Tutorial: GitHub Issues og GitHub Projects

Fra user story til Done på et Kanban-board

Mål: I skal oprette et GitHub repository og et GitHub Project, beskrive en user story som et Issue, nedbryde den i tasks som sub-issues, organisere arbejdet og til sidst lukke issue'et fra en commit.

Før du går i gang

Du skal have en GitHub-konto og være logget ind. Tutorialen arbejder direkte på main i browser.

1. Opret et GitHub repository

1. Klik på + øverst til højre på GitHub og vælg New repository.
2. Giv repository'et navnet demo-repo.
3. Slå Add README til. Det giver repository'et en første commit og en main-branch.
4. Klik Create repository.

Kontrollér: Du står nu i repository'et og kan se filen README.md.

2. Opret et GitHub Project

GitHub Projects er projektstyringsdelen. Her skal du lave et board, som viser arbejdets flow.

5. Gå til din profil (øverst til højre) og vælg Profile.
6. Vælg fanen Projects.
7. Klik New project.
8. Vælg Kanban template.
9. Klik Create project.
10. Giv projektet navnet Demo Project.
11. Check at board har Status-felterne:

```
Backlog → Ready → In Progress → In Review → Done
```

Hvis projektet er oprettet med andre status-felter, kan du omdøbe, tilføje eller fjerne statusværdier.

2a. Gør Project public, hvis andre skal kunne se det

Et nyt projekt kan være private. Hvis fx vejleder, eksaminator eller censor skal kunne se projektet, skal synligheden være public.

- Åbn projektet og klik på ··· øverst til højre.
- Vælg Settings.
- Find Visibility under Danger zone.
- Skift fra Private til Public.

Bemærk: Project og repository har hver sin visibility. Et public Project gør ikke Issues fra et private repository synlige for personer uden adgang til repository'et.

2b. Link Project til repository

Repository og Project er oprettet hver for sig. Nu linker du projekt til repository, så det bliver synligt og let tilgængeligt fra repository'ets Projects-fane.

12. Gå tilbage til repository'et demo-repo.
13. Klik Projects i repository'ets øverste menu.
14. Klik Link a project.
15. Find og vælg Demo Project.
16. Kontrollér, at Demo Project nu står på repository'ets Projects-side.

Vigtigt: At linke Project til repository tilføjer ikke automatisk repository'ets Issues til Project. Det enkelte Issue skal stadig tilføjes senere.

3. Opret labels i repository

Labels oprettes i repository og bruges til at kategorisere Issues.

Vigtigt: Et Issue hører til et repository. Et GitHub Project organiserer og viser Issues. Det samme Issue kan derfor ses både i repository'ets Issues og som et item på Project-boardet.

17. Gå tilbage til repository.
18. Vælg Issues og derefter Labels.
19. Klik New label.
20. Opret følgende labels. Vælg gerne tydeligt forskellige farver.

Label	Bruges til	Farveforslag
story	User stories	Blå
tech	Tech stories	Gul
bug	Fejl	Rød

Bemærk: I denne tutorial bruger vi story, tech og bug til at skelne mellem forskellige typer af Issues. Et Issue kan have flere labels, men i denne øvelse nøjes vi med disse tre. Man kan slette dem, som GitHub allerede har oprettet, hvis man ikke vil bruge dem.

Eksempler på de tre kategorier:

Label	Bruges til	Eksempel
story	User stories	Som medlem vil jeg kunne tilmelde mig et hold, så jeg kan deltage i træningen
tech	Tech stories	Applikationen skal anvende en connection pool til databaseforbindelser
bug	Fejl	Holdtilmelding fejler, hvis holdet kun har én ledig plads

4. Opret en user story som Issue

21. Vælg Issues i repository.
22. Klik New issue.

23. Brug titlen: Registrer kørsel for praktikbesøg.

Skriv følgende user story i beskrivelsen:

```
Som underviser vil jeg kunne registrere kørsel for praktikbesøg,
så jeg kan få kørselsgodtgørelse.
```

4a. Tilføj acceptkriterier som checkliste

Tilføj denne Markdown under user story:

```
## Acceptkriterier

- [ ] Dato for kørslen kan angives
- [ ] Antal kørte kilometer kan angives
- [ ] Kilometer skal være større end 0
- [ ] Registreringen kan gemmes
- [ ] Den gemte registrering kan efterfølgende ses
```

24. Tryk Create: Når issue vises, bliver linjerne til klikbare checkbokse.

Husk: Acceptkriterier beskriver krav til resultatet. Tasks beskriver arbejdet, der udføres for at realisere story'en.

4b. Se story i GitHub Projects

Project er linket til repository'et, men det betyder ikke, at alle Issues automatisk er med på boardet. Nu tilføjer du dette konkrete Issue.

25. Åbn det Issue, du netop har oprettet.
26. Find Projects i højre side af Issue.
27. Vælg Demo Project.
28. Åbn Demo Project.
29. Kontrollér, at user story'en vises på boardet.
30. Sæt status til Backlog, hvis den ikke allerede har denne status.

Kontrollér: User story'en findes under repository'ets Issues og på Project-boardet. Det er det samme Issue – ikke en kopi.

