

Instrukcja

IDE, edytor kodu oraz system kontroli wersji

Integrated Development Environment (IDE) oraz edytor kodu

IDE (Integrated Development Environment) to zintegrowane środowisko programistyczne, które zapewnia programistom narzędzia do tworzenia, testowania, debugowania oraz wdrażania oprogramowania w jednym miejscu. Łączy w sobie różne funkcje tj. edytor kodu, narzędzia do zarządzania projektami, kompilatory, debuggery i inne, aby ułatwić i usprawnić proces tworzenia oprogramowania. **IDE** dostarcza kompleksowe rozwiązanie dla programistów, umożliwiając im efektywną pracę nad projektem.

Edytory kodu to narzędzia skoncentrowane głównie na edycji kodu źródłowego. Oferuje funkcje tj. podświetlanie składni, numerowanie linii, a także udostępnianie pewnych funkcji, ułatwiających pracę z kodem. Elementy te, nie czynią z nich narzędzi kompleksowych tj. **IDE**. Są zwykle bardziej minimalistyczne i elastyczne, co pozwala programistom dostosować je do swoich preferencji i potrzeb. Wiele edytorów kodu wymaga od programistów integrowanie różnych narzędzi tj. kompilatory czy też debuggery. Na potrzeby zajęć, przedstawione zostanie narzędzie **Visual Studio Code**, będące edytorem kodu.

Visual Studio Code

Visual Studio Code (VSC) to bezpłatny, lekki, zaawansowany i elastyczny edytor kodu opracowany przez firmę *Microsoft*. Narzędzie to, jest popularne wśród programistów do tworzenia różnego rodzaju aplikacji, w tym aplikacji webowych, mobilnych, desktopowych i wiele innych. Zdobył popularność ze względu na swoją prostotę, elastyczność oraz bogatą funkcjonalność. Jest używany przez programistów na różnych platformach (Windows, macOS, Linux). Do korzyści wynikających z jego używania możemy zaliczyć:

- darmowy,
- open source,
- wsparcia dla wielu języków programowania,
- lekki i szybki,
- integracja z systemem kontroli wersji,
- rozszerzenia,
- inteligentne funkcje edytora,
- debugowanie,
- wbudowany terminal,
- współpraca z chmurą.

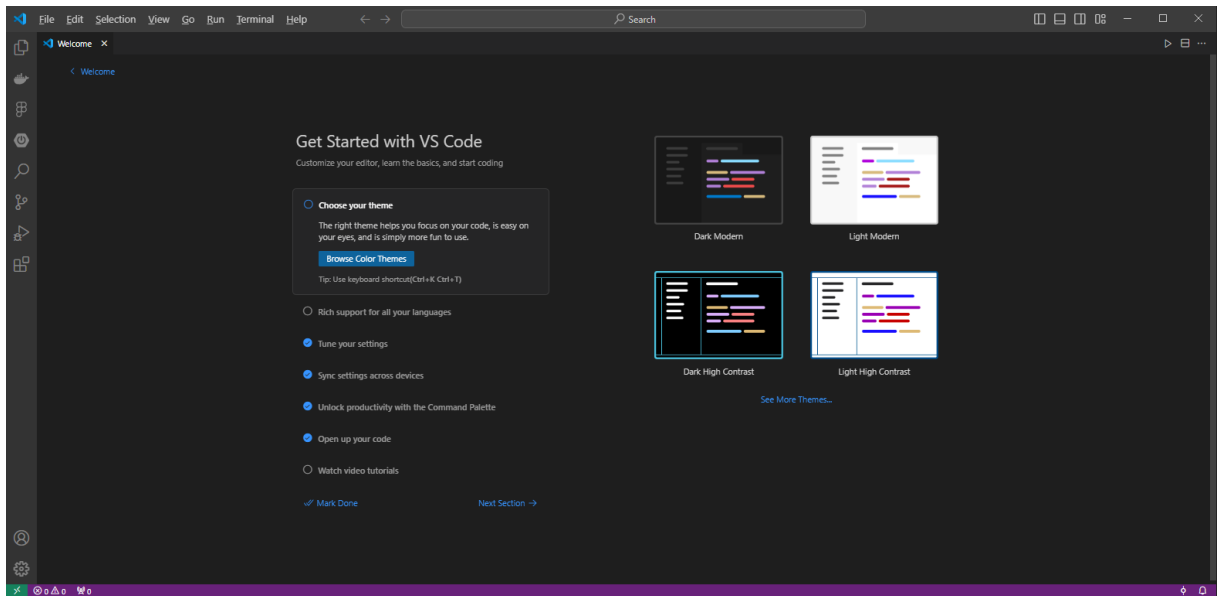
Instalacja

Aby zainstalować **Visual Studio Code** (na systemie operacyjnym Windows), należy wykonać następujące kroki:

