Software Lifecycle Management

Scenario “Quadrat”

API welche es dem Nutzer ermöglicht, das Quadrat einer Zahl zu berechnen

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> ”

Jordi Rieder

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

api/square?n=10 → 100

Dokumentation

Link zum Repo: https://github.com/jrieder-tgm/square.git

1. Erstellen eines neuen Spring Initializr Projektes in Intellj. Maven auswählen und Spring Web bei den Dependencies auswählen.

2. Neues Repository für dieses Projekt erstellen. Dafür in Intellj auf VCS → Share Project in Github klicken. Aktuellen Stand pushen.

3. In Github auf Actions navigieren um einen einen Workflow/Piipeline für den Master zu definerien. Hierfür wird eine „Java with Maven“ Action konfiguriert. In der yaml Konfigurationsdatei muss die Java Version abgeändert werden. Weiters wird der optionale Code am wird des Files wird gelöscht und folgender Code wird eingefügt umd das Artefakt bei jedem Push zu automatisch erzeugen.

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

4. Definieren eines weiteren Workflows für den „develop“ Branch. Hier wird lediglich die Java Version sowie der Name des Branches im yaml File geändert. Weiters wird der optinale Code am Ende des Files gelöscht.

5. Repository lokal am Computer pullen

6. Neuen „develop“ Branch vom Master aus erstellen. (Master → new Branch from selected) und diesen pushen.

7. Nun wird das Board erstellt, hierfür muss in Github auf Projects → New project navigiert werden. Anschließend muss das Projekt in den Settings auf Public gestellt werden.

8. Auf dem konfigurierten Board wird eine neue Userstory /api/suqare mit Beschreibung erstellt und direkt auf „In Progress“ verschoben da jetzt die Implementierung des Features beginnt.

9. In Intellj wird ausgehend von develop Branch ein neuer feature Branch für die User Story erstellt und direkt in diesen gecheckoutet.

10. Es wird eine neue Java Klasse Controller erstellt. Über der Klassendefiniton muss die @RestController Annotation eingefügt werden. Anschließend wird eine neue Methode suqare Implementiert, welche mit der @GetMapping Annotation auf denn Get Endpunkt „/api/square gemappt wird.

11. Feature auf GitHub pushen

12. Nun wird auf GitHub ein Compare&Pull Request erstellt von „feature“ auf „master“. Anschließend wird gemerged und der Feature Branch kann gelöscht werden.

13. Compare&Pull Request durchführen, von „develop“ Branch in den „master“. Anschließend wird gemerged. Hierbei wird ebenfalls der Workflow ausgeführt um das Artifakt zu generieren

14. Sollte die Pipline funktionieren kann die Usersstory im Board auf Done gesetzt werden und das Artfakt kann unter Actions von der letzten Aktion heruntergeladen werden

15. In Intellj wird noch der lokale Feature Branch gelöscht und der aktuelle von Stand von Github gefetchet und upgedatet um die aktuelle Version zu haben.

16. Zuletzt wird noch das Readme erstellt und gepusht