

Software Lifecycle Management

<Quadrat> (e.g., Capital API)

<A client wants an API to calculate the square of a number> (e.g., We are a small Geo Company in Vienna. A client wants an API ...)

Project

You should implement a REST-based server in Java (use [Spring Boot](#)). The service should be able to return the desired information using REST.

Requirements

Use GitHub or Azure DevOps for the project and follow the correct DevOps procedure. Use a Kanban board to manage your User Stories and use a git branching model (preferable gitflow) to manage your code. Track your development process by updating your Kanban board and write at least one unit tests for every requirement. A Continuous Integration pipeline that produces the finished software artifact should be implemented as well.

Document

- the whole process
- the user stories
- the repository URL
- the usage of the software

in a Readme file with explanatory text. Submit the code (including the .git folder and ALM files) as a zip file (please put the Readme inside the zip file).

You can use external resources as long as you mark them: “ // taken from: <URL> ”

Points

- Documentation of the process: 15%
- Requirement definitions (User Stories): 15%
- Correct status / Linking / Branching (Kanban, Git): 15%
- Implementation: 15%
- Testing: 15%
- Pipeline (Continuous Integration and Maven): 15%
- Artefacts (Continuous Delivery): 10%

All elements must be present in the documentation.

References

</api/square?v=10 -> 100>

(e.g., api/capital/Austria → Vienna)

Emre Yesildag ic22b076

GitHub Repository: <https://github.com/eyesildag/SLMTestQuadratYesildag>

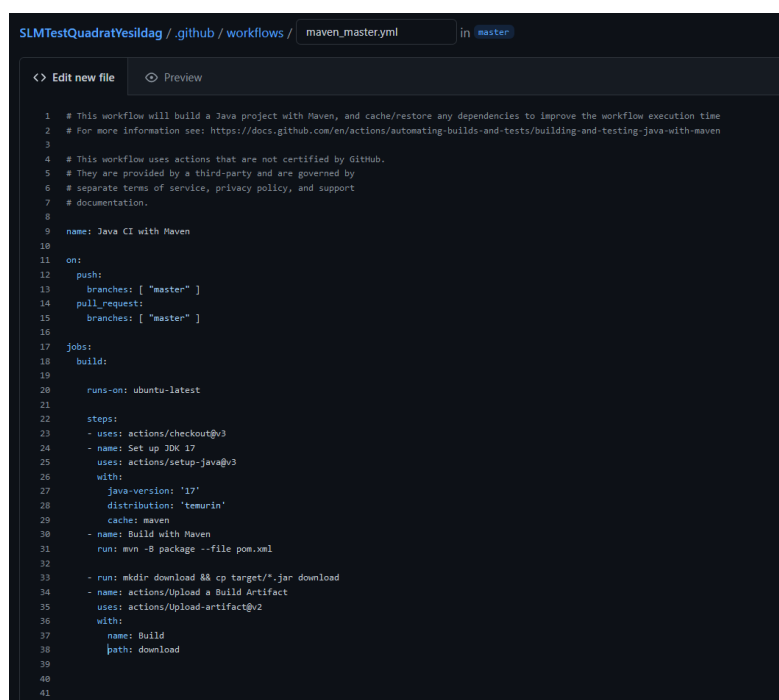
Durchführung des Tests

In IntelliJ ein neues Projekt mit Spring Web anlegen. Language Java, Type: Maven, JDK 17. Und gleichzeitig ein Git Repository miterstellen.

Nach der Erstellung des Projekts wird dieses auf GitHub geshared. Hauptmenü -> Git -> GitHub -> Share Project on GitHub. Dabei wird alles committed und gepusht.

