Metodika git

Autor: Samuel Kačeriak

Github je online nástroj na verzionovanie projektov. Okrem samotného verzionovania ponúka množstvo iných funkcionalít ako napríklad bug tracking, task management a ďalšie. Na tomto projekte sme sa rozhodli používať tento nástroj hlavne kvôli týmto funkciám a dobrých predchádzajúcim skúsenostiam.

Projekt máme rozdelený do troch repozitárov: web, aplikačný frontend a aplikačný backend. V každom repozitári sa nachádza súbor .gitignor slúžiaci na ignorovanie súborov, ktoré nepotrebujeme uploadovať online(napríklad node_modules). Repozitáre sú verejne dostupné na odkaze: https://github.com/vanetnuggets. Pre efektívnejšiu spoluprácu sme si spoločne zadefinovali niekoľko pravidiel používania Githubu.

Pravidlá používania

1. Do main branch sa priamo nepushuje, pridávanie nového kódu je zabezpečené pull requestami s child branchov
2. Každá nová funkcionalita sa developuje vo vlastnej banchi
3. Pred mergom do main branche je potrebné dôkladné odtestovanie implementovanej funkcionality
4. Po vytvorení merge requestu je potrebné vyplniť merge template
5. Merge request by mal byť skontrolovaný aspoň jedným iným členom tímu
6. Merge vykonávame až po rebasenutí main branche aby nedochádzalo ku konfliktom
7. Merguje autor requestu
8. Ak merge opravuje nejakú vedenú chybu v issues na githube, treba ju k nemu pridať a označiť jej stav
9. Bonusové pravidlo: commit musí začínať emotikonom

Github actions

Použitím github actions sa automaticky po mergnutí do main branche spustí akcia, ktorá aktualizuje príslušný repozitár na vzdialenom stroji. Týmto ušetríme množstvo času, ktorý by sme museli venovať manuálnej aktualizácií po každom jednom commite. O to si ale treba dať väčší pozor aby boli commity skontrolované a bez zbytočných bugov.

Príklady

Uvedieme aj niekoľko príkladov použitia gitu aj s konkrétnymi príkazmi:

- **Práca na feature branchi a update z main branch**

Tento postup využívame pri tomto projekte bežne, pretože každý člen pracuje na svojej branchi a až keď je feature hotová tak sa merguje do main branch.

git add . -pridáme súbory

git status -skontrolujeme či sme pridali správne súbory

git commit -m “<opis čo sa zmenilo>” -commit s výstižným popisom

git fetch -stiahneme si updaty z remote branch

git stash -ak sme nepridávali všetky súbory, musíme urobiť stash

git rebase origin/main -aplikujeme naš commit na main branch aby nedošlo ku konfliktom

git push -f origin <feature_branch> -ak sme dávali rebase tak musíme použiť prepínač f, pretože sme updatli main

git stash apply -ak sme si predtým stashli súbory teraz si ich vieme znova aplikovať

- **Vytvorenie feature_branch**

Takýmto spôsobom vieme vytvoriť separátny branch na ktorom, vieme implementovať novú funkcionalitu.

git checkout -b <feature_branch>

- **Prepínanie medzi branchami**

Nasledujúcim príkazom sa vieme prepínať medzi jednotlivými branchami. Ak máme necommitnuté súbory najskôr je potrebné vykonať príkaz stash.

git checkout <branch_name>

- **Merge request**

Merge request používame po implementovaní a otestovaní novej funkcionality na našej child branchi. Pre merge do main branch nastavíme aspoň jedného reviewera a počkáme na schválenie. Po schválení a za predpokladu nevzniknutia konfliktov môžeme branch mergnúť. Takto vieme minimalizovať riziko bugov v main branch.

- **Merge conflict**

Po vzniknutí konfliktu je potrebné upraviť súbory v ktorých konflikt nastal. Po úprave je potrebné znova súbory pridať a použiť príkaz:

git merge --continue a git push.