Man kan også oprette issues direkte fra boardet. Og man kan automatisere workflow, så et nyt issue i repository automatisk kommer på boardet. Vælg Workflows under Project og aktiver ”Auto-add to project”.

4c. Sæt labels på story'en

31. Åbn user story-issue'et.
32. Klik Labels i højre side.
33. Vælg story.

4d. Nedbryd user story i tasks som checkliste

En user story består som regel af flere arbejdsopgaver, som vi kan tilføje til issue, så vi husker dem.

34. Åbn user story-issuet.
    Tilføj denne Markdown under acceptkriterierne:

    ## Tasks
    - [ ] Lav datamodel for kørselsregistrering
    - [ ] Opret tabel til kørselsregistrering
    - [ ] Opret relevante Java-klasser
    - [ ] Implementér JDBC-metode til at gemme registreringen
    - [ ] Test Test registrering af kørsel

Tasks kan altså være modellering, databasearbejde, programmering eller test.

4e. Sæt tidsestimat på story

Kanban-templaten har allerede oprettet feltet **Estimate**, som kan bruges til jeres estimat.

35. Åbn user story-issue'et.
36. Find **Estimate** under **Projects** i højre side.
37. Klik Enter number....
38. Sæt Estimate på user story til fx 5.

Bemærk: GitHub-feltet er et talfelt. Hvad tallet betyder, skal teamet aftale. Vi arbejder med story points og ikke timer.

5. Assign et teammedlem

Assignee viser, hvem der har ansvar for at drive et Issue fremad.

39. Åbn et Issue.
40. Find Assignees i højre side.
41. Vælg det teammedlem, der skal have ansvar for issue'et.
42. Når teammedlemmet **begynder arbejdet**, åbnes Project-boardet, og issue'et trækkes fra **Ready** til **In Progress**.

At én person har ansvar for story, betyder ikke nødvendigvis at den person skal lave det hele. Det kan betyde: Der er én, som har ansvaret for at drive denne story frem til Done. En anden kan sagtens lave datamodellen og en tredje kan lave JDBC-delen. De koordinerer det i teamet. GitHub behøver ikke nødvendigvis registrere **hvem der udførte hver del-aktivitet**.

6. Luk user story med en commit

Nu skal du se sammenhængen mellem kodehistorikken og projektstyringen. Vi laver en lille ændring direkte på main.

Før commit: Sørg for, at user story'en er tilføjet til Project. Flyt den eventuelt til In Review, og notér dens issue-nummer, fx #1.

43. Lav en lille ændring i README.md eller en anden fil i repository.
44. Commit ændringen direkte til main med en closing keyword-besked. Hvis story er Issue #1, kan du skrive:

```
Update project documentation. Closes #1
```

Du kan også bruge fx Fixes #1 eller Resolves #1. Det afgørende er issue-nummeret og et closing keyword.

45. Hvis du arbejder lokalt, push committen til GitHub. Hvis du redigerer direkte på GitHub, commit ændringen til main.

Vigtigt: Et closing keyword i en commit lukker et issue, når committen er på repository'ets default branch – her main.

7. Se issue er flyttet til Done

46. Gå til repository'ets Issues.
47. Kontrollér, at user story-issue er Closed.
48. Åbn Demo Project.
49. Kontrollér, at story står i Done.

GitHub Projects har et indbygget workflow, der som standard kan sætte et item til Done, når det tilknyttede Issue lukkes.

Hvis issue er Closed, men ikke står i Done:

- Åbn Project.
- Åbn projektmenuen og vælg Workflows.
- Find workflowet, der sætter Status til Done, når et Issue lukkes.
- Kontrollér, at workflowet er slået til.

Resultatet: Commit med Closes #1 → Issue #1 bliver Closed → Project-workflow sætter Status til Done.

8. Recap: Hvad har du lige gjort?

- Oprettet et repository med en main-branch.
- Oprettet et GitHub Project med et Kanban-board.
- Oprettet og anvendt labels.
- Oprettet en user story som et Issue.
- Formuleret acceptkriterier som en Markdown-checkliste.
- Forbundet repository-Issue og GitHub Project.
- Nedbrudt user story i tasks som en Markdown-checkliste (man kan også bruge sub-issues, hvis tasks er store nok til, at teamet har behov for at styre dem selvstændigt med deres egen assignee, men i denne tutorial har vi valgt den simple løsning)

- Tilføjet et estimat i Project.
- Tildelt ansvar med Assignee.
- Koblet en commit til et Issue med Closes #nummer.
- Set et lukket Issue blive sat til Done på Project-boardet.

9. Den samlede model

```
Repository
  └── Issue: User story
        ├── Acceptkriterier (checkliste)
        ├── Labels
        ├── Assignee
        └── Sub-issues = tasks
              ├── modellering
              ├── database
              ├── programmering
              └── test

GitHub Project
  └── viser Issue + Project-felter
        ├── Status: Backlog → Ready → In Progress → In Review → Done
        └── Estimate

Git commit på main
  └── "Closes #1"
        → Issue lukkes
        → Project-status bliver Done
```