Qualitätssicherungskonzept

Version 1.0

Jan Gruchott

Dokumentenhistorie

Datum	Version	Beschreibung	Autor
26.03.2021	Version 1.0	Erstellung des Dokuments	Jan Gruchott

Inhaltsverzeichnis

1. Eingesetzte Werkzeuge/Tools

Projektmanagement:

Notion -> gemeinsam an Dokumenten arbeiten | Meeting Protokolle | gemeinsamer Bereich | News | Projektablaufplan | to-do's

MS Teams -> Dateiverwaltung/ -austausch | Kommunikation

Analyse:

Lucidchart -> UML Diagramme | intuitive Bedienung | gute Templates

Entwurf:

Lucidchart -> UML Diagramme | intuitive Bedienung | gute Templates

Implementierung:

IDE -> VisualStudio | Unterstützung Codeerstellung
GitHub -> Codeverwaltung | Versionsverwaltung

Server:

PuTTY -> dient als SSH Client und stellt die Verbindung zum Server her
WinSCP -> Daten- und Dateitransfer

Test:

Angular:
Karma -> Testframework für Angular

Python:
Unit Tests -> Testframework von Python

Dokumentation:

Dokumentationen:
Word -> Dokumentenerstellung
Excel -> Tabellenkalkulation | Zeiterfassung erstellen

Codedokumentation:
Pydoc -> umfangreiche Pythondokumentation, die Docstrings mit einbaut, spricht es wird eine Dokumentation des Quellcodes abrufbar

2. Entscheidung für Werkzeuge

Alle der aufgeführten Werkzeuge waren vorher mindestens einem Teammitglied bekannt. Falls nun ein Werkzeug benötigt wurde, wurde sich abgesprochen, ob jemand Erfahrung mit einem Tool hat welches die Anforderungen erfüllen kann. Falls mehrere Tools zur Auswahl standen wurde abgewogen welches der Tools die Anforderungen besser erfüllt und eine Entscheidung für das sinnhaftere Werkzeug gefällt. Falls sich herausstellt, dass ein Tool nicht so funktioniert wie erwartet wird dem Team eine Option vorgeschlagen und gemeinsam entschieden. So waren beispielsweise einige Teammitglieder nicht mit der Dateiverwaltung in Notion zufrieden. Eine Option war, Dateien in unserem MS Teams Team zu verwalten. Nach einer gemeinsamen Absprache war klar, dass Teams das geeignetere Tool ist und die Dateiverwaltung wurde verlegt. Zur Kommunikation wurde MS Teams ebenfalls gewählt, da uns die Anwendung allen durch Vorlesungen und aus den Praxisphasen bekannt war und durch eine schnelle Kommunikationsform via Chat oder Sprache geeignet erschien. Lucidchart wird in unserem Team zur Erstellung von UML-Diagrammen verwendet. Die Entscheidung liegt unter anderem der intuitiven Bedienung, hilfreichen Templates und der Erfahrung mit dem Tool zugrunde. Einige von uns haben das Werkzeug bereits für ihre T1000 verwendet und waren bereits mit der Bedienung vertraut. Als IDE wurde sich für VisualStudio Code entschieden. Dies ermöglicht eine Codevervollständigung, Debugging, Testen sowie Git-Management zusammengefasst in einem Tool. Das können andere IDE's auch, jedoch lag die Entscheidung der vorhandenen Erfahrung zu Grunde. Ein weiterer Vorteil von VS Code ist, dass man alle im Projekt benötigten Programmiersprachen in einer Anwendung hat. Für GitHub wurde sich entschieden, da es das uns gängigste Tool in Richtung Versions-/Codeverwaltung ist. Die Tools für die Server stellen eine Ausnahme zu den restlichen Tools dar. Wir wussten das wir in die Richtung SSH Client und Daten-/Dateitransfer ein Tool benötigen, jedoch hatte damit noch keiner Erfahrung. Somit haben wir uns an das Internet gewendet und nach geeigneten Tools gesucht. PuTTY und WinSCP waren die Tools für die wir uns entschieden da sie die Anforderungen erfüllen können. Für Tests benutzen wir für Angular das Tool Karma, da es von Haus aus mitgeliefert wird. Das gleiche gilt für Python mit den Unit Tests. Zur Dokumentenerstellung wurde sich für Word entschieden, da alle mit dem Tool und dessen Funktionsumfang vertraut sind. Selbiges gilt für Excel, dem Team ist das Tool bekannt und die umfangreichen Funktionalitäten sind gleich für mehrere Dokumente einsetzbar. Für die Dokumentation in Python ist nach Recherche das Tool Pydoc herausgestochen. Innerhalb von Pydoc ist eine umfangreiche Python Dokumentation enthalten. Außerdem wird diese vorhandene Dokumentation erweitert, durch die eingebauten docstrings in Funktionen, Klassen, Modulen und Methoden. Spricht es wird automatisch eine Dokumentation über das entworfene Programm angelegt. Diese Funktion hat überzeugt und deshalb wurde sich dafür entschieden.

