Softwarequalität – Aufgabe 6 – Team A

Lösung zu Schritt 5:

1. Es wurden insgesamt 22 Issues gefunden, davon betreffen 5 Issues die Security, 9 Issues die Reliability und 8 Issues die Maintainability
2. Das zuerst auftauchende Issue mit der Severity Blocker heißt „Add password protection to this database“. Das Problem ist, dass der Zugriff auf eine Datenbank ohne einen Passwortschutz eine Sicherheitslücke, die jeder ausnutzen kann. Daten können so von jedem genutzt und manipuliert werden. Es gibt noch 3 weitere Issues mit der Severity Blocker
3. Wir konnten keinen Issue mit der Severity „Critical“ finden. Ist hiermit eher als Severity „Blocker“ oder „High“ gemeint gewesen?

Schritt 6:

Der Fehler liegt im Schritt Run SonarSource/sonarqube-quality-gate-action@v1, Quality Gate has failed in Zeile 50.

Das neue Issue ist von der Severity Medium, gehört zur Maintainability und heißt „Remove this commented out code“

Aufgabe mit PetClient

Aufgabe 1:

1. 91,9%
2. Circa 1200 Lines of Code
3. 296 Lines of Code werden abgedeckt

Aufgabe 2:

1. model und vet
2. system / PetClinicRuntimeHints.java
3. Service-Klassen beinhalten oft die Hauptlogik, während Model-Klassen nur die Daten aufnehmen und aufbereiten, also wird bei Serviceklassen potenziell mehr getestet, dementsprechend ist hier die Coverage höher

Aufgabe 3:

1. CacheConfiguration.java
2. CacheConfiguration.java, WelcomeController.java, PetController.java
3. Die Controller-Klassen sollten priorisiert getestet werden, da diese die viele Funktionalitäten beinhalten

Aufgabe 4:

Da es nur den main branch gibt, können wir keine weiteren branches ansehen.

Wenn wir im Nachhinein richtig gesehen haben, dann haben wir das Projekt nur mit dem main-branch geforked, in dem Projekt gibt es ja abseits davon noch mehr branches

Aufgabe 5:

1. Ein Produktionsprojekt sollte definitiv eine hohe Testcoverage haben, damit hier so wenig Fehler wie möglich passieren. Wahrscheinlich kann man nicht immer alles abdecken, aber eine hohe Code Coverage sollte vorhanden sein
2. Model-Klassen bzw. alles, was sich oder den Output einer Aktion nicht verändert, Dinge die keine Berechnungen oder Weiterverarbeitung mit sich ziehen, etc. -> keine Business-Logik & Datenverarbeitung
3. Bei neuen Features die in einem branch entwickelt werden, bei Bugfixes oder Breaking Changes