

## **Przedmiot: Narzędzia do automatyzacji budowy oprogramowania**

### **Lab 1: Podstawy Git/GitHub**

#### **1. Wprowadzenie do systemów kontroli wersji**

##### **Czym jest kontrola wersji?**

System kontroli wersji (VCS - Version Control System) to narzędzie, które śledzi zmiany w plikach w czasie, umożliwiając powrót do wcześniejszych wersji, porównywanie zmian i współpracę wielu osób nad tym samym projektem.

##### **Rodzaje systemów kontroli wersji:**

###### **1. Lokalne systemy kontroli wersji**

- Przechowują zmiany w lokalnej bazie danych
- Przykład: RCS (Revision Control System)
- Ograniczenia: brak współpracy, ryzyko utraty danych

###### **2. Scentralizowane systemy kontroli wersji (CVCS)**

- Przechowują historię zmian na centralnym serwerze
- Przykłady: SVN (Subversion), CVS
- Zalety: kontrola dostępu, łatwiejsze zarządzanie
- Wady: pojedynczy punkt awarii, problemy z łącznością

###### **3. Rozproszone systemy kontroli wersji (DVCS)**

- Każdy użytkownik posiada pełną kopię repozytorium
- Przykłady: Git, Mercurial
- Zalety: praca offline, brak pojedynczego punktu awarii, szybkość

##### **Dlaczego Git?**

- Stworzony przez Linusa Torvaldsa (twórcę Linuxa) w 2005 roku
- Niezwykle szybki i wydajny
- Rozproszony model pracy
- Wsparcie dla nieliniowego rozwoju (gałęzie)
- Integralność danych (wykorzystanie funkcji skrótu SHA-1)
- Powszechne użycie w branży IT
- Darmowy i open-source

##### **GitHub i inne platformy**

- **GitHub** - platforma hostingowa dla repozytoriów Git, obecnie należąca do Microsoft
- **GitLab** - alternatywa dla GitHub z dodatkowymi funkcjami CI/CD
- **Bitbucket** - platforma od Atlassian, zintegrowana z innymi produktami firmy

## 2. Podstawowe komendy Git

### Instalacja i konfiguracja Git

#### Instalacja:

##### Windows:

Pobierz instalator z [git-scm.com](https://git-scm.com) i zainstaluj

##### Linux (Ubuntu/Debian):

```
sudo apt update
sudo apt install git
```

##### macOS:

```
brew install git
```

lub zainstaluj Xcode Command Line Tools:

```
xcode-select --install
```

#### Podstawowa konfiguracja:

*# Ustawienie nazwy użytkownika*

```
git config --global user.name "Twoje Imię i Nazwisko"
```

*# Ustawienie adresu email*

```
git config --global user.email "twoj.email@example.com"
```

*# Sprawdzenie konfiguracji*

```
git config --list
```

#### Tworzenie repozytorium

*# Inicjalizacja nowego repozytorium*

```
git init
```

*# Klonowanie istniejącego repozytorium*

```
git clone https://github.com/uzytkownik/repozytorium.git
```

#### Podstawowy workflow

*# Sprawdzenie statusu repozytorium*

```
git status
```

*# Dodanie plików do poczekalni (staging area)*

```
git add nazwa_pliku          # Dodanie pojedynczego pliku
```

```
git add .                    # Dodanie wszystkich zmienionych plików
```

```
git add *.txt                 # Dodanie wszystkich plików .txt
```

*# Zatwierdzenie zmian (commit)*

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

*# Commit z pominięciem git add (tylko dla śledzonych plików)*  
git commit -am "Opis zmian"

#### Przeglądanie historii

*# Podstawowe wyświetlenie historii*  
git log

*# Historia w jednej linii*  
git log --oneline

*# Historia z graficzną reprezentacją*  
git log --graph --oneline --all

*# Historia ze zmianami dla konkretnego pliku*  
git log -p nazwa\_pliku

#### Synchronizacja z repozytorium zdalnym

*# Dodanie zdalnego repozytorium*  
git remote add origin https://github.com/uzytkownik/repozytorium.git