3. Qualitätsstandard

Coding Conventions:

- Gute Verbalisierung von Variablen
- Frontend: Methoden kommentieren und feste Typzuweisung bei Variablen und Returnvalues
- Sinnvolle/möglichst exakte Bezeichnung für Klassen und Methoden
- Kommentierung von Quellcode (falls nicht eindeutig erkennbar)
- Backend: Kommentierung der Aufgabe/Funktion jeder Klasse/Methode (docstrings)
- Testen von fertiggestelltem Code
- Testabdeckung: Zu jeder Komponente mind. ein Test
- Codeformatierung entsprechend autopep8 Standard (VisualStudio Code)
- Code Versionierung und Autor angeben (GitHub)
- Getesteter Code integrieren
- 4 Augen Prinzip

Python Name Conventions:

Bezeichner-Typ	Regeln für Benennung	Beispiele
Klasse / Interface	- Erster Buchstabe groß (CapWords)	MyClass
Methoden / Funktionen	- Alles lowercase - Wörter getrennt durch unterstrich - Erstes Argument ist immer self bei Methoden von Klassen	run(self) add_name(id, name)
Globale Variablen	- Alles lowercase - Zwei Unterstriche vorne und hinten - Wörter getrennt durch unterstrich	__max_health__ __health__
Lokale Variablen	- Alles lowercase - Wörter getrennt durch unterstrich	max_health
Konstanten	- Alles uppercase - Wörter getrennt durch unterstrich	MAX_OVERFLOW TOTAL

Python Comment Conventions:

Dokumentiert wird der Code durch Kommentare sowie docstrings zur Beschreibung von Klassen, Methoden und Funktionen.

Beispiele:

One-line Docstrings:

```
def square(a):
    """Return argument a is squared"""
    return a**a
```

Multi-line Docstrings:

```
def some_function(argument1):
    """Summary or Description of the Function

    Parameters:
    argument1 (int): Description of arg1

    Returns:
    int:Returning value

    """
    return argument1
```

Angular Name Conventions:

Bezeichner-Typ	Regeln für Benennung	Beispiele
Klasse / Interface	- Erster Buchstabe groß (CapWords)	MyClass
Methoden / Funktionen	- lower camelcase	run() runServer()
Variablen	- lower camelcase	maxHealth
Konstanten	- Alles uppercase - Wörter getrennt durch unterstrich	MAX_OVERFLOW TOTAL

Test Python:

Um einen Unittest in Python durchzuführen kann ein von Python selbst vorhandenes Framework verwendet werden. Dieses Framework ermöglicht es primitive Funktionen wie ein assert möglichst simpel umzusetzen.

```
import unittest

class TestStringMethods(unittest.TestCase):

    def test_upper(self):
        self.assertEqual('foo'.upper(), 'FOO')
```

Um einen Test zu erstellen muss man zunächst eine entsprechende Testklasse initialisieren. Diese Testklasse beinhaltet dann die verschiedenen Testmethoden, welche wie üblich ein assert enthalten.

Test Angular:

Benutzt wird das Tool Karma. Tests sind hierbei wie folgt aufgebaut:

```
it('should be created', () => {
    expect(service).toBeTruthy();
  });
```

Ein Test beginnt immer mit dem Methodenaufruf *it*, darauf folgt eine Testbeschreibung in Form eines Strings, welche immer mit *should* beginnen sollte. Mit dem Aufruf *expect* wird der Erwartungswert angegeben. Der Erwartungswert muss ein Wahrheitswert sein. Verantwortlicher für die Tests ist der Ersteller des Codes. Der Testbeauftragte überprüft ob die Test in einem angemessenen Umfang durchgeführt wurden.

4. Dokumentation

Verantwortlicher für Dokumentation ist primär der Ersteller von Code. Beim Pair Programming ist das Paar gemeinsam verantwortlich. Außerdem ist der Verantwortliche für Qualitätssicherung und Dokumentation dazu beauftragt Code auf seine Qualität (bspw. auch Dokumentation) zu prüfen und gegebenenfalls mit dem Ersteller des Codes zu verbessern. Dokumentiert wird der Code durch Kommentare sowie docstrings zur Beschreibung von Klassen und Methoden. Dokumentation außerhalb von Code ist durch folgende Vorlagen einheitlich zu halten:

Dokumentenvorlage.docx

Lasten_Pflichtenheftvorlage.docx