LU10b - Review mit Pull Requests

Auf einen Blick
In den meisten Teams wird nicht in grossen Sitzungen reviewt, sondern **jede Änderung** einzeln, bevor sie in den Hauptzweig kommt. Auf GitHub geschieht das mit einem **Pull Request** (PR): Die Autorin schlägt die Änderung vor, ein bis zwei Reviewer kommentieren sie Zeile für Zeile und geben sie frei oder verlangen Änderungen.

Lernziele

Sie können

- den Ablauf eines Reviews mit Pull Request beschreiben,
- einen Pull Request auf GitHub reviewen und Kommentare schreiben,
- hilfreiche Review-Kommentare formulieren und Verbesserungen vorschlagen.

Ablauf

1. Die Autorin erstellt einen **Branch** und committet ihre Änderung, z.B. `fix/event-filter`.
2. Sie pusht den Branch und eröffnet auf GitHub einen **Pull Request** gegen `main`. In der Beschreibung steht, **was** geändert wurde und **warum** (z.B. `Fixes #12`).
3. Automatische Prüfungen laufen: Tests und Linter in der CI-Pipeline (siehe LU13).
4. Ein oder zwei **Reviewer** lesen die Änderungen im Reiter *Files changed* und kommentieren einzelne Zeilen.
5. Die Reviewer schliessen das Review ab mit
    - **Comment**: nur Anmerkungen,
    - **Approve**: Änderung ist in Ordnung,
    - **Request changes**: Änderung muss überarbeitet werden.
6. Die Autorin passt den Code an und pusht erneut. Der PR aktualisiert sich automatisch.
7. Ist der PR freigegeben und sind alle Prüfungen grün, wird er in `main` zusammengeführt (*Merge*).

Unterschied zum formalen Review

	Formales Review (Inspektion)	**Review mit Pull Request**
Umfang	ganzes Programm oder grosser Teil	eine Änderung, oft nur wenige Zeilen
Zeitpunkt	einmalig, geplant	bei jeder Änderung, laufend
Rollen	Moderator, Autor, Gutachter	Autor, 1–2 Reviewer
Protokoll	Review-Protokoll	Kommentare im PR
Verbesserungsvorschläge	erst nach der Sitzung	direkt im Kommentar erwünscht

Gute Review-Kommentare

Im Pull Request dürfen und sollen Reviewer **Verbesserungen vorschlagen**. Dabei gilt:

- **Konkret:** Ort, Problem und Vorschlag nennen.
- **Begründet:** Warum ist der Vorschlag besser? Verweis auf Standard oder Prinzip.
- **Gewichtet:** Kennzeichnen, was zwingend ist und was nur eine Anregung. Viele Teams schreiben dafür `nit:` (Kleinigkeit) vor optionale Kommentare.
- **Freundlich:** Über den Code sprechen, nicht über die Person. Fragen statt Befehle.

wenig hilfreich	**hilfreich**
«Das ist falsch.»	«`event.timestamp` ist ein String. `.date()` führt hier zu einem `AttributeError`. Vorschlag: `datetime.fromisoformat(event.timestamp).date()`.»
«Schlechter Name.»	«nit: `c` sagt wenig aus. Wie wäre `failed_subjects`?»
«Warum so kompliziert?»	«Die drei Schleifen sind identisch. Könnten wir sie in eine Funktion `subject_grade(grades)` auslagern (DRY)?»

Mit der Funktion *Suggest changes* (Symbol ± im Kommentarfeld) können Reviewer auf GitHub direkt einen Codevorschlag machen, den die Autorin mit einem Klick übernimmt.

Kontrollfragen

1. Welche drei Abschlüsse kann ein Reviewer auf GitHub wählen?
2. Warum werden Pull Requests klein gehalten?
3. Formulieren Sie einen hilfreichen Kommentar zum Befund «except Exception fängt jeden Fehler ab».

Weiterführende Links

- GitHub Docs: Überprüfen vorgeschlagener Änderungen in einem Pull Request
- Google Engineering Practices: Code Review

M450-LU10, M450-D1I, M450-D1A

Marcel Suter

From:
https://wiki.bzz.ch/ - **BZZ - Modulwiki**

Permanent link:
https://wiki.bzz.ch/modul/m450/learningunits/lu10/pullrequest

Last update: **2026/11/23 09:20**