

Maintenance Monitor

SLM Semesterprojekt

Bernhard Bauer, Julian Rektenwald, Berkay Yalcinkaya

Inhaltsverzeichnis

Repository URL	3
Screenshots	3
Readme erstellen	3
Issues	4
Kanban-Board	5
Programm Development	6
Programm Testing	12
Postman	Fehler! Textmarke nicht definiert.
Testfunktionen	Fehler! Textmarke nicht definiert.
Actions	15
Programmdurchlauf	15



Repository URL

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

(du solltest eh schon eingeladen sein)

Screenshots

Readme erstellen

 README.md 

SLM_Projekt_Monitor

Github Repository für das Semesterprojekt in SLM

Es soll ein Maintenance Monitor programmiert werden, der den derzeitigen Zustand des Kraftwerks darstellt.

Wenn alles funktioniert soll "Everything operates as expected" ausgegeben werden. Dabei soll es den Wartungsarbeitern möglich sein die Nachricht manuell zum Beispiel auf "Service checks: No power until 5:00 pm" zu ändern. Anschließend sollte die Nachricht wieder resettet werden.

Setup

Auf dem main Branch ist der letzte funktionierende und getestete Stand, der development Branch wird von den Entwicklern verwendet.

Das Github-Repo lokal clonen und dann den Ordner in IntelliJ (oder einem anderen Programm) öffnen. Danach sollte die Spring-Boot application über die Klasse "SlmProjektMonitorApplication" gestartet werden.

Funktion

Über localhost:8080/api/message kann man dann den aktuellen Stand des Maintenance-Monitors abrufen. Über /api/message/set?m=Wartung kann man eine neue Wartungsmeldung hinzufügen Über /api/message/reset kann man die aktuelle Wartungsmeldung wieder löschen

Beim setzen und resettet wird die aktuelle Zeit aktualisiert, damit man sehen kann wann die Seite das letzte mal bearbeitet wurde.

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

Issues/User-Stories

<input type="checkbox"/>	10 Open ✓ 0 Closed	Author ▾	Label ▾	Projects ▾	Milestones ▾	Assignee ▾	Sort ▾
<input type="checkbox"/>	Github Action #11 opened 28 minutes ago by bua02						
<input type="checkbox"/>	HTML-Seite erstellen #10 opened 30 minutes ago by bua02						
<input type="checkbox"/>	Documentation #9 opened 1 hour ago by bua02						
<input type="checkbox"/>	User Story - Reset Message #7 opened 1 hour ago by bua02						
<input type="checkbox"/>	User Story - Set message #6 opened 1 hour ago by bua02						
<input type="checkbox"/>	User Story - deliver Message #5 opened 1 hour ago by bua02						
<input type="checkbox"/>	User Stories schreiben #4 opened 1 hour ago by bua02						
<input type="checkbox"/>	Springboot Projekt erstellen #3 opened 2 hours ago by bua02						
<input type="checkbox"/>	Kanban Board erstellen #2 opened 2 hours ago by bua02						
<input type="checkbox"/>	Neues Repository erstellen #1 opened 3 days ago by bua02						

Das ist eine Liste von einigen der im Laufe des Projektes erstellten Issues/User-Stories. (siehe [hier](#)).



bua02 commented 1 hour ago • edited ▾
Owner
😊
⋮

Soll

Es soll möglich sein eine Nachricht in das Terminal einzugeben, die anschließend auf dem Maintenance Monitor angezeigt wird.

Abgrenzungen

- Es werden nur ganze Strings geschickt
- Eingeebene Strings müssen nicht ausgewertet werden
- Kein Errorhandling für falsche User-Eingaben
- Keine Fehlermeldungen notwendig

