Quality Assurance Document
Umsetzung eines Maulwurf Company-Spiels

1. Definition of Done

Das Backlog enthält eine Übersicht über alle Aufgaben, die im Projektverlauf noch zu erledigen sind. Alle User Stories sollten klar und eindeutig formuliert und vom Product Owner abgesegnet worden sein.

Für die Fertigstellung der Software sind alle User Stories des Backlogs vollständig abzuarbeiten.

Damit die einzelnen, fertig bearbeiteten Backlog Elemente jedes Entwicklers fehlerfrei und formal passend in das Gesamtprojekt integriert werden können, ist der folgende Ablauf für die Bearbeitung der User Stories und deren Sub-Tasks einzuhalten:

I) Organisierte User Stories
- Die User Story wurde im Sprint-Meeting einem Entwickler/Entwicklerteam zugeordnet
- Der Beginn der Entwicklung wurde vom Product Owner abgesegnet

II) Code
- Die User Story inklusive aller Sub-Tasks wurden vollständig implementiert
- Bei der Programmierung wurden die Qualitätsstandards des Google Java Style Guides eingehalten

III) Code-Review
- Geschriebener Code wurde entweder im Pair Programming entwickelt oder durch andere Teammitglieder überprüft

IV) Dokumentation
- Der neu geschriebene Quellcode wurde verständlich kommentiert
- Das DevOps Dokument wurde kontinuierlich aktualisiert

V) Test
- Sowohl manuelle als auch automatische Tests wurden basierend auf dem erarbeiteten Testplan durchgeführt

VI) Fertigstellung
- Das fertige Ergebnis wurde dem Product Owner vorgestellt und von diesem auf Vollständigkeit geprüft
- Das implementierte Feature wurde in den Main Branch eingefügt

I) Organisierte User Stories

Eine neu zu bearbeitende User Story muss im Sprint-Meeting mit dem Projektteam besprochen und einem einzelnen Entwickler oder einer Entwicklergruppe zugeordnet werden, damit die Entwicklung der User Stories in einer sinnvollen Reihenfolge erfolgt.

Der Product Owner muss dem Entwicklungsstart jeder User Story zustimmen.

II) Code

Zu unseren Qualitätsstandards gehört ein vollständiger, verständlicher, übersichtlicher und kommentierter Code.

Beim Erstellen der Quellcodes halten wir uns an die aktuellen Code Standards von **Google Java Style Guide**[1], insbesondere die Verwendung der CamelCase-Notation. Die Klassennamen sollen grundsätzlich Substantive oder Nominalphrasen sein, während Methodennamen Verben oder Verbalphrasen sein sollen.

Zu Beginn jeder Quellcodedatei ist ein Urheberrechtshinweis anzugeben.

Die Dokumentation der Klassen und Methoden erfolgt in englischer Sprache. Diese sollte kurz und knapp sein und eine Erklärung über die Aufgabe und Funktionsweise der Methode bzw. Klasse sowie eine Auflistung der übergebenen Variablen und des Rückgabewertes enthalten. Dabei verwenden wir den Javadoc Standard gemäß der Google Java Style Guide Konventionen.

Damit diese Code Konventionen entsprechend eingehalten werden, benutzt das Team das google-java-format Plugin[2] zur Unterstützung.

III) Code-Review

Bevor der Code in den Development Branch eingefügt wird, muss sichergestellt werden, dass mindestens ein anderer Entwickler den Code überprüft hat (Review). Für allein implementierte Features muss ein Merge-Request in GitLab erfolgen, beim Pair Programming kann der Code direkt übernommen (gemerged) werden.

IV) Dokumentation

Neu bearbeitete Features müssen vollständig dokumentiert werden, um eine zukünftige Veränderung und Erweiterung des Programms problemlos zu ermöglichen.