*# Wyświetlenie zdalnych repozytoriów*  
git remote -v

*# Pobieranie zmian z repozytorium zdalnego*  
git fetch origin

*# Pobieranie i łączenie zmian (fetch + merge)*  
git pull origin main

*# Wysyłanie zmian do repozytorium zdalnego*  
git push origin main

#### Cofanie zmian

*# Cofnięcie zmian w pliku roboczym (uwaga: bezpowrotne!)*  
git checkout -- nazwa\_pliku

*# Cofnięcie plików z poczekalni*  
git reset HEAD nazwa\_pliku

*# Modyfikacja ostatniego commita*  
git commit --amend -m "Nowy opis commita"

*# Cofnięcie ostatniego commita (zachowuje zmiany w katalogu roboczym)*  
git reset HEAD~1

*# Cofnięcie ostatniego commita (usunięcie zmian)*  
git reset --hard HEAD~1

### 3. Tworzenie i zarządzanie repozytorium na GitHub

#### Tworzenie konta na GitHub

1. Przejdź do [github.com](https://github.com)
2. Kliknij "Sign up" i postępuj zgodnie z instrukcjami
3. Wybierz darmowy plan dla konta indywidualnego
4. Zweryfikuj adres email

#### Tworzenie nowego repozytorium na GitHub

1. Kliknij przycisk "+" w prawym górnym rogu, a następnie "New repository"
2. Podaj nazwę repozytorium
3. Dodaj opcjonalny opis
4. Wybierz widoczność (publiczne/prywatne)
5. Opcjonalnie zaznacz "Initialize this repository with a README"
6. Kliknij "Create repository"

#### Połączenie lokalnego repozytorium z GitHub

```
# Jeśli utworzyłeś nowe repozytorium lokalnie
git remote add origin https://github.com/uzytkownik/repozytorium.git
git branch -M main # Zmiana nazwy brancha na 'main' (jeśli potrzebne)
git push -u origin main
```

```
# Jeśli klonujesz istniejące repozytorium
git clone https://github.com/uzytkownik/repozytorium.git
cd repozytorium
```

#### Uwierzytelnianie na GitHub

##### Poprzez HTTPS (z zapisem poświadczeń):

```
# Zapisanie poświadczeń (credentials)
git config --global credential.helper store
# lub w pamięci podręcznej na określony czas (np. 1 godzina)
git config --global credential.helper 'cache --timeout=3600'
```

##### Poprzez SSH:

1. Generowanie klucza SSH:

```
ssh-keygen -t ed25519 -C "twoj.email@example.com"
```

2. Dodanie klucza SSH do agenta SSH:

```
eval "$(ssh-agent -s)"
ssh-add ~/.ssh/id_ed25519
```

3. Dodanie klucza publicznego do GitHub:

- Skopiuj zawartość pliku ~/.ssh/id\_ed25519.pub
- Przejdź do GitHub -> Settings -> SSH and GPG keys -> New SSH key
- Wklej klucz i dodaj

4. Testowanie połączenia:

```
ssh -T git@github.com
```

#### GitHub Web Interface - podstawowe funkcje

- Przeglądanie kodu (Code)
- Issues - zgłaszanie i śledzenie problemów
- Pull requests - proponowanie i recenzowanie zmian
- Actions - automatyzacja zadań (CI/CD)
- Projects - zarządzanie projektami
- Wiki - dokumentacja projektu
- Insights - statystyki i analityka

#### 4. Dobre praktyki tworzenia README i dokumentacji

##### README.md - wizytówka projektu

README to pierwszy dokument, który zobaczy osoba odwiedzająca Twoje repozytorium. Powinien zawierać:

##### 1. Nazwa i krótki opis projektu

```
# Nazwa Projektu
```

```
Krótki opis czym jest projekt i do czego służy.
```

##### 2. Spis treści (dla dłuższych README)

```
## Spis treści
```

- [Instalacja](#instalacja)
- [Użycie](#użycie)
- [Dokumentacja](#dokumentacja)
- [Licencja](#licencja)