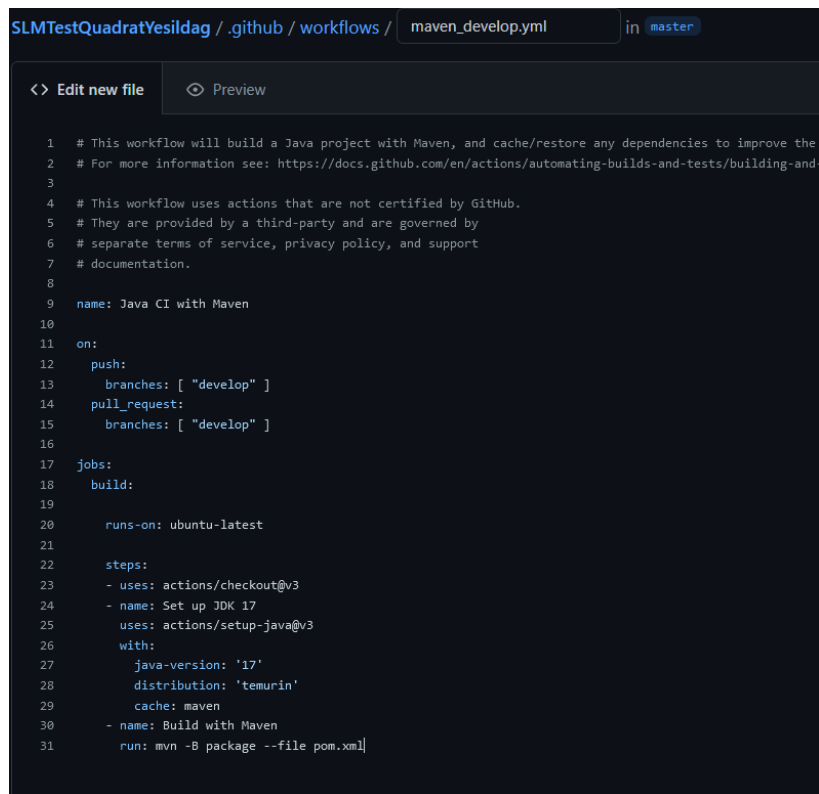
Nun werden auf GitHub die Workflows für den Master und Develop Branch erstellt.

So sieht das File für den Master aus



```
1 # This workflow will build a Java project with Maven, and cache/restore any dependencies to improve the workflow execution time
2 # For more information see: https://docs.github.com/en/actions/automating-builds-and-tests/building-and-testing-java-with-maven
3
4 # This workflow uses actions that are not certified by GitHub.
5 # They are provided by a third-party and are governed by
6 # separate terms of service, privacy policy, and support
7 # documentation.
8
9 name: Java CI with Maven
10
11 on:
12   push:
13     branches: [ "master" ]
14   pull_request:
15     branches: [ "master" ]
16
17 jobs:
18   build:
19     runs-on: ubuntu-latest
20
21     steps:
22     - uses: actions/checkout@v3
23     - name: Set up JDK 17
24       uses: actions/setup-java@v3
25       with:
26         java-version: '17'
27         distribution: 'temurin'
28     - name: Build with Maven
29       run: mvn -B package --file pom.xml
30
31     - run: mkdir download && cp target/*.jar download
32     - name: Upload a Build Artifact
33       uses: actions/upload-artifact@v2
34       with:
35         name: Build
36         path: download
37
38
39
40
41
```

So sieht das File für den develop aus:

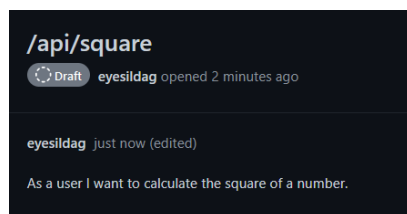


```
1 # This workflow will build a Java project with Maven, and cache/restore any dependencies to improve the
2 # For more information see: https://docs.github.com/en/actions/automating-builds-and-tests/building-and-
3
4 # This workflow uses actions that are not certified by GitHub.
5 # They are provided by a third-party and are governed by
6 # separate terms of service, privacy policy, and support
7 # documentation.
8
9 name: Java CI with Maven
10
11 on:
12   push:
13     branches: [ "develop" ]
14   pull_request:
15     branches: [ "develop" ]
16
17 jobs:
18   build:
19
20     runs-on: ubuntu-latest
21
22     steps:
23     - uses: actions/checkout@v3
24     - name: Set up JDK 17
25       uses: actions/setup-java@v3
26       with:
27         java-version: '17'
28         distribution: 'temurin'
29         cache: maven
30     - name: Build with Maven
31       run: mvn -B package --file pom.xml]
```

Wenn man mit dem Bearbeiten der Files fertig ist klicke ich am Ende auf Start Commit und kann diese dann von GitHub pullen. Wichtig bei der Bearbeitung war die Java Version richtig einzustellen und beim master die Erstellung vom Artifact mitzunehmen. Zusätzlich wurde der optionale Code am Ende wird entfernt um Probleme zu vermeiden.

