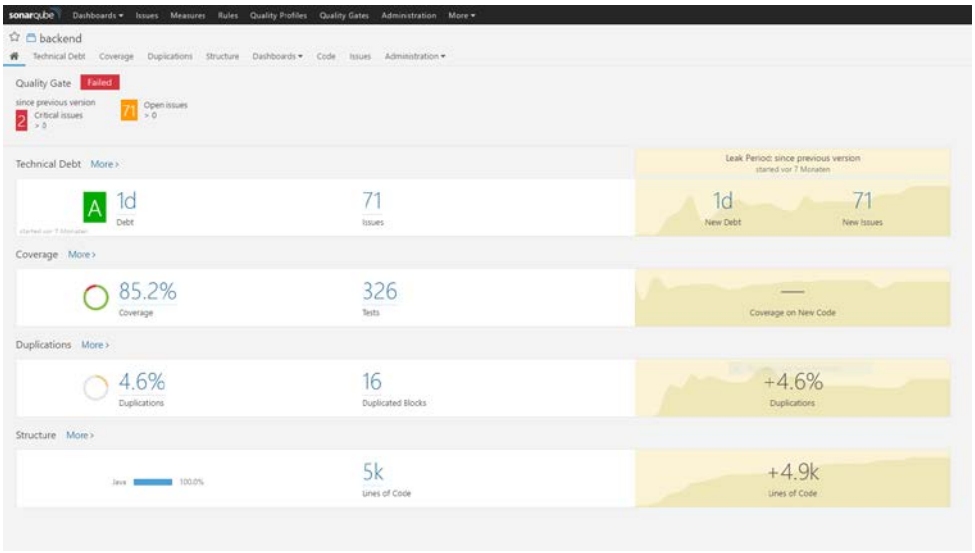
Refactoring von StudentBox

Problem

Während meiner PAWI-Arbeit habe ich die Website studentbox.ch erstellt. Der Quellcode dieser Applikation wird mit SonarQube einer statischen Codeanalyse unterzogen. SonarQube zeigt folgende Übersicht über das Projekt an:



Dabei fällt auf, dass das Quality Gate nicht bestanden wurde, weil zwei kritische Issues vorhanden sind. Zudem hat das Projekt eine technische Schuld von einem Tag.

Ziel

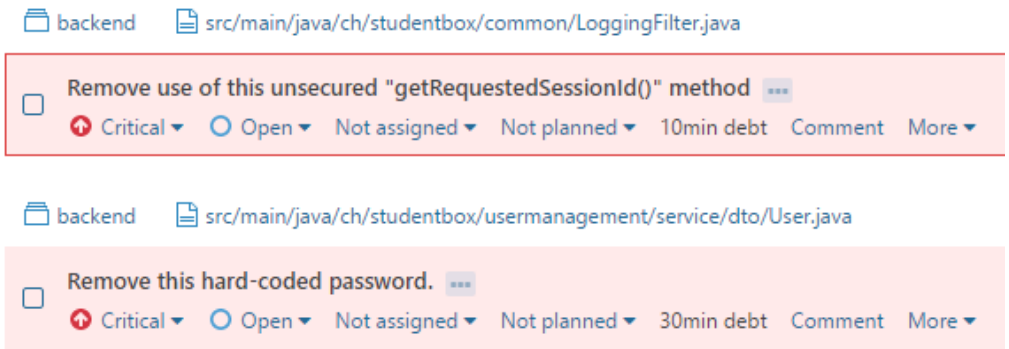
Das Ziel dieses Refactorings ist das Quality Gate zu bestehen und die technische Schuld des Projektes zu senken.

Messung des Fortschritts

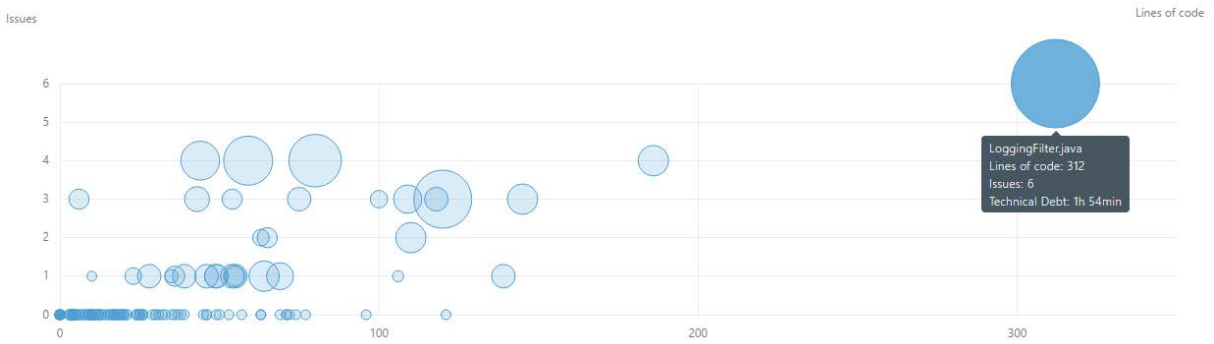
Während des Refactorings sollte sich die Anzahl der offenen Issues und die technische Schuld verkleinern.

