

Sprawozdanie nr 01

Temat: Wprowadzenie do GITa

Teoria:

GIT jest to rozproszony system kontroli wersji. Głównym zastosowaniem takiego systemu jest pomaganie w śledzeniu zmian oraz łączeniu zmian w projektach.

Główną zaletą samego GITa jest fakt, że działa on offline. Każda osoba pracująca nad danym projektem ma dostęp do pełnoprawnej kopii repozytorium. Można więc wykonywać zmiany w projektach, nie będąc połączonym do sieci i publikować je, kiedy tylko chcemy i dopiero wtedy wymagany będzie dostęp do sieci. Zabezpiecza to jednocześnie nasz projekt, w razie gdyby serwery online upadły.

GitHub to hostingowy serwis pozwalający na wgrywanie swoich projektów do repozytorium online. Wraz z GITem pozwalają one na wiele ułatwień, m. in.:

- Wiele osób może pracować nad jednym projektem, gdzie dokładnie widać, kto, kiedy i jakie zmiany wprowadził;
- W łatwy sposób można scalać pracę, a w razie konfliktów można je w prosty sposób rozwiązywać;
- Można udostępnić swój kod online do tzw. code review i pokazać swój kod innym;
- Mamy dostęp do pełnej historii zmian, co nawet w projektach jednoosobowych przydaje się w razie napotkania większych błędów.
- Możemy tam opublikować aplikację do użytku zewnętrznego od razu ustalając licencję, na której polegać będzie dystrybucja takiego programu.

Przebieg zadania:



Żeby zacząć pracę z GitHubem należy założyć konto na platformie github.com i zainstalować GIT na swoim urządzeniu.

Następnie można zacząć od stworzenia repozytorium online na GitHub. W tym celu klikamy przycisk „new” przy Repositories i wyświetla nam się okno tworzenia repozytorium:

Create a new repository


A repository contains all project files, including the revision history. Already have a project repository elsewhere? [Import a repository.](#)


Owner * Repository name *

 NasterQ / Repo 

Great repository names are short and memorable. Need inspiration? How about [vigilant-goggles?](#)

Description (optional)

☒  **Public**
Anyone on the internet can see this repository. You choose who can commit.

☐  **Private**
You choose who can see and commit to this repository.

Initialize this repository with:
Skip this step if you're importing an existing repository.

☒ **Add a README file**
This is where you can write a long description for your project. [Learn more.](#)

☒ **Add .gitignore**
Choose which files not to track from a list of templates. [Learn more.](#)

☒ **Choose a license**
A license tells others what they can and can't do with your code. [Learn more.](#)

Create repository

Jedyne wymagane pola to właściciel oraz nazwa.

Dodatkowo można dopisać opis repozytorium, zaznaczyć czy ma być publiczne czy prywatne (jako że repozytorium tworzone będzie na rzecz projektu dla Akademii Marynarki Wojennej i tym samym chcę, żeby osoba nadzorująca miała bezpośredni dostęp, wybieram opcję Public)

Można również od razu dodać plik README, .gitignore – w którym zamieszczone są informacje o plikach i folderach, które nie powinny być śledzone przez GIT oraz wybrać licencję, na której chcemy opublikować nasz kod.

Po stworzeniu repozytorium pojawia nam się okno Quick Setup:

Quick setup — if you've done this kind of thing before

Set up in Desktop or HTTPS SSH <https://github.com/NasterQ/Repo.git>

Get started by creating a new file or uploading an existing file. We recommend every repository include a README, LICENSE, and .gitignore.

...or create a new repository on the command line

```
echo "# Repo" >> README.md
git init
git add README.md
git commit -m "first commit"
git branch -M main
git remote add origin https://github.com/NasterQ/Repo.git
git push -u origin main
```

...or push an existing repository from the command line

```
git remote add origin https://github.com/NasterQ/Repo.git
git branch -M main
git push -u origin main
```

...or import code from another repository

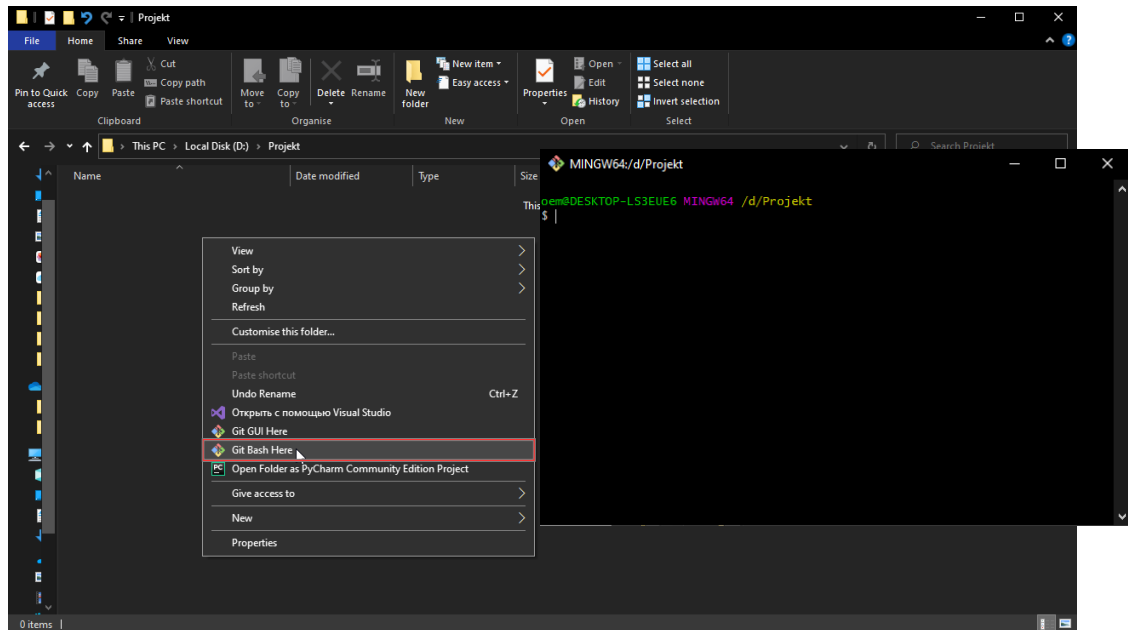
You can initialize this repository with code from a Subversion, Mercurial, or TFS project.

Import code

Z którego możemy się dowiedzieć jak podpiąć lokalne repozytorium do tego online.

W tym miejscu należy skonfigurować nasze repozytorium lokalne.

Wybieramy więc folder, który chcemy, żeby był naszym repozytorium, otwieramy menu kontekstowe i wybieramy opcję „Git Bash Here”



Otwiera nam się terminal widoczny po prawej stronie i to w nim wykonujemy resztę pracy.

Żeby stworzyć nowe repozytorium GIT używamy komendy: *git init*

Tym samym stworzył się właśnie ukryty katalog *.git*

W folderze Projekt, w katalogu „Sprawozdanie 1” mam umieszczony kod kalkulatora w języku C++. Jeśli teraz chciałbym dodać mój folder wraz z tym kodem muszę dodać oba te elementy do tzw. staging area. Robię to za pomocą komendy: *git add*

Cały czas mogę też sprawdzić co dokładnie dodaję do staging area, używając komendy *git status*.

```
oem@DESKTOP-LS3EUE6 MINGW64 /d/Projekt
$ git init
Initialized empty Git repository in D:/Projekt/.git/

oem@DESKTOP-LS3EUE6 MINGW64 /d/Projekt (master)
$ dir
Sprawozdanie\ 1

oem@DESKTOP-LS3EUE6 MINGW64 /d/Projekt (master)
$ ls Sprawozdanie\ 1/
calculator.cpp

oem@DESKTOP-LS3EUE6 MINGW64 /d/Projekt (master)
$ git status
On branch master

No commits yet

Untracked files:
  (use "git add <file>..." to include in what will be committed)
    Sprawozdanie 1/

nothing added to commit but untracked files present (use "git add" to track)

oem@DESKTOP-LS3EUE6 MINGW64 /d/Projekt (master)
$ git add Sprawozdanie\ 1/

oem@DESKTOP-LS3EUE6 MINGW64 /d/Projekt (master)
$ git status
On branch master

No commits yet

Changes to be committed:
  (use "git rm --cached <file>..." to unstage)
    new file:   Sprawozdanie 1/calculator.cpp

oem@DESKTOP-LS3EUE6 MINGW64 /d/Projekt (master)
$
```

Mam katalog Sprawozdanie 1, a w nim plik calculator.cpp

Sprawdzam status, GIT nie śledzi jeszcze żadnych plików

Dodaje mój folder z plikiem do staging area

Ponownie sprawdzam status, GIT już śledzi wybrane pliki i są one przygotowane do commita

Teraz gdy już pliki znajdują się w staging area mogą wykonać operację commit i wysłać je do repozytorium (*repository*). Wykonuję operację: `git commit -m`

Dopisek „-m” służy do dodania wiadomości (opisu) commita.

```
MINGW64/d/Projekt

oem@DESKTOP-LS3EUE6 MINGW64 /d/Projekt (master)
$ git commit -m "
> Uploading the calculator code"
[master (root-commit) 6385a70] Uploading the calculator code
1 file changed, 131 insertions(+)
 create mode 100644 Sprawozdanie 1/calculator.cpp

oem@DESKTOP-LS3EUE6 MINGW64 /d/Projekt (master)
$ git status
On branch master
nothing to commit, working tree clean

oem@DESKTOP-LS3EUE6 MINGW64 /d/Projekt (master)
$
```

Następnie, aby nasze pliki „wypchnąć”, tj. załadować do repozytorium na GitHub musimy dodać „remote”, czyli poinformować GIT, gdzie wysłać nasze pliki. Osiągamy to za pomocą komendy:

`git remote add origin [link do repozytorium GitHub]`

W ten sposób jesteśmy już gotowi, na wysłanie naszych plików do repozytorium na GitHub, robimy to za pomocą polecenia: `git push -u origin master`

gdzie master to nazwa naszej gałęzi (branch). Warto wspomnieć, że GitHub poleca zmianę gałęzi na main, ale nie jest to konieczne.

```
MINGW64/d/Projekt

oem@DESKTOP-LS3EUE6 MINGW64 /d/Projekt (master)
$ git status
On branch master
nothing to commit, working tree clean

oem@DESKTOP-LS3EUE6 MINGW64 /d/Projekt (master)
$ git remote add origin https://github.com/NasterQ/Repo.git

oem@DESKTOP-LS3EUE6 MINGW64 /d/Projekt (master)
$ git push -u origin master
Enumerating objects: 4, done.
Counting objects: 100% (4/4), done.
Delta compression using up to 8 threads
Compressing objects: 100% (2/2), done.
Writing objects: 100% (4/4), 1.13 KiB | 1.13 MiB/s, done.
Total 4 (delta 0), reused 0 (delta 0), pack-reused 0
To https://github.com/NasterQ/Repo.git
 * [new branch]      master -> master
Branch 'master' set up to track remote branch 'master' from 'origin'.

oem@DESKTOP-LS3EUE6 MINGW64 /d/Projekt (master)
$ |
```

Tym samym nasz folder oraz kod w nim zawarty są dostępne na naszym repozytorium GitHub.

