Protokoll zum ersten Treffen (23. Juni 2016)

Bei unserem ersten Treffen wurden die Grundlagen des Projektes sowie die ersten Zuständigkeiten besprochen.

Nach der Lektüre der Aufgabenstellung wurden die Erweiterungen ausgesucht, die auf jeden Fall in das Programm aufgenommen werden sollen. Dabei haben wir uns für „ATDD“ und „Babysteps“ entschieden. „Tracking“ wollten wir jedoch nicht gänzlich ausschließen, sodass wir uns die Implementierung dieser Funktion offen halten.

Als Format für zu ladende Tests haben wir uns aufgrund der guten Möglichkeit zum Laden in Java für XML entschieden. Das Programm öffnet standardmäßig eine *catalog.xml*-Datei in dem Ordner, in dem das Programm liegt, um die Aufträge zu laden. Optional kann jedoch auch eine andere Datei im Menü gewählt werden, die dem Nutzer zugeschickt wird und woanders liegt.

In dem Programm, das für Übungszwecke an Universitäten eingesetzt werden können soll, muss auch eine Abgabefunktion enthalten sein. Diese möchten wir auch implementieren, die dann die Abgabe als eine zip-Datei organisiert. Diese zip-Datei kann dann wie gehabt über das Abgabesystem abgegeben werden. Optional ist auch eine Abgabe für jede einzelne Aufgabe auswählbar.

Damit die Implementierung dem Studenten vereinfacht wird, ist auch ein Highlighting für die Textboxen einzubauen. Dafür haben wir uns vorerst für „richtextfx“ entschieden, jedoch ist auch eine andere „Open Source“-Variante vorstellbar. Die damit einhergehenden Lizenzen müssen daher auch in die Lizenzdatei eingetragen werden.

Wir versuchen, FXML-Dateien aufgrund der besseren Lesbarkeit und auch Wartung zu verwenden und eine CSS-Datei für das gesamte Projekt für ein einheitliches Aussehen zu verwenden.

Der Code soll weitestgehend objektorientiert sein, um die Funktionen und Möglichkeiten von JavaFX bestmöglich ausschöpfen zu können.

Ersteinmal möchten wir uns offen halten, ob Tests verpflichtend genutzt werden müssen. Wir stellen jedem Mitglied frei, wie er seine eigenen Programmteile testet. Kommt es jedoch dazu, dass wir innerhalb der Gruppe merken, dass einzelne Klassen immer wieder Fehler aufweisen, so behalten wir uns vor, einen Katalog von Tests bei jedem Meeting verpflichtend zu übergeben, sodass mögliche Fehlerquellen vom gesamten Team ausgeschlossen werden können.

Damit es zu einer Übersichtlichkeit und der Masterbranch immerzu funktioniert und stets den gewünschten Code enthält, werden die einzelnen Features in einem jeweils separaten Branch entwickelt werden. Nach Programmierung des Features wird dann ein „Pull Request“ gestellt, sodass alle Teammitglieder ihr Feedback abgeben können und der Programmierende Verbesserungen einbauen kann. Es wird dann gemerged, wenn alle einverstanden sind. Sollte es zu Mergekonflikten kommen, wird vor einem Teamtreffen auch schon darüber diskutiert, sodass für kein Subprojekt Nachteile entstehen.

Zuständigkeiten bis zum nächsten Treffen

Mitglied	**Zuständigkeiten**
Eduard	ZIP-Library und Dr. Bendisposto's Library analysieren, um geeignete Einbettung ins Programm zu finden
Oliver	Absprache mit einzelnen Mitgliedern, um die Grundlogik im allgemeinem zu entwickeln: Welche Methoden fragen was ab? Welche Methoden bekommen was übergeben? (Eine „innere“ Dokumentation des Programms. Soll heißen, dass das Programm nicht als Blackbox gesehen wird.)
Lucas	GUI entwerfen: Erste CSS und FXML
Josias	Travis- und Gradle-Konfigurationen vornehmen. Ersten Commit im Master vornehmen, damit weitere Branches eine Basis haben. Finden eines passenden Java-Editors in FX
Niklas	XML lesen und speichern. Dabei geht es um Katalog einlesen und Abspeichern dessen, was ein Studierender zwischenspeichern möchte.

Diese Zuständigkeiten sind für die erste Woche zu sehen. Darin sind die grundlegenden Vorarbeiten abgedeckt, sodass die nächsten Wochen weitere Verfeinerungen abgestimmt werden können.