Zustand vor Refactoring

Das Quality Gate wird nicht bestanden, weil zwei kritische Issues noch offen sind.



Beides sind Issues welche zu Sicherheitsproblemen führen können. Das nachfolgende Diagramm zeigt die Hotspots der technischen Schuld.



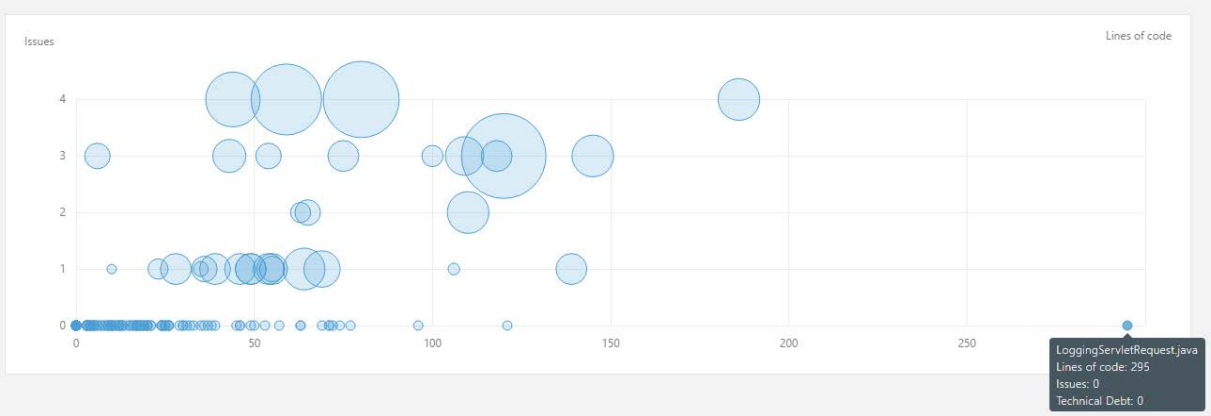
Dabei fällt besonders die Klasse oben rechts im Diagramm auf. Diese Klasse hat auch einen kritischen Issue offen welcher das Bestehen des Quality Gates verhindert. Bei der genaueren Analyse der Klasse fällt auf, dass für einen HttpServletRequest ein Wrapper als Inline-Klasse erstellt wurde.

```
//Needed to do this because when the Builder for the Sentry Logger calls the 'getUserPrincipal' method,
//If it's not set, one cannot correlate failed requests to users.
HttpServletRequest wrappedRequest = new HttpServletRequest() {

    private Principal principal = originalRequest.getUserPrincipal();


    @Override
    public String getAuthType() { return originalRequest.getAuthType(); }

    @Override
    public Cookie[] getCookies() { return originalRequest.getCookies(); }

    @Override
    public long getDateHeader(String name) { return originalRequest.getDateHeader(name); }

    @Override
    public String getHeader(String name) { return originalRequest.getHeader(name); }

    @Override
    public Enumeration<String> getHeaders(String name) { return originalRequest.getHeaders(name); }

    @Override
    public Enumeration<String> getHeaderNames() { return originalRequest.getHeaderNames(); }
```

SonarQube hat auch entsprechende Issues für diese Klasse erstellt.

Zustand nach Refactoring

Nachdem die innere Klasse in eine eigenständige Klasse ausgelagert wurde, haben sich die technischen Schulden dieser Klasse auf 0 reduziert.



Diese Beobachtung spiegelt sich auch in SonarQube wieder, wo die Klasse nicht mehr aus dem Diagramm hervorsticht. Dadurch dass mehrere kritische Issues gelöst wurden, konnte auch das Quality Gate bestanden werden.

Verwendete Refactorings

Um die innere anonyme Klasse zu extrahieren, wurde das *Extract Class* Refactoring angewendet.

Fazit

Das Analyse-Tool SonarQube bietet eine optimale Plattform um Schwachstellen im Code zu erkennen und diese mittels Refactorings zu beheben. In einem nächsten Schritt werde ich versuchen die Issues auf 0 herunterzukriegen. Wenn dann ein neuer Issue auftritt kann man den Build fehlschlagen lassen und der Entwickler ist gezwungen den Issue zuerst zu beheben, bevor der Build erfolgreich durchläuft. Dadurch kann eine optimale Codequalität garantiert werden.