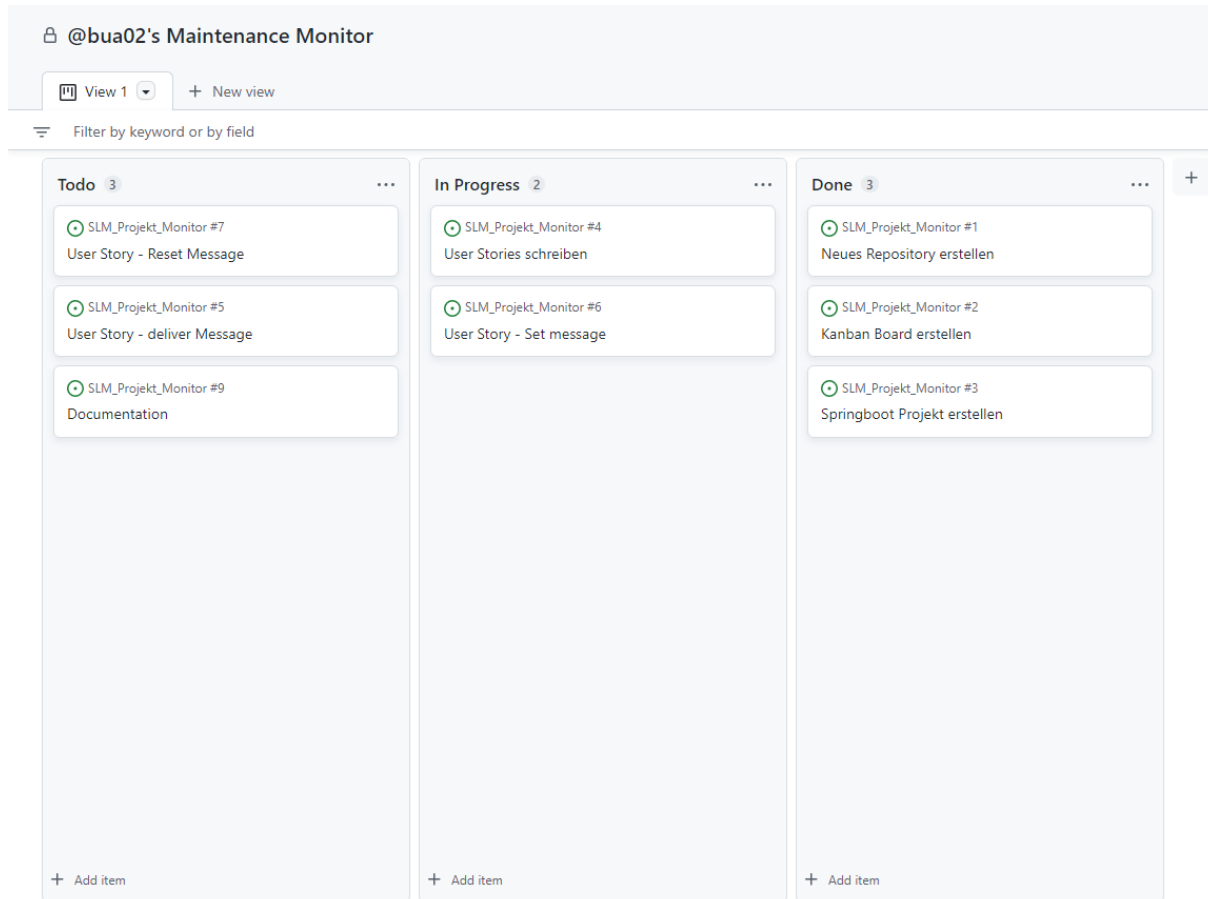
Akzeptanzkriterien / Tesntinput

- `/api/message/set?m=Service+checks:+No+power+until+5:00+pm -> "Service checks: No power until 5:00 pm`
- `/api/message/set?m=Error+345 -> "Error 345"`