Nun kehre ich in meine IDE zurück und pulle die neuen Files. Jetzt erstelle ich mir vom aus einen neuen Branch namens develop. Den neuen Branch pushe ich gleich auf GitHub.

Auf GitHub navigiere ich auf Projects und erstelle mir ein neues Board. Weiters setze ich diesen auf Public damit dieser dann am Ende auch für andere ersichtlich ist. Ich erstelle eine User Story.



Das Feature befindet sich anfangs auf Todo. Ich setze diesen in "In Progress", da ich diesen gleich implementieren werde.

Um an dem Feature nun auch arbeiten zu können muss ich vom develop Branch aus ein neuen Branch erstellen namens feature-square.

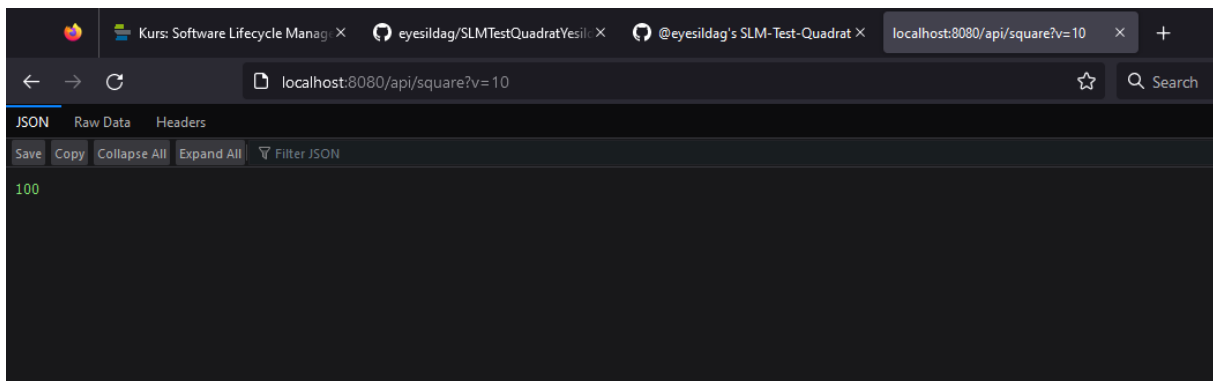
Nun benötige ich eine Controller Klasse um an den Features zu arbeiten. Ich erstelle mir eine neue Klasse namens SquareController. Bei der Klasse brauch ich die Annotation @RestController und bei der Feature Methode, welche ich square nennen werde die Annotation @GetMapping("/api/square").

Die Implementierung des Features ist nun auf dem feature-square Branch abgeschlossen und kann committed und auf GitHub gepusht werden.

Auf GitHub erscheinen nun pull requests. Ich muss einmal base:develop <- feature-square auswählen und beim zweiten mal base:master <- master. Das heisst einmal wird der feature branch in den develop und dann der develop in den master gemerged. Der feature Branch wird nicht mehr benötigt und GitHub schlägt mir vor diesen zu löschen, welches ich auch tu.

Ich navigiere mich zu den Actions und kontrolliere ob alles gepasst hat. Lade mir schliesslich die JAR-File runter und füge dem in die finale Abgabe hinzu.

Das Feature ist fertig implementiert und funktioniert. Am Board stelle ich das Feature auf "Done".



Auf Intellij Fetze ich nun und update den master und develop branch. Ich checke out auf den develop dann auf den master und kann nun jetzt den lokalen feature-square branch löschen, da dieser nicht mehr benötigt wird.

Am Ende erstelle ich auf GitHub ein Readme.md mit kleinen Informationen über den Test.