Reusable Artifacts

Ziel und Hintergrund

Etwas wiederverwendbares («Reusable Artifact») zu erstellen, dass andere Teams direkt in ihren Repos nutzen können. So können Prozesse vereinheitlicht oder erleichtert werden.

Mögliche Artefakte für Pull Requests & Reviews

1. Pull Request Template (PR-Template)

Dieses Template erscheint automatisch, sobald ein Teammitglied einen neuen Pull Request erstellt. Es sorgt für eine schönere und einheitlichere Struktur und erleichtert ebenfalls Reviewers.

Dateipfad:
`.github/pull_request_template.md`

Inhalt des Templates:
- Titel & Beschreibung der Änderung
- Änderungen-Checkliste
  - Code lokal getestet
  - Reviewer hinzugefügt
  - …
- Bereich für Screenshots

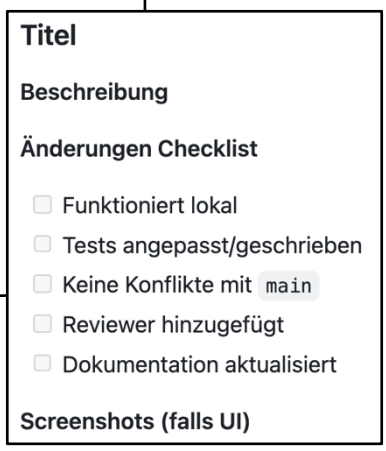
Beispiel:

```
### Titel

**Beschreibung**
<!-- Beschreibe die Änderungen und warum sie notwendig ist? -->

**Änderungen Checklist**
- [ ] Funktioniert lokal
- [ ] Tests angepasst/geschrieben
- [ ] Keine Konflikte mit `main`
- [ ] Reviewer hinzugefügt
- [ ] Dokumentation aktualisiert

**Screenshots (falls UI)**
```



2. Review-Checklist

Diese Checkliste unterstützt Reviewer:innen einen Pull Request effizient zu überprüfen, bevor dieser akzeptiert wird. Die Checkliste kann direkt in das README oder auch ins PR-Template eingebaut werden.

Dateipfad:
`Artifact/review_checklist.md`

Inhalt:
- Titel & Beschreibung
- Inhalts und Qualitätskriterien
    - Verständliche Beschreibung
    - Keine Sicherheitsprobleme
    - Sauberer Code
    - Tests erfolgreich
    - …

Beispiel:

```
### Review Checklist – Pull Requests

Diese Checkliste hilft Reviewer:innen, Pull Requests effizient zu prüfen:

**Inhalt**
- [ ] Der Titel beschreibt klar, was geändert wurde.
- [ ] Es gibt eine verständliche Beschreibung (Warum & Was?).
- [ ] Naming und Struktur sind konsistent.

**Qualität**
- [ ] Der Code/Markdown ist verständlich und dokumentiert.
- [ ] Keine unnötigen Dateien oder Kommentare.
- [ ] Keine offensichtlichen Sicherheitsprobleme.
- [ ] Alle Tests (falls vorhanden) laufen erfolgreich.
```

Review Checklist – Pull Requests

Diese Checkliste hilft Reviewer:innen, Pull Requests effizient zu prüfen:

Inhalt

- ☐ Der Titel beschreibt klar, was geändert wurde.
- ☐ Es gibt eine verständliche Beschreibung (Warum & Was?).
- ☐ Naming und Struktur sind konsistent.

Qualität

- ☐ Der Code/Markdown ist verständlich und dokumentiert.
- ☐ Keine unnötigen Dateien oder Kommentare.
- ☐ Keine offensichtlichen Sicherheitsprobleme.
- ☐ Alle Tests (falls vorhanden) laufen erfolgreich.

3. CODEOWNERS-Datei

In dieser Datei wird definiert, wer für welche Bereiche des Repositories verantwortlich ist und wer bei jedem neuen Pull Request automatisch als Reviewer hinzugefügt wird. Somit macht es den Review-Prozess klarer und schneller. Die CODEOWNERS-Datei kann im Root oder auch im .github Verzeichnis abgelegt werden.

Dateipfad:
`/CODEOWNERS` oder `.github/CODEOWNERS`

Beispiel:

```
1    # Main.java belongs to hslualexanderb
2    /src/calculator/Main.java @DenisRadi
3
4    # Calculator.java belongs to iaenzler
5    /src/calculator/Calculator.java @DenisRadi
6
7    # All other files belong to everyone
8    * @DenisRadi @Raniatissira @Noni2402 @Enis8
```

- Zeile 1&2 bzw. 4&5 definieren, wer für den jeweiligen Codeabschnitt verantwortlich ist und wer automatisch als Reviewer hinzugefügt wird, sobald ein Pull Request erstellt wird.

- Zeile 7&8 legen fest, wer für alle übrigen Dateien im Repository als Reviewer zuständig ist.
    - In diesem Fall ist der Scrum Master @DenisRadi als Hauptverantwortlicher eingetragen, mit der Möglichkeit, die Review-Aufgaben an die danebenstehenden Personen zu delegieren.

Ergebnis

Im Rahmen des Themas „Pull Requests & Reviews“ wurden mehrere wiederverwendbare Artefakte erstellt, um den Review-Prozess auf GitHub zu strukturieren und zu automatisieren.

1. **Pull Request Template** – zur Vereinheitlichung der Änderungsdokumentation und zur Qualitätssicherung.
2. **Review Checklist** – zur Unterstützung der Reviewer:innen bei der strukturierten Kontrolle von Änderungen.
3. **CODEOWNERS-Datei** – zur automatischen Zuweisung von Reviewern (insbesondere dem Scrum Master) und zur klaren Verantwortlichkeitsregelung.

Fazit

Mit dem erstellten Reusable Artifact wurde ein klar strukturierter und automatisierter Review-Prozess auf GitHub umgesetzt. Durch die Kombination aus Pull Request Template, Review Checklist und CODEOWNERS-Datei wird sichergestellt, dass Änderungen stets geprüft, dokumentiert und nachvollziehbar sind.

Das Artefakt ist leicht wiederverwendbar und verbessert die Teamkommunikation, Qualitätssicherung und Transparenz im Entwicklungsprozess.