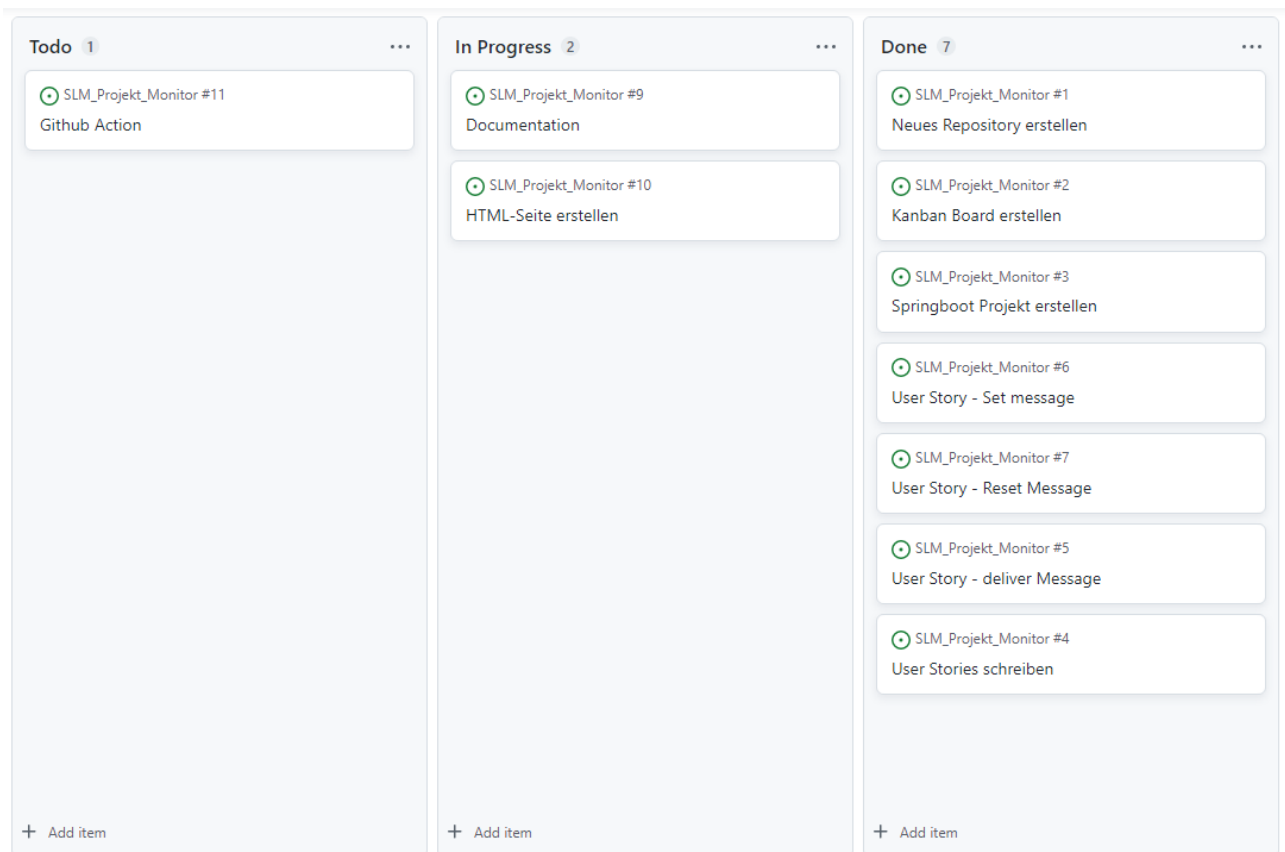
User-Stories/Issues wurden, wie in diversen LVs geübt, mit drei großen Überschriften „Soll“, „Abgrenzungen“ und „Akzeptanzkriterien“ erstellt.

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 ☺):

- SImProjektMonitorApplication (von Spring-Boot erstellt)
- Monitor
 Model, welches die Daten speichert und Basisfunktionalität anbietet
 Attribute:
 - lastUpdate – speichert den Zeitpunkt 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

```
package com.fhtw.slm_projekt_monitor;

import java.text.SimpleDateFormat;

import static
com.fhtw.slm_projekt_monitor.MessageUtil.EVERYTHING_WORKS_AS_EXPECTED_MESSA
GE;
import static com.fhtw.slm_projekt_monitor.MessageUtil.OK_MESSAGE;

public class Monitor {

    private String monitorMessage;
    private String lastUpdate = getCurrentTime();

    public Monitor() {
        this.monitorMessage = "";
    }

    public Monitor(String monitorMessage) {
        this.monitorMessage = monitorMessage;
    }

    public String set(String monitorMessage) {

        this.monitorMessage = monitorMessage;
        this.lastUpdate = getCurrentTime();

        return OK_MESSAGE + "\n" + this.lastUpdate;
    }

    public String reset() {

        this.monitorMessage = "";
        this.lastUpdate = getCurrentTime();

        return OK_MESSAGE;
    }

    private String getCurrentTime() {
        return new SimpleDateFormat("HH:mm:ss").format(new
java.util.Date());
    }

    public String getMonitorMessage() {

        if (this.monitorMessage.length() == 0) {
            return EVERYTHING_WORKS_AS_EXPECTED_MESSAGE;
        }

        return this.monitorMessage;
    }

    public String getLastUpdate() {
        return this.lastUpdate;
    }
}
```

