

Git i GitHub — wersjonowanie i GitHub Pages

Czas: 2 × 45 min

praca indywidualna

Cel: Prowadzenie historii zmian projektu w Git (commit, gałąź, scalenie z rozwiązaniem konfliktu), wysłanie repozytorium na GitHub i opublikowanie strony przez GitHub Pages.

Po zajęciach potrafisz:

- konfiguruje Git i zakłada repozytorium z plikiem `.gitignore` wykluczającym pliki prywatne i robocze
- zatwierdza zmiany z opisowymi komunikatami i czyta historię (`log`, `diff`, `status`)
- tworzy gałąź, scala ją z główną i samodzielnie rozwiązuje konflikt
- łączy repozytorium lokalne z GitHubem (`remote`, `push`, `pull`)
- publikuje stronę statyczną przez GitHub Pages i sprawdza działanie ścieżek względnych

Sprzęt i materiały

- komputer z Windows 11 i dostępem do internetu
- projekt strony wizytówki z zadań 18–19 albo folder `r1_28_projekt` z Pliki_do_zadan
- konto na GitHub (założone w domu lub na lekcji; wymagany wiek co najmniej 13 lat)

Oprogramowanie

- **Git for Windows** — Git Bash lub terminal w VS Code; zawiera Git Credential Manager (logowanie przez przeglądarkę) git-scm.com/downloads
- **GitHub** — repozytorium publiczne i GitHub Pages (bezpłatnie dla repozytoriów publicznych) github.com
- **Visual Studio Code** — panel Source Control pokazuje te same zmiany co `git status` code.visualstudio.com

Przygotowanie przed zajęciami

- Załóż konto na GitHub i włącz uwierzytelnianie dwuskładnikowe (GitHub wymaga go od większości kont).
- Sprawdź w GitHub → Settings → Emails swój adres prywatny typu `...@users.noreply.github.com` — użyjesz go w konfiguracji Git.

Bezpieczeństwo

- Repozytorium publiczne widzi każdy. Nie wysyłaj haseł, notatek prywatnych ani danych osobowych — dlatego plik `notatki_prywatne.txt` musi trafić do `.gitignore` przed pierwszym commitem.
- Na komputerze szkolnym po zajęciach wyloguj się z GitHub w przeglądarce i usuń zapisane poświadczenia (Menedżer poświadczeń Windows → Poświadczenia systemu Windows → wpisz `git:https://github.com`).
- Nie używaj prawdziwego adresu e-mail w `user.email`, jeśli nie chcesz go publikować w historii commitów — użyj adresu noreply.

Przebieg zadania

1 Repozytorium lokalne

Pracuj w terminalu (Git Bash lub Terminal w VS Code) w folderze projektu `zad28`. Po każdym poleceniu czytaj komunikat Git.

```
git config --global user.name "Imię Nazwisko"
git config --global user.email "login@users.noreply.github.com"
git config --global init.defaultBranch main
git init
git status
```

1. G1. Skopiuj swój projekt z zadań 18–19 (lub `r1_28_projekt`) do folderu `nazwisko_imie/zad28`. Skonfiguruj Git jak wyżej i wyświetl ustawienia `git config --list --global`.
2. G2. Załóż repozytorium (`git init`) i wyświetl `git status`. Zapisz, które pliki są „Untracked”.
3. G3. Utwórz plik `.gitignore` wykluczający: `notatki_prywatne.txt`, wszystkie pliki `*.bak`, folder `.vscode/` oraz pliki systemowe `Thumbs.db` i `desktop.ini`. Sprawdź `git status --short` — wykluczonych plików ma już nie być na liście.
4. G4. Dodaj pliki (`git add .`) i zatwierdź: `git commit -m "Pierwsza wersja strony wizytówki"`. Wyświetl `git log --oneline`.
5. G5. Zmień jeden tekst na stronie głównej, obejrzyj zmianę poleceniem `git diff` (co oznaczają linie z `-` i `+`?) i zatwierdź ją z opisowym komunikatem.

2 Gałąź, scalenie i konflikt

Konflikt powstaje, gdy w dwóch gałęziach zmieniono tę samą linię. Git zostawia w pliku oba warianty między znacznikami `<<<<<<`, `=====` i `>>>>>>` — Ty decydujesz o wersji końcowej.

1. G6. Utwórz gałąź i przejdź na nią: `git switch -c kontakt`. Dodaj podstronę `kontakt.html` (adres fikcyjny, e-mail w domenie `example.com`), link „Kontakt” w menu strony głównej oraz link w stopce. Zatwierdź. Przełącz się na `main` i sprawdź, czy plik `kontakt.html` jest widoczny w folderze.
2. G7. Na gałęzi `main` zmień tę samą linię stopki strony głównej (np. dopisz miasto) i zatwierdź.
3. G8. Scal gałąź: `git merge kontakt`. Odczytaj komunikat o konflikcie, otwórz plik w VS Code, połącz obie zmiany w jedną stopkę, usuń znaczniki konfliktu, a potem `git add` i `git commit`. Usuń scaloną gałąź (`git branch -d kontakt`).
4. G9. Wyświetl historię jako graf: `git log --oneline --graph --all` i zrób zrzut. Poleceniem `git ls-files` sprawdź, które pliki są śledzone.

3 GitHub i GitHub Pages

Na GitHub utwórz **puste** repozytorium publiczne (New repository; bez README, .gitignore i licencji), np. `wizytowka`.

```
git remote add origin https://github.com/LOGIN/wizytowka.git
git remote -v
git push -u origin main
```

1. G10. Połącz repozytorium lokalne z GitHub i wyślij gałąź `main`. Przy pierwszym `push` zaloguj się w oknie przeglądarki. Odśwież stronę repozytorium i sprawdź, czy nie ma tam plików z `.gitignore`.
2. G11. Włącz publikację: Settings → Pages → Build and deployment → Source: Deploy from a branch → Branch: `main`, folder `/ (root)` → Save. Po 1–2 minutach otwórz adres `https://LOGIN.github.io/wizytowka/`. Sprawdź, czy wczytują się style, grafiki i wszystkie podstrony; jeśli nie — popraw ścieżki (wielkość liter, ukośniki `/`, ścieżki względne), zatwierdź i wyślij ponownie.
3. G12. Na GitHub dodaj plik `README.md` (Add file → Create new file) z nazwą projektu, autorem i adresem strony. Pobierz zmianę poleceniem `git pull` i sprawdź `git log --oneline`.
4. Zrób zrzuty: repozytorium na GitHub z listą commitów, ustawienia Pages z adresem strony, opublikowana strona. Na koniec usuń poświadczenia z komputera szkolnego (patrz Bezpieczeństwo).

W sprawozdaniu umieść

- ☐ tabela G1–G12 z poleceniami i najważniejszymi komunikatami
- ☐ zawartość pliku `.gitignore`
- ☐ fragment pliku z konfliktem (przed rozwiązaniem) i wersja końcowa
- ☐ zrzuty: `git log --oneline --graph --all`, repozytorium na GitHub, ustawienia Pages, opublikowana strona
- ☐ adres repozytorium i adres strony na GitHub Pages
- ☐ odpowiedzi na pytania kontrolne

Pytania kontrolne

1. Czym różni się obszar roboczy, poczekalnia (staging) i repozytorium? Które polecenia przenoszą zmiany między nimi?
2. Dlaczego `.gitignore` trzeba utworzyć przed pierwszym commitem? Co się stanie, gdy plik z hasłem trafi już do historii?
3. Kiedy powstaje konflikt przy `git merge` i jak go rozwiązać?
4. Czym różni się `git push` od `git pull`? Czym jest `origin`?

Kryteria oceny

Kryterium	Punkty
G1–G4 — konfiguracja, repozytorium, poprawny .gitignore (4 reguły), pierwszy commit bez plików prywatnych	4
G5 — git diff i commit z opisowym komunikatem	1
G6–G9 — gałąź, konflikt rozwiązany bez pozostawionych znaczników, graf historii z rozgałęzieniem	5
G10–G12 — repozytorium na GitHub, działająca strona na GitHub Pages (style, grafiki, podstrony), README pobrane przez pull	5
Sprawozdanie: zrzuty, adresy, fragment konfliktu	2
Odpowiedzi na pytania kontrolne	3
Razem	20

Sprawozdanie przygotuj na wzorze dołączonym do zadania (plik .docx o tym samym numerze).