---

[1] https://google.github.io/styleguide/javaguide.html
[2] https://plugins.jetbrains.com/plugin/8527-google-java-format/

Neben der Dokumentation innerhalb des Quellcodes selbst, wie in ***II) Code*** beschrieben, wird ebenfalls ein DevOps Dokument mit Handbuch abgegeben. In diesem Dokument wird der Aufbau des Programmcodes inklusive der Strukturen erläutert und ein Handbuch erstellt, welches genau erklärt, wie das Programm gebaut, installiert und verwendet werden kann.

Sämtliche Anpassungen am Programm müssen in die oben genannten Dokumente übernommen werden.

Die finale Dokumentation bei der Endabgabe beinhaltet zudem ein Interface Dokument, in dem eine Übersicht der Klassen sowie die genaue Spezifikation des Protokolls und die Definition des Dateiformats für Konfigurationen enthalten sind.

V) Test

Nachdem der Quellcode für die zu bearbeitende User Story inklusive sämtlicher Sub-Tasks fertiggestellt wurde, muss das Programm auf Fehler und Funktionalität geprüft werden, bevor der Programmteil in den Main Branch übernommen werden darf.

Die Überprüfung erfolgt sowohl manuell als auch automatisch.

Der genaue Ablauf und die Kriterien jedes vorgesehen Tests ist unter ***2. Testplan*** beschrieben. Diese Kriterien sind von jedem Entwickler einzuhalten.

VI) Fertigstellung

Fertiger Quellcode wird beim Product Owner vorgestellt, von diesem überprüft und mit dessen Zustimmung in den Main Branch übernommen und gilt somit als abgeschlossen.

Sollten nicht alle zu beachtenden Punkte zufriedenstellend erfüllt sein, muss die User Story im nächsten Sprint erneut bearbeitet werden.

Bei grundlegenden Problemen mit einer User Story oder deren Sub-Tasks müssen die Punkte der User Story neu evaluiert und die User Story gegebenenfalls ergänzt oder aufgeteilt werden.

2. Testplan

Jede fertig implementierte User Story muss vor ihrer Abnahme auf vollständige Funktionalität getestet werden.

Um einen Überblick über die Tests aller Komponenten zu ermöglichen, sollen die Testresultate in einem Dokument gesammelt und strukturiert werden. Hier muss insbesondere erkennbar sein, welche Teile des Codes durch automatisierte und/oder manuelle Tests abgedeckt wurden.

Die automatisierten Tests werden aus GitLab CI entnommen und dem Dokument hinzugefügt.

Die folgenden Punkte müssen beim Testen erfolgreich durchlaufen werden:

I) Unit Test
- einzelne Methoden testen
- automatische Tests über GitLab CI durchführen

II) Code-Review
- Qualitätsstandards einhalten
- Laufzeit und Performance manuell testen

III) Integration
- Zusammenspiel verschiedener Programmteile testen

IV) Usertests
- manuelle Tests durch mehrere Entwickler
- Tester waren nicht an der Entwicklung des zu testenden Features beteiligt

V) Systemtest
- vollständige Funktionalität testen
- Beteiligung des gesamten Projektteams

VI) Abnahme
- Vorstellung der Testergebnisse und Abstimmung mit dem Product Owner

I) Unit Test

Für das automatisierte Testen verwenden wir das Framework JUnit, um Teile des Programms isoliert testen zu können und auch nach der initialen Implementierung die Funktionalität dieser Teile sicher zu stellen.

Unit-Tests ermöglichen es hierbei, die Teile des Programms strukturiert und über die Entwicklung standardisiert zu testen. Dabei ignorieren wir einfache “is”-Methoden und konzentrieren uns auf die Teile des Programms, die entsprechend ihrer Komplexität fehleranfälliger sind.

Zudem dienen diese Tests dazu, Probleme exakten Methoden und Bereichen des Codes zuordnen zu können, um so die Fehlersuche zu minimieren.