- **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


```
package com.fhtw.slm_projekt_monitor;

import org.springframework.http.MediaType;
import org.springframework.stereotype.Controller;
import org.springframework.web.bind.annotation.*;

import static
com.fhtw.slm_projekt_monitor.MessageUtil.EVERYTHING_WORKS_AS_EXPECTED_MESSAGE;

@Controller
public class MonitorController {
    private final Monitor monitor = new Monitor();

    @RequestMapping(value = "/api/message", method = RequestMethod.GET)
    @ResponseBody
    public String getMessage() {
        if (monitor.getMonitorMessage() == null ||
monitor.getMonitorMessage().isEmpty()) {
            return "No message set";
        }

        return monitor.getMonitorMessage();
    }

    @RequestMapping(value = "/api/message/reset", method =
RequestMethod.GET)
    @ResponseBody
    public String reset() {
        return monitor.reset();
    }

    @RequestMapping(value = "/api/message/set", method = RequestMethod.GET)
    @ResponseBody
    public String setMessage(@RequestParam(name= "m") String
monitorMessage) {
        return monitor.set(monitorMessage);
    }

    @RequestMapping(value = "/", produces = MediaType.TEXT_HTML_VALUE)
    @ResponseBody
    public String showMonitor() {
        if
(monitor.getMonitorMessage().equals(EVERYTHING_WORKS_AS_EXPECTED_MESSAGE))
{
            return TextContent.getGreenMonitor(monitor.getMonitorMessage(),
monitor.getLastUpdate());
        }
        return TextContent.getRedMonitor(monitor.getMonitorMessage(),
monitor.getLastUpdate());
    }
}
```

- 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

```
package com.fhtw.slm_projekt_monitor;
```

```
public class MessageUtil {  
    public static final String EVERYTHING_WORKS_AS_EXPECTED_MESSAGE =  
    "Everything works as expected";  
    public static final String OK_MESSAGE = "ok";  
}
```

- **TextContent**

Liefert das HTML für die Startseite zurück

Methoden:

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

```
package com.fhtw.slm_projekt_monitor;

public class TextContent {
    public static String getRedMonitor(String message, String lastUpdate) {
        return String.format("<html lang=\"en\">\n" +
            "<head>\n" +
            "    <meta charset=\"UTF-8\">\n" +
            "    <title>Monitor</title>\n" +
            "</head>\n" +
            "<body style=\"background-color:#D22B2B;\">\n" +
            "\n" +
            "<h1 style=\"text-align:center; color:white;\">Maintenance
Monitor</h1>\n" +
            "<p style=\"text-align:center; color:white;\">%s</p>\n" +
            "<p style=\"text-align:center; color:white;\">last update:
%s</p>\n" +
            "\n" +
            "</body>\n" +
            "</html>", message, lastUpdate);
    }

    public static String getGreenMonitor(String message, String lastUpdate)
    {
        return String.format("<html lang=\"en\">\n" +
            "<head>\n" +
            "    <meta charset=\"UTF-8\">\n" +
            "    <title>Monitor</title>\n" +
            "</head>\n" +
            "<body style=\"background-color:#32CD32;\">\n" +
            "\n" +
            "<h1 style=\"text-align:center; color:white;\">Maintenance
Monitor</h1>\n" +
            "<p style=\"text-align:center; color:white;\">%s</p>\n" +
            "<p style=\"text-align:center; color:white;\">last update:
%s</p>\n" +
            "\n" +
            "</body>\n" +
            "</html>", message, lastUpdate);
    }
}
```

Programm Testing

Testing der Funktionalität mittels Postman

The screenshot shows the Postman interface with a workspace named 'SLM / Summenrechner Abfrage'. A GET request is configured to 'localhost:8080/api/message'. The 'Query Params' section is empty. The 'Body' tab is selected, showing a 'Pretty' view with the response: '1 Everything works as expected' and '2 10:41:06'. The status bar indicates 'Status: 200 OK', 'Time: 57 ms', and 'Size: 201 B'.

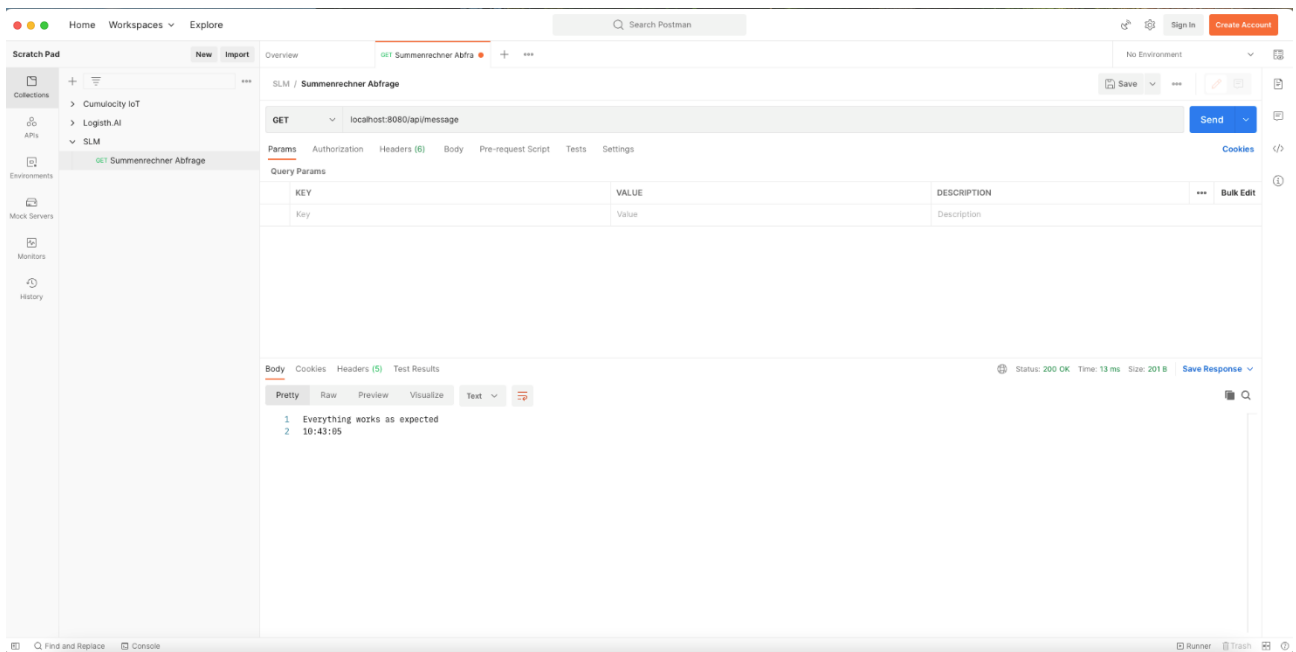
The screenshot shows the Postman interface with the same workspace. The GET request is now configured to 'localhost:8080/api/message?m=Wartung+bis+ca+10:00'. The 'Query Params' section contains one parameter: 'm' with the value 'Wartung+bis+ca+10:00'. The 'Body' tab is selected, showing a 'Pretty' view with the response: '1 ok' and '2 10:41:55'. The status bar indicates 'Status: 200 OK', 'Time: 20 ms', and 'Size: 175 B'.

Postman interface showing a GET request to `localhost:8080/api/message`. The response status is 200 OK, and the body contains the following text:

```
1 Wartung bis ca 10:00
2 10:42:49
```

