Qualitätssicherungskonzept JavaTiles

Organisatorische Massnahmen

Um eine effektive Zusammenarbeit zu ermöglichen, haben wir ein Netzwerkprotokoll implementiert, das klare Befehle zwischen Server und Client ermöglicht. Dadurch können wir flexibler arbeiten, indem zwei Teammitglieder am Server und zwei am Client arbeiten.
Darüber hinaus haben wir mithilfe von Ganttproject die Aufgaben klar verteilt und festgelegt, wer welche Aufgabe bis wann erledigen muss.
Ausserdem haben wir eine WhatsApp-Gruppe gebildet, um uns gegenseitig immer auf dem neuesten Stand zu halten, sei es über neue Probleme, die aufgetaucht sind, sei es um spontan ein Meeting zu organisieren oder auch um neue Ideen für das Spiel einzubringen.

Technische Massnahmen

Die Integration von Checkstyle in IntelliJ ist ein wichtiger Bestandteil unserer QA-Massnahmen, um die Einhaltung von Coding-Standards sicherzustellen. Durch regelmäßige Überprüfungen während des Entwicklungsprozesses gewährleisten wir eine konsistente Codequalität. Dies fördert eine verbesserte Zusammenarbeit im Team und ermöglicht eine frühzeitige Identifizierung potenzieller Probleme.

Zusätzlich zum Checkstyle haben wir auch das Plug-in Tabnine installiert, welches ein KI-gesteuertes Codevervollständigungstool ist. Tabnine soll uns helfen, unseren Code schneller zu schreiben und auch weniger Fehler zu machen.

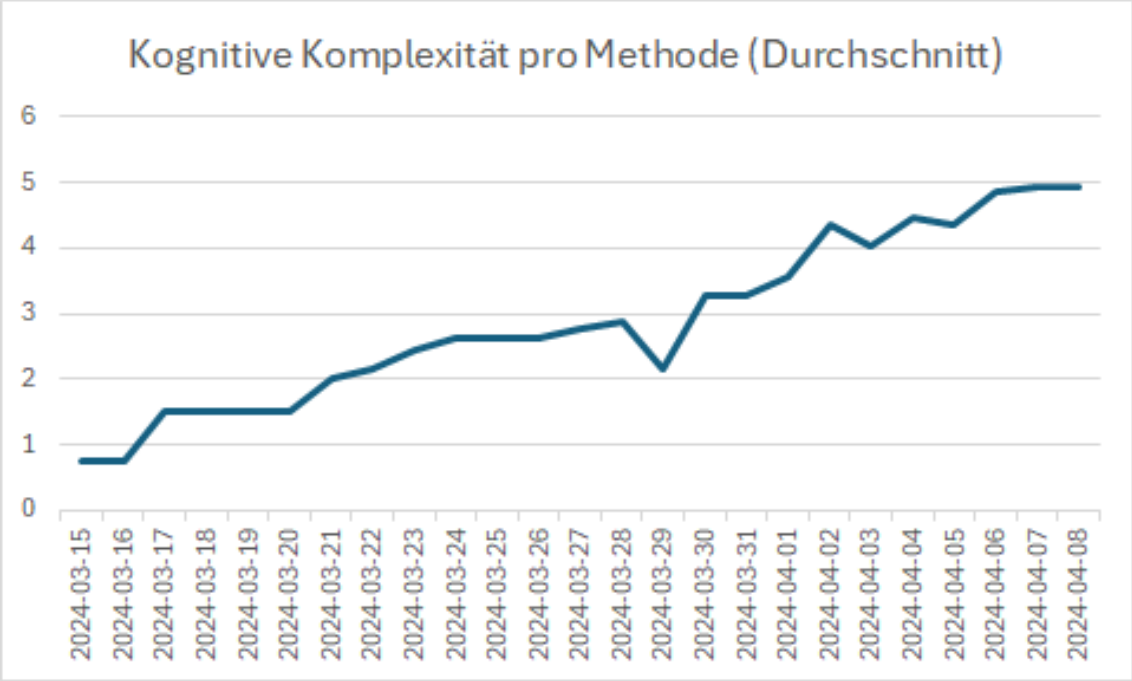
Um mögliche Fehler schneller zu finden und zu beheben, haben wir die Bibliothek Apache Log4j 2 importiert. Diese ermöglicht es uns, den Zustand unserer Anwendung detailliert zu überwachen und zu protokollieren. Durch präzises und konfigurierbares Logging auf verschiedenen Ebenen (DEBUG, INFO, WARN, ERROR) können wir die Ursachen von Problemen schnell identifizieren und beheben, ohne dabei auf invasive Debugging-Methoden angewiesen zu sein. Mit Log4j 2 haben wir die Möglichkeit, unsere Log-Nachrichten flexibel zu gestalten, was von unschätzbarem Wert ist, wenn es darum geht, den Überblick über den Betriebszustand unserer Anwendung in Echtzeit zu behalten. Die Konfiguration von Log4j 2 lässt sich nahtlos an unsere Bedürfnisse anpassen, und durch die Nutzung von asynchronen Loggern minimieren wir Leistungseinbußen, die durch das Logging verursacht werden könnten.

Analytische Massnahmen

Problematische Codeabschnitte wurden im Rahmen eines gemeinsamen Teamdurchlaufs überprüft. Neue Änderungen oder Implementierungen wurden stets mit dem gesamten Team kommuniziert, um sicherzustellen, dass jedes Mitglied auf dem neuesten Stand ist und vor allem den Code versteht. Es liegt uns sehr am Herzen, dass jedes Teammitglied gleichermaßen in das Projekt eingebunden ist, damit das Beste aus allen Einzelbeiträgen einfließen kann.
Eigene Codeabschnitte der Teammitglieder wurden kritisch geprüft und diskutiert, um die Qualität und Kohärenz des Gesamtprojekts sicherzustellen. Immer wieder haben wir Simulationen an unserem Code durchgeführt, um mögliche Schwachstellen ausfindig zu machen. Daraufhin konnten wir entsprechende Änderungen durchführen, um unser Programm stetig zu verbessern.

Das Checkstyle Plug-in hilft uns auch den Code zu analysieren, indem es beispielsweise anzeigt, falls Bibliotheken nicht gebraucht werden und auch wenn eine Zeile zu lang ist, um auch den Code möglichst lesbar zu halten oder auch auf fehlende Javadoc-Kommentare aufmerksam zu machen.

Metrik



Für unser Projekt haben wir als Metrik den Durchschnitt der kognitiven Komplexitäten der Methoden von metricsReloaded (dort genannt cognitive complexity) genommen. Unser Ziel ist es natürlich, diesen Wert nicht allzu gross werden zu lassen.
Auf dem obigen Graphen sehen sie die Entwicklung dieses Wertes über den bisherigen Verlauf unseres Projekts. Was einem vielleicht auffällt, ist dass dieser Wert bisher ziemlich konstant wuchs. Hauptverantwortlich dafür sind die Methoden, die die Requests auf dem Server und dem Client behandeln.
Um den Wert in Zukunft tiefer zu halten, werden wir häufigere Sessions zum Refactoring einplanen, um solche grossen Methoden in kleinere aufzuteilen.