Automatisierte Tests innerhalb der GitLab Umgebung werden durch GitLab CI durchgeführt. GitLab CI ist ein Tool, mit dem sich Container Umgebungen (Docker) zum Ausführen von Tests automatisch erzeugen lassen. Hierfür werden über eine yaml Datei im Repository die einzelnen Tests definiert, die in diesen Containern bei Merges und Merge-Requests ausgeführt werden sollen. So soll die Funktionalität der Programmteile sichergestellt und Fehler abgefangen werden, bevor diese in das fertige Produkt integriert werden. Ebenso geben diese Testläufe dem Team Meldung, falls beispielsweise ein Test nicht fehlerfrei durchlief.

II) Code-Review

Jeder Code muss auf die Einhaltung der Qualitätsstandards geprüft werden. Falls bereits eine Überprüfung innerhalb des Pair Programmings erfolgte, kann dieser Schritt übersprungen werden.

In dem Code-Review muss zunächst geprüft werden, ob die Vorgaben des Google Java Style Guides eingehalten wurden.

Zusätzlich wird der Code auf mögliche Fehler untersucht und subjektiv auf Sinnhaftigkeit überprüft. Dies beinhaltet zum Beispiel die Bewertung, ob ein angewandter Algorithmus sinnvoll für die Lösung eines Problems ist.

III) Integration

Integrationstests sind als nächster Schritt nach Unit-Tests anzusehen. Anstatt nur einzelne Programmteile und Methoden zu testen, wird hier das Zusammenspiel verschiedener Teile des Programms getestet. Dafür werden entsprechende Testszenarien definiert.

IV) Usertests

In den Usertests werden neue Komponenten von mindestens einer anderen Person manuell getestet. Dieses Vorgehen soll die Genauigkeit des Tests erhöhen und die Anzahl möglicherweise übersehener Fehler in den Programmfunktionen minimieren.

Dabei sollen die testenden Personen zunächst eigenständig jedes mögliche Szenario des zu testenden Features ausprobieren. Im zweiten Schritt werden die Testergebnisse mit denen der anderen Tester verglichen und das Feature gegebenenfalls erneut von allen getestet.

Es werden Tests für Standard- und Randfälle erzeugt, um sicherzustellen, dass die Funktionen fehlerfrei durchlaufen.

Bei den Tests soll außerdem sichergestellt werden, dass sich Laufzeit und Performance, gemäß der subjektiven Einschätzung des jeweiligen Testers, in einem akzeptablen Rahmen befinden.

V) Systemtest

Vor dem Erreichen bestimmter Meilensteine (Messe, Endabgabe beim Kunden) müssen die neuen Komponenten im gesamten Programm getestet werden.

Hier wird nicht mehr auf Fehler in einzelnen Funktionen, sondern die erfolgreiche Integration größerer Teilsysteme ins Gesamtsystem, getestet.

Das Programm ist auf vollständige Funktionalität zu prüfen. Dies soll je nach Plattform, auf die sich das neue Feature bezieht, auf mindestens einem Endgerät mit Windows, Ubuntu oder Mac OS X mit Java SE 11 oder mindestens einem Endgerät ab Android Version 5.0.1 (API-Level 21) erfolgen. Dabei ist auch die Netzwerkkommunikation, das heißt die Kommunikation zwischen PC, Android, AI und Server, zu testen.

Es ist sicherzustellen, dass das Programm die definierten Spezifikationen einhält.

Systemtests müssen von den Testmanagern rechtzeitig organisiert werden, damit genug Vorlaufzeit für die Beseitigung eventuell auftretender Fehler zur Verfügung steht.

Bei den Systemtests sollte das gesamte Projektteam beteiligt sein, da auf diese Weise jeder Entwickler insbesondere auf die von ihm entwickelte Komponente achten kann und dadurch möglichst keine Teilbereiche des Programms übersprungen werden.

VI) Abnahme

Das Testverfahren gilt als abgeschlossen, wenn alle Punkte des oben beschriebenen Testplans erfolgreich durchlaufen, die Testresultate vom Entwickler vorgestellt und der Abschluss der Testphase anschließend durch den Product Owner bestätigt wurde.