Software Lifecycle Management

Quadrieren

Web-Applikation berechnet das Quadrat einer Zahl, welches über die URL übergeben wird.

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): 20%
- Implementation: 15%
- Pipeline (Continuous Integration and Maven): 20%
- Artefacts (Continuous Delivery): 10%

All elements must be present in the documentation.

References

<square example>

api/square?m=5 → “25”

Testdurchführung

1. In IntelliJ ein neues Projekt erstellen unter „**File/New Project**“ und als Projekt „**Spring Initializr**“ auswählen. Unter den Konfigurationen „**JDK 17**“ und „Maven“ wählen. Als Dependencies „Spring Web“ auswählen.

2. Danach das Projekt auf GitHub hochladen unter „**VCS/Share Project on GitHub**“. Das Repository heißt „**TestHediyehlooQuadrieren**“

3. Jetzt wird eine Pipeline, um das für Projekt das Artefact erstellen zu können, ein „**master**“ Branch erstellt. Unter „**Actions**“, dann „**New Workflow**“ auf GitHub “Java with Maven” auswählen und auf “Configure“ drücken. Im Maven File muss die Java Version auf 17 verändert werden, der „optional“ Teil am Ende gelöscht werden und der folgende Code unten eingefügt werden (aufpassen mit den Einrückungen):

```
- run: mkdir download && cp target/*.jar download
 - name: actions/Upload a Build Artifact
   uses: actions/Upload-artifact@v2
    with:
     name: Build
     path: download
```

Danach ein „commit“

4. Jetzt wird eine neue Pipeline für den „**develop**“ Branch erstellt. Unter „Actions“ „Java with Maven“ auswählen und im Maven-File den „optional“ Teil löschen, Java Version auf 17 ändern und bei den „Branches“ „develop“ eingeben. Danach „commit“ in den Master

5. Das GitHub-Repository wieder Lokal auf IntelliJ pullen und prüfen ob die Workflows enthalten sind

6. Jetzt muss man auf der IDE ein neuen Branch erstellen vom Master-Branch, unter „New Branch from Selected“. Dieser neue Branch wird auf GitHub gepusht.

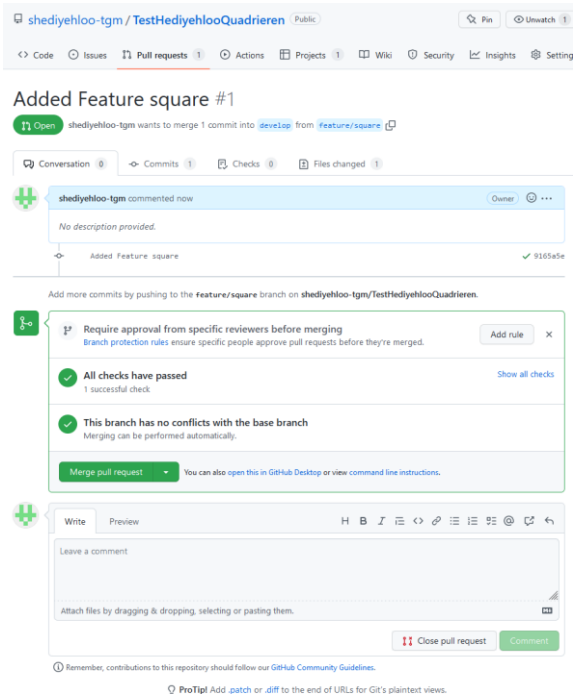
7. Jetzt muss man auf GitHub ein Projekt erstellen und dieses mit dem Repo verbinden. In der Leiste oben auf „Projects“ klicken und dann auf „New project“. Als Template „Board“ auswählen und auf „Create“ drücken. Darin kann man dann die ToDo’s für die Features definieren. Das Projekt muss unter den Einstellungen auf „Public“ gestellt werden. Das neue Feature wird auf „In Progress“ gestellt auf dem Board, da jetzt mit der Implementierung begonnen wird.

8. Für das Feature wird in IntelliJ ein neuer Branch aus dem „develop“ Branch erstellt, der neue Branch heißt „feature/square“. Damit man dies macht muss man wieder „New Branch from Selected“ auswählen aus dem „develop“ Branch.

9. Es wird eine neue Java Klasse, TestSquareController.java, für das Feature angelegt und gleichzeitig der „RestController“ definiert, dies gelingt mit der Notation „@RestController“. Für das Feature benötigt man für die Web-Applikation noch die Notation „@GetMapping“

10. Nachdem das Feature erstellt wurde, muss man es auf GitHub pushen.

11. Auf GitHub muss man dann ein „pull request“ ausführen. Im Tab „pull request“ auf „New pull request” klicken, danach 2x auf „Create Pull request” klicken. Der „feature/square“ Branch wird in den „develop“ gemerged wird. Danach auf „Merge pull request“ drücken. Wenn es gemerged wurde kann man den Feature-Branch löschen mit „Delete Branch“



12. Danach kann man zur Überprüfung die Pipeline für den „develop“ Branch überprüfen, ob alles gebuildet wurde
13. Wenn alles gepasst hat, muss man nur noch einen „Pull-Request“ vom „develop“ in den „Master“ Branch durchführen, dies führt auch die Pipeline aus um das JAR-File bzw. Artifact zu generieren.
14. Danach muss man unter „Actions“ das für den Master-Branch generierte JAR-File runterladen und das „ToDo“ in „Done“ verschieben im Projektboard auf GitHub.
15. In IntelliJ muss man nur noch die jeweiligen Branches updaten und in den Master „checkouten“, der nicht benötigte „feature/square“ Branch kann lokal gelöscht.
16. Danach wurde ein Readme erstellt auf GitHub.

Das Projekt ist unter dem Link erreichbar: https://github.com/shediyehloo-tgm/TestHediyehlooQuadrieren