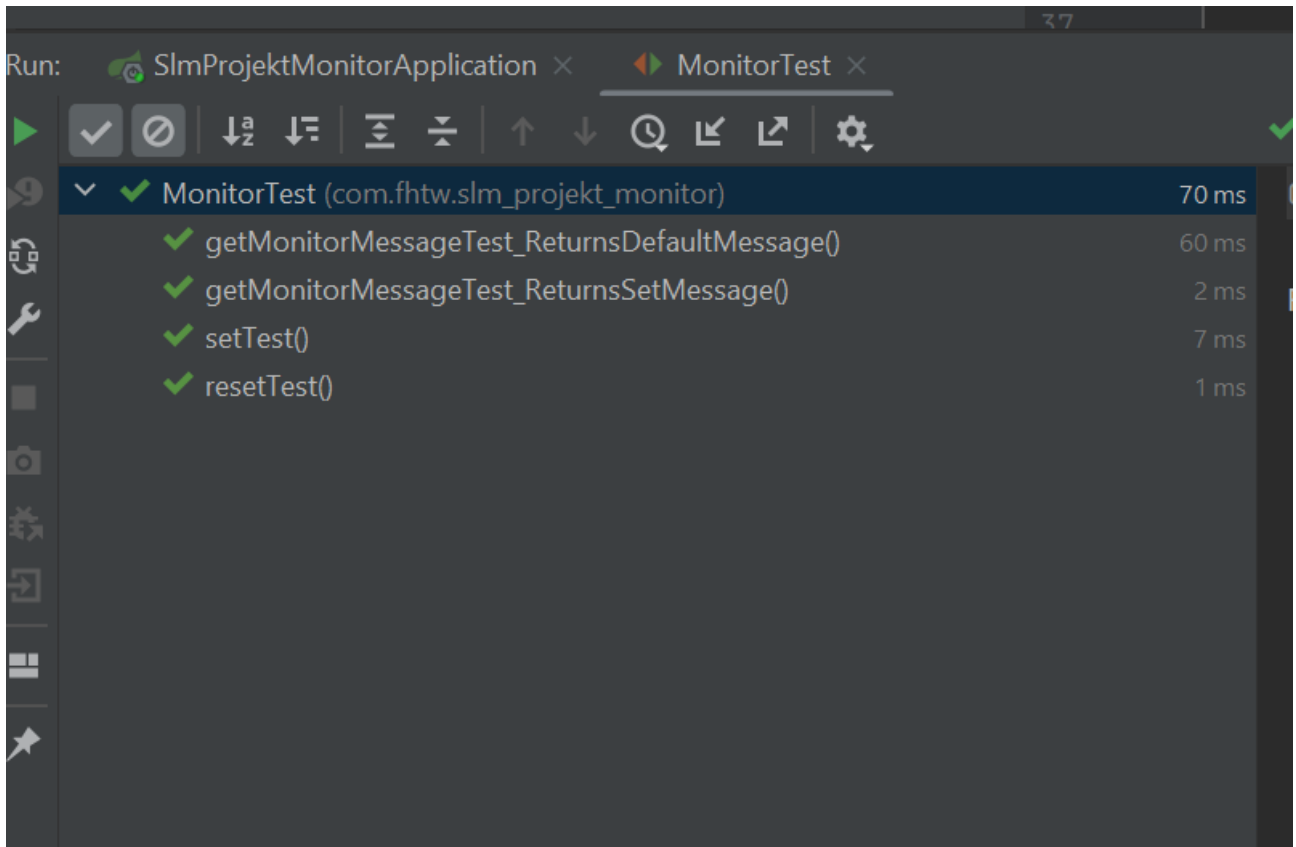
Postman interface showing a GET request to `localhost:8080/api/message/reset`. The response status is 200 OK, and the body contains the following text:

```
1 ok
2 10:43:05
```



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.



Actions

Programmdurchlauf

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



Wartungsmeldung hinzufügen (<http://localhost:8080/api/message/set?m=Wartung+bis+10:00>)
ok 15:26:13

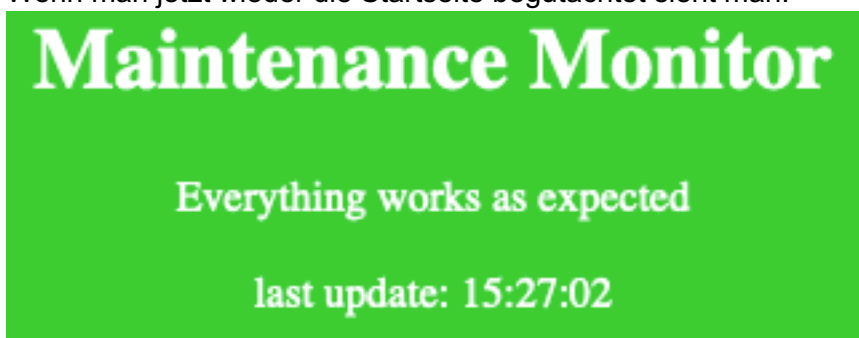
Er zeigt ok an und die Zeit, wenn man jetzt wieder auf die Startseite geht sieht man das hier:



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

ok

Wenn man jetzt wieder die Startseite begutachtet sieht man:



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.