##### 3. Instrukcje instalacji

```
## Instalacja
```

```
```bash
npm install moj-projekt
# lub
pip install moj-projekt
```
```

##### 4. Przykłady użycia

```
## Użycie
```

```
```python
import moj_modul
```

```
wynik = moj_modul.funkcja('argument')
print(wynik)
````
```

5. Dokumentacja lub link do dokumentacji
6. Informacje o licencji
7. Informacje o współtworzeniu (Contributing)
8. Badge'e (opcjonalnie) - wskaźniki statusu build, pokrycia testami, itp.

![Build Status](https://travis-ci.org/uzytkownik/repozytorium.svg?branch=main)

![Coverage](https://codecov.io/gh/uzytkownik/repozytorium/branch/main/graph/badge.svg)

#### Przykładowy szablon README.md

# Nazwa Projektu

Krótki opis projektu i jego funkcjonalności.

![Przykładowy screenshot](sciezka/do/screenshot.png)

#### ## Funkcjonalności

- Funkcjonalność 1
- Funkcjonalność 2
- Funkcjonalność 3

#### ## Instalacja

```
``bash
# Klonowanie repozytorium
git clone https://github.com/uzytkownik/repozytorium.git
```

```
# Przejście do katalogu projektu
cd repozytorium
```

```
# Instalacja zależności
npm install
````
```

#### ## Użycie

```
``javascript
const modul = require('moj-modul');
modul.funkcja('argument');
````
```

#### ## Dokumentacja

Pełna dokumentacja dostępna jest [tutaj](https://example.com).

#### ## Licencja

Ten projekt jest licencjonowany pod [licencją MIT](https://pl.wikipedia.org/wiki/Licencja\_MIT).

#### ## Autorzy

- Twoje Imię - [GitHub](https://github.com/twoj-login)

#### ## Podziękowania

- Podziękowania dla osób lub organizacji, które pomogły

### ### Inne ważne pliki dokumentacyjne

1. **\*\*LICENSE\*\*** - plik zawierający pełną treść licencji
2. **\*\*CONTRIBUTING.md\*\*** - wytyczne dla osób chcących współtworzyć projekt
3. **\*\*CHANGELOG.md\*\*** - historia zmian w projekcie
4. **\*\*CODE\_OF\_CONDUCT.md\*\*** - zasady zachowania dla społeczności projektu
5. **\*\*github/ISSUE\_TEMPLATE/\*\*** - szablony dla zgłaszanych problemów
6. **\*\*github/PULL\_REQUEST\_TEMPLATE.md\*\*** - szablon dla pull requestów

## 5. Ćwiczenia praktyczne

### Ćwiczenie 1: Utworzenie i konfiguracja repozytorium Git

1. Utwórz katalog na swój projekt: `mkdir moj-projekt`
2. Przejdź do tego katalogu: `cd moj-projekt`
3. Zainicjalizuj repozytorium Git: `git init`
4. Skonfiguruj swojego użytkownika Git:

```
git config user.name "Twoje Imię i Nazwisko"
git config user.email "twoj.email@example.com"
```

5. Sprawdź status repozytorium: `git status`

### Ćwiczenie 2: Pierwszy commit

1. Utwórz plik README.md w swoim repozytorium:  

```
echo "# Mój Projekt" > README.md
```
2. Dodaj kilka linii tekstu do README.md za pomocą edytora
3. Sprawdź status zmian: `git status`
4. Dodaj plik do poczekalni: `git add README.md`
5. Zatwierdź zmiany: `git commit -m "Dodano podstawowy README"`
6. Sprawdź historię commitów: `git log`

