Praca z Git w projekcie CerbDesk

Twoje Imie

December 8, 2024

1 Wprowadzenie

Przy pracy nad projektem **CerbDesk** w zespole, kluczowe jest utrzymywanie repozytorium w synchronizacji z GitHub oraz odpowiednia organizacja zmian. Poniżej opisane zostały kroki, jak poprawnie aktualizować repozytorium.

2 Regularna aktualizacja projektu

Zanim rozpoczniesz prace, zawsze pobierz najnowsze zmiany z głównej gałezi (`main`):

```
git pull origin main
```

Listing 1: Pobieranie najnowszych zmian

3 Dodawanie i wypychanie zmian

Po wprowadzeniu zmian do projektu wykonaj nastepujace kroki:

3.1 Dodanie plików do systemu kontroli wersji

Dodaj wszystkie zmienione pliki:

```
git add .
```

Listing 2: Dodawanie plików do Gita

3.2 Zatwierdzenie zmian

Zatwierdź zmiany, dodajac opisowy komunikat:

```
git commit -m "Opis␣wprowadzonych␣zmian"
```

Listing 3: Commitowanie zmian

3.3 Wypchniecie zmian na GitHub

Wypchnij zmiany do zdalnego repozytorium:

```
git push origin main
```

Listing 4: Wysyłanie zmian na GitHub

4 Praca zespołowa i rozwiazywanie konfliktów

Jeżeli ktoś inny wprowadził zmiany w miedzyczasie, moga wystapić konflikty przy próbie wypchniecia własnych zmian. Aby to rozwiazać:

1. Najpierw pobierz najnowsze zmiany:

```
git pull origin main
```

Listing 5: Pobieranie najnowszych zmian

2. Jeśli wystapia konflikty, otwórz odpowiednie pliki, znajdź sekcje oznaczone przez Git (np. `HEAD`) i rozwiaż konflikty recznie.

3. Po rozwiazaniu konfliktów dodaj poprawione pliki:

```
git add <nazwa_pliku>
```

Listing 6: Dodawanie poprawionych plików

4. Zatwierdź zmiany:

```
git commit -m "Rozwi zanie␣konflikt w"
```

Listing 7: Commitowanie po rozwiazaniu konfliktów

5. Wypchnij zmiany ponownie:

```
git push origin main
```

Listing 8: Wypychanie po rozwiazaniu konfliktów

5 Tworzenie nowych gałezi

Przy dodawaniu wiekszych funkcjonalności zaleca sie prace na osobnych gałeziach:

1. Utwórz nowa gałaź:

```
git checkout -b nazwa-funkcji
```

Listing 9: Tworzenie nowej gałezi

2. Po zakończeniu pracy wypchnij gałaź na GitHub:

```
git push origin nazwa-funkcji
```

Listing 10: Wysyłanie gałezi na GitHub

3. Na GitHub stwórz **Pull Request** (PR), aby zintegrować zmiany z główna gałezia (`main`).

6 Śledzenie stanu repozytorium

6.1 Sprawdzanie statusu

Aby zobaczyć, które pliki zostały zmodyfikowane:

```
git status
```

Listing 11: Status plików

6.2 Historia commitów

Aby przejrzeć historie commitów:

```
git log --oneline
```

Listing 12: Przegladanie historii commitów

7 Podsumowanie

W trakcie pracy nad projektem pamietaj, aby regularnie synchronizować repozytorium (`git pull`) oraz dodawać i zatwierdzać zmiany z opisowymi komunikatami (`git commit`). W przypadku pracy zespołowej korzystaj z gałezi i Pull Requestów.