Projektbeschreibung Refactoring Messdaten-Verwaltung

Entwicklungsteam:

Patrick Stoffel und Hans-Jürg Nett

Ausgangslage:

Im Rahmen der Semesterarbeit Web und Mobile Frontend Entwicklung wurde durch oben erwähntes Entwicklungsteam und Cla Tschenett das Projekt Messdaten-Verwaltung umgesetzt.

Die Applikation wurde für den produktiven Einsatz bei der Firma Trumpf in Grüsch geplant, um Produktionsabläufe zu automatisieren.

Das Projekt wurde serverseitig mit dem Play-Framework in Java implementiert und browserseitig durch HTML5, CSS durch JavaScript ergänzt. Diese Technologien wurden gewählt, da für die Umsetzung des Projekts Techniken eingesetzt werden sollten, die während des Semesters behandelt wurden.

Die Firma Trumpf setzt im betroffenen Bereich bei selbst entwickelten Applikationen hauptsächlich .Net mit C# ein und möchte aus Gründen des Knowhows und der Kompatibilität kein Java verwenden.

Aus diesem Grund wurde das Projekt durch Hans-Jürg Nett in seiner Funktion als Software-Entwickler bei der Firma Trumpf nach .Net als Web API portiert.
Dabei wurde das Framework ASP.Net mit C# als Projektgrundlage gewählt und das Frontend wurde wieder mit HTML5, CSS und JavaScript umgesetzt.

Bei der Portierung wurde im ersten Schritt der Fokus auf die Übernahme der Funktionalität und deren teilweise Anpassung gelegt. Die Klassenhierarchie und das logische Konzept sind mehrheitlich erhalten geblieben, die Tests fehlen aber noch komplett.

Aufgabe:

Unsere Semesterarbeit zielt nun darauf ab, das portierte Projekt Messdaten-Verwaltung auf Basis der in diesem Semester erlernten Themen wie Clean-Code, Design-Pattern oder Testautomatisierung zu überarbeiten.

Es soll keine neue Funktionalität implementiert werden, ausser es ist aus Gründen des Refactorings notwendig.

Ziel der Semesterarbeit ist, das Projekt besser lesbar, wartungsfreundlich und erweiterbar zu gestalten.

Durch die Implementierung entsprechender Tests sollen zukünftige Arbeiten am Projekt abgesichert werden können.

Termine:

Projektstart:	22.11.2017
Sprint Planning:	08.12.2017
Abgabe Inkrement:	30.01.2018
Präsentation:	01.02.2018

Mockups:

Es wird auf Mockups verzichtet, da keine Änderungen am Frontend vorgesehen sind.

User Storys:

ID	User Story
1	Als Entwickler möchte ich, dass der Code für Menschen als Ablauf lesbar ist, damit ich auf möglichst viele Kommentare verzichten kann.
2	Als Entwicklungs-Teamleiter möchte ich, dass die Applikation modular aufgebaut ist, damit Code-Duplizierung vermieden werden kann.
3	Als Entwicklungs-Teamleiter möchte ich, dass die Softwarequalität mit Hilfe eines Tools für kontinuierliche Integration gesteigert wird.
4	Als Entwickler möchte ich meine Klassen und Methoden testbar machen, damit die korrekte Funktion durch Regressionstests abgesichert werden kann.
5	Als Entwickler möchte ich in meinem Code auf geringe Kopplung achten, damit möglichst wenig Abhängigkeiten zwischen Klassen entstehen.
6	Als Entwickler möchte ich in meinem Code auf möglichst hohe Kohäsion achten, damit die Verantwortlichkeit und Aufgabe der Klasse/Methode klar definiert ist.
7	Als Entwickler möchte ich in der Applikation ein sauberes Error-Handling implementieren, damit Fehler abgefangen und der User informiert werden kann.
8	Als Entwickler möchte ich das Projekt in eine Web-Applikation zur Konfiguration der Geräte und eine Web-API zum Lesen der Messwerte aufteilen. Damit kann die Web-Applikation zur Konfiguration einmalig zentral und die Web- API auf jedem Host mit angebundenen Messgeräte installiert werden.

Produktbacklog:

ID User Story	Priority	Estimate	Business Value	Task	Sprint	Status
1	1	13	20	Benennung von Variablen auf Lesbarkeit, Verständlichkeit und Ausdruckskraft prüfen.	1	in progress
1	1	20	30	Benennung Klassen und Methoden prüfen, ob sie mit deren Inhalt korrespondieren.	1	in progress
1	1	13	20	Kommentare überarbeiten und wo möglich löschen.	1	in progress
7	1	20	30	Fehler-Handling implementieren, um Framework-Exceptions zu wrappen und Null-Rückgabewert zu vermeiden.	1	in progress
4	1	20	30	Unit-Tests für öffentliche Methoden implementieren.	1	in progress
4	1	20	20	Öffentliche Methoden testbar machen.	1	in progress
3	1	15	30	Geeignetes Tool für die kontinuierliche Integration evaluieren und mit der Solution verknüpfen.	1	in progress
6	1	20	20	Methoden auf Abstraktionslevel prüfen.	1	in progress
2	1	13	20	Projekt auf Möglichkeiten der Elimination von dupliziertem Code prüfen.	1	in progress
5	1	13	20	Klassen auf Abhängigkeiten prüfen und wo möglich überarbeiten.	1	open
8	2	20	20	Architektur konzipieren, um das Projekt in eine Web-API und eine Web-Applikation zu trennen.	2	open

Prinzip Skizze: