Scenariusz zajęć: Praca z gałęziami w Git

Cel zajęć:

- Zrozumienie podstawowych i zaawansowanych operacji na gałęziach w Git.
- Praktyczne zastosowanie gałęzi w różnych scenariuszach programistycznych.
- Omówienie zalet i potencjalnych wad pracy z gałęziami.

Plan zajęć

1. Wprowadzenie do gałęzi w Git (10 minut)

- **Definicja gałęzi:** Gałąź w Git to niezależna linia rozwoju, która pozwala na pracę nad nowymi funkcjonalnościami, poprawkami błędów lub eksperymentami bez wpływu na główną linię kodu.
- **Podstawowe komendy:**
    - **Tworzenie gałęzi: git branch <nazwa-gałęzi>**
        - **Opis:** Tworzy nową gałąź o podanej nazwie. Gałąź ta jest kopią bieżącej gałęzi, ale nie przełącza się na nią automatycznie.
    - **Przełączanie się na gałąź: git checkout <nazwa-gałęzi>**
        - **Opis:** Przełącza się na istniejącą gałąź. Zmienia bieżący kontekst pracy na wskazaną gałąź.
    - **Tworzenie i przełączanie się na nową gałąź: git checkout -b <nazwa-gałęzi>**
        - **Opis:** Tworzy nową gałąź i automatycznie przełącza się na nią. Jest to skrót do dwóch komend: git branch i git checkout.
    - **Łączenie gałęzi: git merge <nazwa-gałęzi>**
        - **Opis:** Łączy zmiany z podanej gałęzi do bieżącej gałęzi. Jeśli wystąpią konflikty, Git poprosi o ich rozwiązanie.
    - **Usuwanie gałęzi: git branch -d <nazwa-gałęzi>**
        - **Opis:** Usuwa wskazaną gałąź. Gałąź musi być zintegrowana z inną gałęzią (np. główną), aby mogła zostać usunięta.
- **Struktura gałęzi:**
    - **Gałąź główna (main/master):** Główna linia rozwoju, która zawiera stabilny kod gotowy do produkcji.
    - **Gałęzie funkcjonalne (feature branches):** Gałęzie tworzone do pracy nad nowymi funkcjonalnościami.
    - **Gałęzie naprawcze (bugfix branches):** Gałęzie tworzone do naprawy błędów.
    - **Gałęzie eksperymentalne (experimental branches):** Gałęzie tworzone do testowania nowych rozwiązań.

2. Praktyczne zastosowanie gałęzi (20 minut)

- **Przykład 1: Praca nad nową funkcjonalnością**

- **Tworzenie gałęzi** feature-login**:**
- git checkout -b feature-login
  - **Opis:** Tworzy nową gałąź o nazwie feature-login i automatycznie przełącza się na nią. Gałąź ta będzie używana do pracy nad funkcjonalnością logowania.
- **Wprowadzanie zmian i commitowanie:**
- echo "Kod logowania" > login.js
- git add login.js
- git commit -m "Dodano funkcjonalność logowania"
  - **Opis:** Dodaje nowy plik login.js z kodem logowania, dodaje go do śledzenia przez Git i zatwierdza zmiany z odpowiednim komunikatem.
- **Łączenie z główną gałęzią:**
- git checkout main
- git merge feature-login
  - **Opis:** Przełącza się na główną gałąź main i łączy zmiany z gałęzi feature-login do głównej gałęzi.
- **Usuwanie gałęzi:**
- git branch -d feature-login
  - **Opis:** Usuwa gałąź feature-login, ponieważ zmiany zostały już zintegrowane z główną gałęzią.

- **Przykład 2: Poprawka błędu**
  - **Tworzenie gałęzi** bugfix-auth**:**
  - git checkout -b bugfix-auth
    - **Opis:** Tworzy nową gałąź o nazwie bugfix-auth i automatycznie przełącza się na nią. Gałąź ta będzie używana do naprawy błędu w autoryzacji.
  - **Wprowadzanie poprawek i commitowanie:**
  - echo "Poprawka błędu autoryzacji" >> auth.js
  - git add auth.js
  - git commit -m "Poprawiono błąd autoryzacji"
    - **Opis:** Dodaje poprawkę do pliku auth.js, dodaje go do śledzenia przez Git i zatwierdza zmiany z odpowiednim komunikatem.

- **Łączenie z główną gałęzią:**
- git checkout main
- git merge bugfix-auth
  - **Opis:** Przełącza się na główną gałąź main i łączy zmiany z gałęzi bugfix-auth do głównej gałęzi.
- **Usuwanie gałęzi:**
- git branch -d bugfix-auth
  - **Opis:** Usuwa gałąź bugfix-auth, ponieważ zmiany zostały już zintegrowane z główną gałęzią.

- **Przykład 3: Praca nad eksperymentalną funkcją**
  - **Tworzenie gałęzi** experiment-new-ui**:**
  - git checkout -b experiment-new-ui
    - **Opis:** Tworzy nową gałąź o nazwie experiment-new-ui i automatycznie przełącza się na nią. Gałąź ta będzie używana do testowania nowego interfejsu użytkownika.
  - **Wprowadzanie zmian i commitowanie:**
  - echo "Nowy interfejs użytkownika" > ui.js
  - git add ui.js
  - git commit -m "Eksperymentalny interfejs użytkownika"
    - **Opis:** Dodaje nowy plik ui.js z kodem nowego interfejsu użytkownika, dodaje go do śledzenia przez Git i zatwierdza zmiany z odpowiednim komunikatem.
  - **Łączenie z główną gałęzią:**
  - git checkout main
  - git merge experiment-new-ui
    - **Opis:** Przełącza się na główną gałąź main i łączy zmiany z gałęzi experiment-new-ui do głównej gałęzi.
  - **Usuwanie gałęzi:**
  - git branch -d experiment-new-ui
    - **Opis:** Usuwa gałąź experiment-new-ui, ponieważ zmiany zostały już zintegrowane z główną gałęzią.

3. Zaawansowane operacje na gałęziach (20 minut)

- **Rebase:** Przepisanie historii commitów, aby były bardziej uporządkowane.
    - **Komenda:** git rebase <nazwa-gałęzi>
    - **Przykład:**
    - git checkout feature-login
    - git rebase main
    - **Opis:** Rebase pozwala na uporządkowanie historii commitów, co ułatwia śledzenie zmian i integrację z główną gałęzią. W tym przykładzie gałąź feature-login jest aktualizowana o najnowsze zmiany z gałęzi main.
    - **Scenariusz:** Pracujesz nad nową funkcjonalnością w gałęzi feature-login, ale w międzyczasie główna gałąź została zaktualizowana. Używasz git rebase main, aby zaktualizować swoją gałąź o najnowsze zmiany z głównej gałęzi.
- **Rozwiązywanie konfliktów:** Co zrobić, gdy podczas łączenia gałęzi wystąpią konflikty.
    - **Przykład:**
    - git merge feature-login
    - **Opis:** Konflikty mogą wystąpić, gdy różne zmiany są wprowadzane w tym samym pliku w różnych gałęziach. Rozwiązywanie konfliktów polega na ręcznym wyborze, które zmiany mają zostać zachowane.
    - **Scenariusz:** Podczas łączenia gałęzi feature-login z główną gałęzią wystąpił konflikt w pliku app.js. Otwierasz plik