Maintenance Monitor

SLM Semesterprojekt

Bernhard Bauer, Julian Rektenwald, Berkay Yalcinkaya

Inhaltsverzeichnis

[Repository URL 3](#_Toc120970990)

[Screenshots 3](#_Toc120970991)

[Readme erstellen 3](#_Toc120970992)

[Issues 4](#_Toc120970993)

[Kanban-Board 4](#_Toc120970994)

[Programm Development 6](#_Toc120970995)

[Programm Testing 6](#_Toc120970996)

[Postman 6](#_Toc120970997)

[Testfunktionen 8](#_Toc120970998)

[Actions 8](#_Toc120970999)

[Programmdurchlauf 8](#_Toc120971000)

# Repository URL

<https://github.com/bua02/SLM_Projekt_Monitor.git>

(du solltest eh schon eingeladen sein)

# Screenshots

## Readme erstellen

Ein Bild, das Text enthält.

Automatisch generierte Beschreibung

Die Readme wurde automatisch erstellt und vom Team bearbeitet – es wird die grundlegende Funktionsweise sowie das Setup erklärt.

## Issues/User-Stories

Ein Bild, das Text enthält.

Automatisch generierte Beschreibung

Das ist eine Liste von einigen der im Laufe des Projektes erstellten Issues/User-Stories. (siehe [hier](https://github.com/bua02/SLM_Projekt_Monitor/issues)).

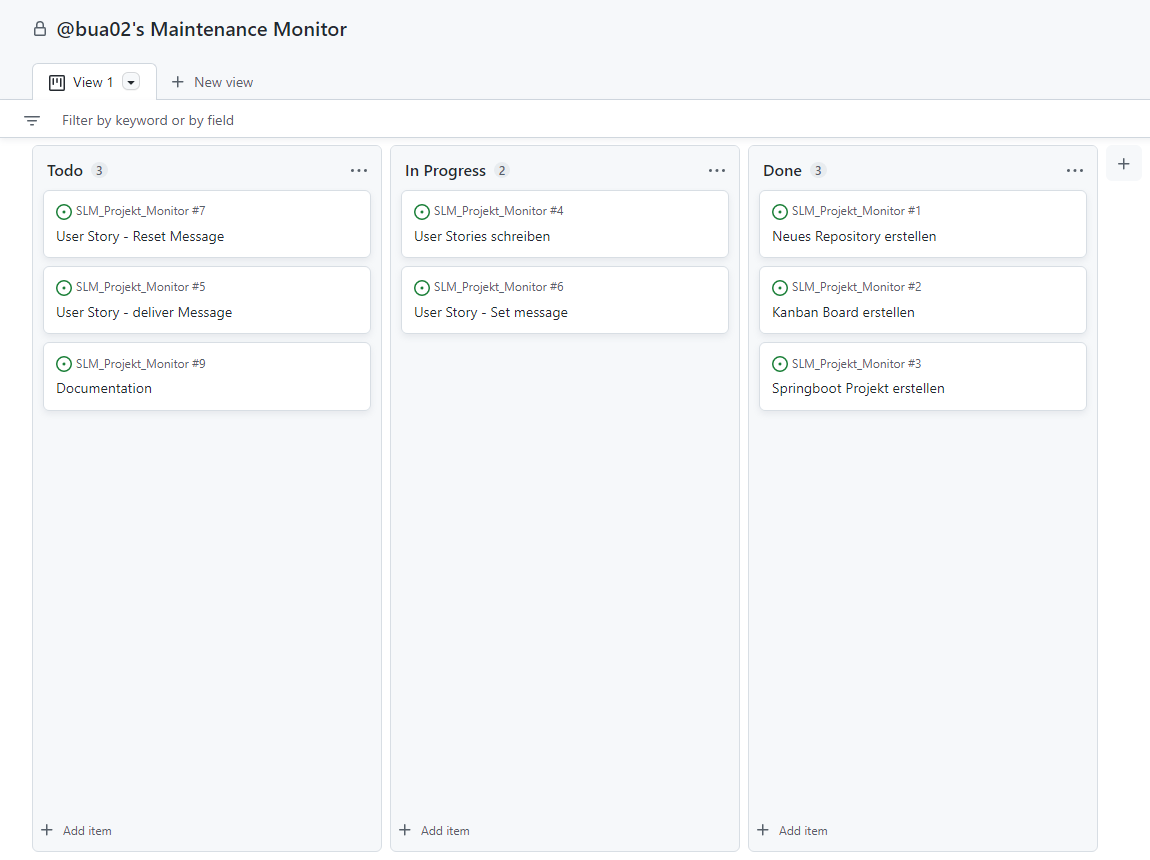
Ein Bild, das Text enthält.