### Ćwiczenie 3: Łączenie z GitHub

1. Utwórz nowe repozytorium na GitHub (bez inicjalizacji README)
2. Połącz lokalne repozytorium z GitHub:  

```
git remote add origin https://github.com/twoj-login/moj-projekt.git
```
3. Wypchnij zmiany do zdalnego repozytorium:  

```
git push -u origin main
```
4. Odśwież stronę GitHub i sprawdź, czy Twoje zmiany są widoczne

#### Ćwiczenie 4: Wprowadzanie i zapisywanie zmian

1. Utwórz nowy plik w repozytorium, np. index.html:

```
<!DOCTYPE html>
<html>
<head>
  <title>Mój Projekt</title>
</head>
<body>
  <h1>Witaj w moim projekcie!</h1>
  <p>To jest strona startowa mojego projektu.</p>
</body>
</html>
```

2. Sprawdź status zmian: git status
3. Dodaj nowy plik do poczekalni: git add index.html
4. Zatwierdź zmiany: git commit -m "Dodano stronę startową"
5. Wypchnij zmiany do GitHub: git push

#### Ćwiczenie 5: Praca z README

1. Zmodyfikuj plik README.md, dodając pełny szablon z sekcjami:
  - Opis projektu
  - Instalacja
  - Użycie
  - Licencja
2. Dodaj znaczniki Markdown do formatowania tekstu
3. Zatwierdź i wypchnij zmiany do GitHub
4. Sprawdź, jak wygląda sformatowany README.md na GitHub

#### Ćwiczenie 6: Cofanie zmian

1. Wprowadź niepożądaną zmianę w pliku index.html
2. Sprawdź status: git status
3. Cofnij zmianę (jeśli nie została dodana do poczekalni):

```
git checkout -- index.html
```

4. Alternatywnie, jeśli zmiana została dodana do poczekalni:

```
git reset HEAD index.html
```

```
git checkout -- index.html
```

5. Sprawdź status po cofnięciu zmian: git status



## **Zadania do samodzielnego wykonania**

### **Zadanie 1: Utworzenie projektu w wybranej technologii**

Utwórz nowe repozytorium dla projektu w wybranej przez siebie technologii (Python, JavaScript, Java, itp.). Zainicjalizuj projekt, dodaj podstawowe pliki struktur katalogów i wypchnij zmiany na GitHub.

### **Zadanie 2: Rozbudowa README**

Stwórz kompleksowy plik README.md dla swojego projektu, zawierający wszystkie istotne sekcje. Wykorzystaj różne elementy składni Markdown.

### **Zadanie 3: Śledzenie i ignorowanie plików**

1. Utwórz plik .gitignore odpowiedni dla Twojej technologii
2. Dodaj do niego wzorce plików, które nie powinny być śledzone (np. pliki tymczasowe, katalogi z zależnościami)
3. Sprawdź, czy Git prawidłowo ignoruje określone pliki

### **Zadanie 4: Historia zmian**

Dokonaj serii zmian w swoim projekcie, wykonując commit po każdej istotnej zmianie. Przeglądaj historię commitów za pomocą różnych wariantów komendy git log.

### **Zadanie 5: Dodatkowe pliki dokumentacji**

Dodaj do swojego repozytorium co najmniej jeden dodatkowy plik dokumentacji (np. CONTRIBUTING.md lub LICENSE).

### **Bibliografia i dodatkowe materiały**

1. Oficjalna dokumentacja Git: [git-scm.com/doc](https://git-scm.com/doc)
2. Oficjalna książka Pro Git: [git-scm.com/book/pl/v2](https://git-scm.com/book/pl/v2)
3. GitHub Docs: [docs.github.com/en](https://docs.github.com/en)
4. GitHub Learning Lab: [github.com/apps/github-learning-lab](https://github.com/apps/github-learning-lab)
5. Interaktywny kurs Git: [learngitbranching.js.org](https://learngitbranching.js.org)
6. Cheat sheet Markdown: [github.com/adam-p/markdown-here/wiki/Markdown-Cheatsheet](https://github.com/adam-p/markdown-here/wiki/Markdown-Cheatsheet)