

```
/**
 * A class which…
 *
 * @author Autor Nummer 1
 * @author Autor Nummer 2
 */
```

Quellcode 1: Aufbau eines Kommentars vor einer Klasse für Javadoc

```
/**
 * Returns/Calculates/…
 *
 * @param first Description
 * @param second Description
 * @return Despription
 */
```

Quellcode 2: Aufbau eines Kommentars vor einer Methode für Javadoc

Für die Entwicklung und Programmierung in diesem Projekt sind folgende Vorgaben zu erfüllen:

1. Es muss vor jeder Klasse und vor jeder Methode (außer getter und setter) ein Kommentar in der beschriebenen Formatierung für Javadoc erstellt werden.
2. Diese Kommentare müssen eine kurze Beschreibung über die jeweiligen Funktionalitäten beinhalten.
3. Bei Klassen müssen neben der Beschreibung auch die Autoren des Codes über das Tag *author* angegeben werden. Dadurch lassen sich bei Problemen oder Fragen schnell Ansprechpartner finden.
4. Bei Methoden müssen neben der Beschreibung die Übergabeparameter über das Tag *param* und der Rückgabewert mit dem Tag *return* beschrieben werden.

Dies sind die Mindestanforderungen für Java-Code. Es dürfen weitere Tags verwendet werden, sodass geschriebener Code so genau wie möglich, aber nicht zu ausführlich erklärt wird.

Verantwortlicher für die Dokumentation im Code ist Fabian Unger. Er überprüft nicht nur die Formatierung, sondern auch den Inhalt der Kommentare. Jedoch sind alle Entwickler selbst dazu verpflichtet, ihre Implementierungen auf Englisch zu dokumentieren. Fabian Unger wird die Einhaltung der Vorgaben mindestens zwei Mal pro Woche und vor wichtigen Abgaben kontrollieren.

4. Externe Dokumentation des Codes

Eine weitere Möglichkeit, Quellcode zu dokumentieren, ist die Verwendung von UML-Diagrammen (Unified Modeling Language). UML ist eine standardisierte Modellierungssprache, die als Kommunikationsmittel zwischen Entwickler und Kunde dient. Deshalb werden neben der internen Dokumentation UML-Diagramme über wichtige Komponenten des Projekts erstellt. Für einen Gesamtüberblick eignen sich UseCase-Diagramme. Diese beinhalten die Akteure und die verfügbaren Funktionen. Um Funktionen

genauer darstellen zu können, sollen Klassendiagramme erstellt werden. Diese bilden auch die Grundlage für den Quellcode, indem die Klassen mit deren Attributen und Methoden in dem Diagramm bereits definiert werden. Die bisher genannten Diagramm sind jedoch alle statisch. Deshalb sollen für manche Funktionen Sequenzdiagramme erstellt werden. Diese beschreiben den Intra- und Intersystem-Datenaustausch aller instanziierten Klassen.

Timo Zink übernimmt als Verantwortlicher für Modellierung diese Verantwortung.

5. Weitere Artefakte

Da bereits festgelegt ist, welche Dokumente erstellt werden müssen, werden in den folgenden Unterkapiteln Gliederungen für die jeweiligen Dokumente definiert. Da die Erstellung mancher Dokumente noch nicht begonnen hat, ist es durchaus möglich, dass sich deren Gliederung im Laufe der Verfassung abändern wird. Trotzdem sollen diese Gliederungen bereits einen Überblick über Aufbau, Umfang und Inhalt des Dokuments geben.

5.1. Designbeschreibung

- Produktübersicht
  - Äußerliche Funktionsmerkmale
- Struktur- und Entwurfsentscheidungen der Anwendung
  - Gesamtarchitektur
- Struktur- und Entwurfsentscheidungen einzelner Pakete

Verantwortlicher: Timo Zink

5.2. Lastenheft

- Zielbestimmung
- Produkteinsatz
- Produktübersicht
- Produktfunktionen
- Produktdaten
- Produktleistungen
- Qualitätsanforderungen
- Glossar

Verantwortlicher: Tobias Hahn

5.3. Pflichtenheft

- Zielbestimmung
- Produkteinsatz
- Produktübersicht
- Produktfunktionen
- Produktdaten
- Produktleistungen
- Qualitätsanforderungen

- Benutzungsschnittstelle
- Nichtfunktionale Anforderungen
- Technische Produktumgebung
- Spezielle Anforderungen
- Gliederung in Teilprodukte
- Ergänzungen

Verantwortlicher: Tobias Hahn

5.4. Dokumentation zum Testkonzept

- Komponenten-Tests
  - Komponenten, Werkzeuge, Organisatorisches
- Integration-Tests
  - Komponentengruppen, Werkzeuge, Organisatorisches
- User-Interface-Tests
  - UI-Elemente, Werkzeuge, Organisatorisches

Verantwortlicher: Frank Sadoine

6. Fazit

Im Rahmen dieses Dokumentationskonzepts wurden Richtlinien zum Verfassen von Code und der Dokumentation festgelegt. Diese gilt es einzuhalten, um so eine hohe Qualität des Endergebnisses zu erzielen. Die hohe Qualität fördert neben der Lesbarkeit und Verständlichkeit auch gleichzeitig eine spätere Weiterentwicklung und Skalierung des Projekts.