Automatisch generierte Beschreibung

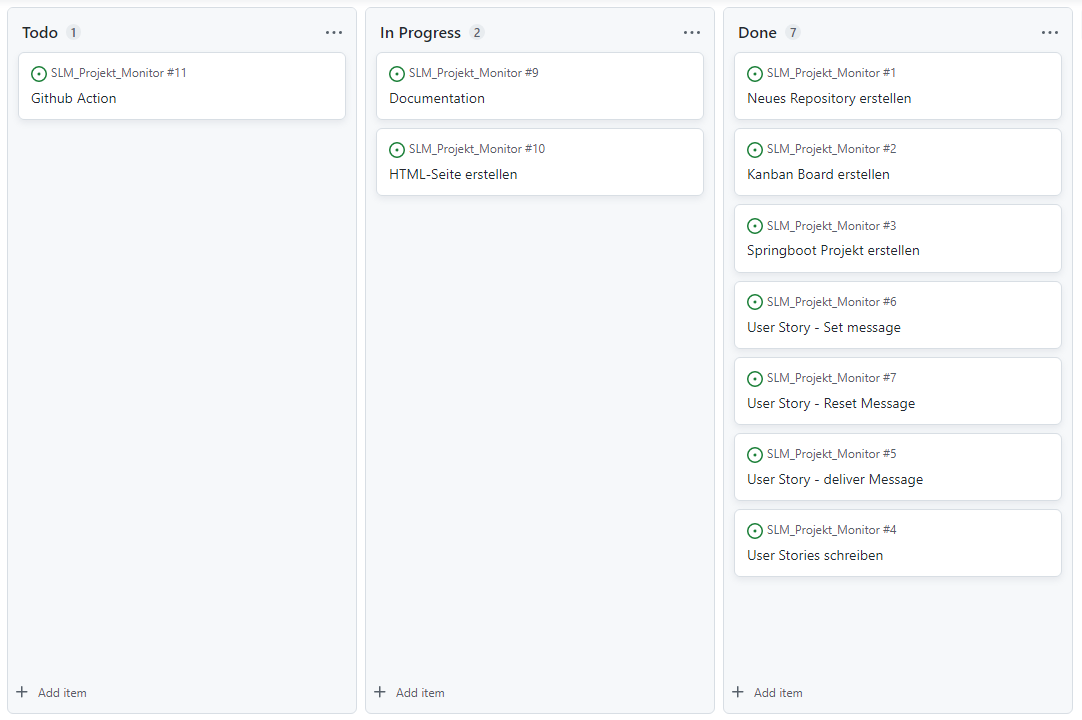
User-Stories/Issues wurden, wie in diversen LVs geübt, mit drei großen Überschriften „Soll“, „Abgrenzungen“ und „Akzeptanzkriterien“ erstellt.

## Kanban-Board

Dieselben User-Stories wurden anschließend in das Kanbanboard übertragen:



Anhand des Kanbanboards war zu erkennen wie das Projekt fortschreitet:



## Programm Development

Folgende Klassen wurden erstellt (auch in den Github logs zu sehen ☺):

* SlmProjektMonitorApplication (von Spring-Boot erstellt)
* Monitor

Model, welches die Daten speichert und Basisfunktionalität anbietet

Attribute:

* + lastUpdate – speichert den Zeitpuntk der letzten Bearbeitung des Monitors
  + monitorMessage – Speichert die aktuelle Wartungsnachricht

Methoden:

* + Set() – setter für die monitorMessage, updatet lastUpdate
  + Reset() – settet monitorMessage leer, updatet lastUpdate
  + getCurrentTime() – gibt die aktuelle Zeit zurück
  + getLastUpdate() – getter für lastUpdate
  + getMonitorMessage()– Wenn eine Message vorhanden ist -> returned sie die Message, sonst returned die Methode die EVERYTHING\_WORKS\_AS\_EXPECTED\_MESSAGE



* MonitorController

Controller, der die Funktionalität des Monitors implementiert hat.

Attribute:

* + Monitor – das Monitor-Objekt, mit dem der Controller arbeitet.

Methoden:

* + getMessage() – Liefert die aktuelle Message des Monitors, im Falle von Null oder keiner Message liefert die Methode „No message set“
  + reset() – resettet die aktuelle Message des Monitors
  + setMessage() – setted die aktuelle Message des Controllers
  + showMonitor() – Returned das HTML für die Startseite des Monitors



* MessageUtil

Klasse mit den final-Strings die der Monitor an mehreren Stellen zurückgibt (damit Strings nicht öfter gehardcoded werden)

Attribute:

* + EVERYTHING\_WORKS\_AS\_EXPECTED\_MESSAGE
  + OK\_MESSAGE



* TextContent

Liefert das HTML für die Starseite zurück

Methoden:

* + getRedMonitor() – gibt das HTML für den Monitor mit rotem Background zurück
  + getGreenMonitor() – gibt das HTML für den Monitor mit grünem Background zurück



## Programm Testing

### Testing der Funktionalität mittels Postman

Ein Bild, das Text enthält.

Automatisch generierte Beschreibung

Ein Bild, das Text enthält.

Automatisch generierte Beschreibung

Ein Bild, das Text enthält.

Automatisch generierte Beschreibung

Ein Bild, das Text enthält.

Automatisch generierte Beschreibung

Ein Bild, das Text enthält.

Automatisch generierte Beschreibung

### Testing mittels Unit-Tests

Die Unit-Tests wurden implementiert und testen die Funktionen des Monitors. Sie sind auf Github verfügbar. Beim Ausführen sieht man, dass alle Unit-Tests positiv abschließen.

Ein Bild, das Text enthält.

Automatisch generierte Beschreibung

## Actions

# Programmdurchlauf

Beim ersten Einstieg In den Monitor (über localhost:8080):

Ein Bild, das Text enthält.

Automatisch generierte Beschreibung

Wartungsmeldung hinzufügen ([http://localhost:8080/api/message/set?m=Wartung+bis+10:00)](http://localhost:8080/api/message/set?m=Wartung+bis+10:00)#)

Er zeigt ok an und die Zeit, wenn man jetzt wieder auf die Startseite geht sieht man das hier:Ein Bild, das Text enthält.

Automatisch generierte Beschreibung

Jetzt kann man über (<http://localhost:8080/api/message/reset>) die Nachricht wieder löschen und er zeigt:



Wenn man jetzt wieder die Startseite begutachtet sieht man: Ein Bild, das Text enthält.

Automatisch generierte Beschreibung

Die Zeiten werden immer geupdatet, wenn man set oder reset durchführt, damit man immer sehen kann, wann der Monitor (die Message) das letzte Mal bearbeitet wurde.