- wejść na oficjalną stronę **VSC** (<https://code.visualstudio.com/>),
- na stronie głównej należy kliknąć przycisk do pobrania dla systemu Windows,
- po zakończeniu pobierania, należy uruchomić pobrany plik instalatora,

- po uruchomieniu instalatora, należy postępować zgodnie z instrukcjami na ekranie. Należy zaakceptować domyślne ustawienia, ale można również dostosować instalację według własnych preferencji,

- po zakończeniu instalacji, możemy uruchomić **Visual Studio Code** z menu Start lub korzystając z utworzonego na pulpicie skrótu.



System kontroli wersji

System kontroli wersji to narzędzie, które umożliwia śledzenie zmian w kodzie źródłowym i zarządzanie historią projektu. Głównym celem systemu kontroli wersji jest umożliwienie programistom efektywną współpracę, śledzenie ewolucji projektu oraz przywracanie poprzednich wersji kodu w przypadku błędów lub innych problemów. Istnieją dwa główne typy **VCS** (z ang. **Version Control System**):

- rozproszony system kontroli wersji (**DVCS**),
- centralizowany system kontroli wersji (**CVCS**).

Do podstawowych funkcji systemów kontroli wersji należą:

- zapis historii zmian,
- śledzenie zmian,
- rozgałęzianie i scalanie (tzw. branching oraz merging),
- współpraca zespołowa,
- śledzenie autorstwa.

Do popularnych systemów kontroli wersji należą:

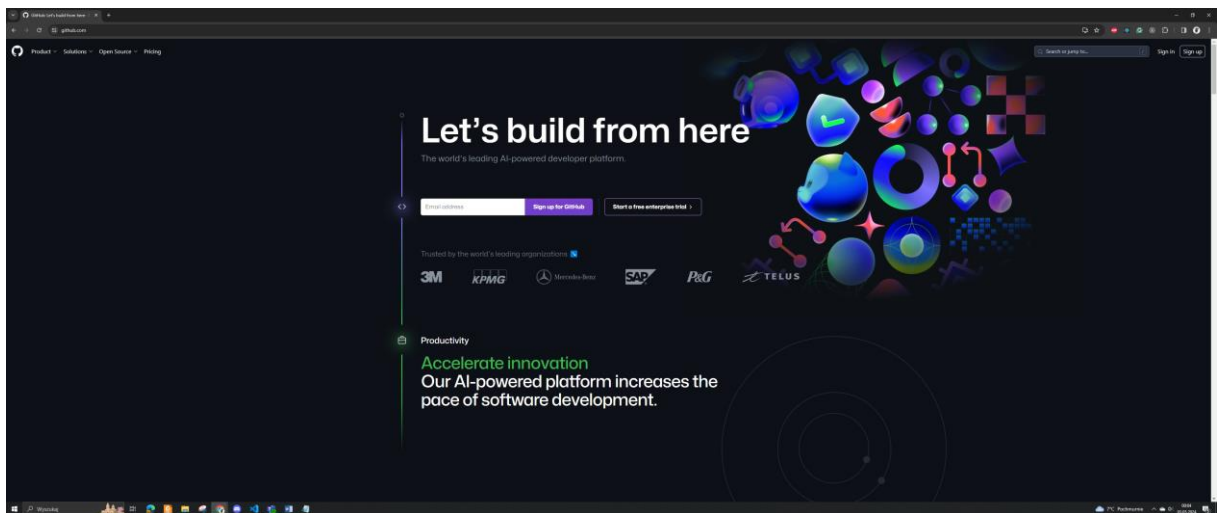
- Git,
- Subversion (SVN),
- Mercurial.

Systemy kontroli wersji to kluczowe narzędzia dla każdego projektu programistycznego, a ich stosowanie przynosi wiele korzyści, tj. lepszą współpracę zespołu, śledzenie błędów, łatwiejsze utrzymanie i zarządzanie projektem. Na potrzeby zajęć, opisany oraz wykorzystywany będzie system kontroli wersji **Git**.

Github

To platforma internetowa umożliwiająca zarządzanie projektem, śledzenie zmian w kodzie źródłowym oraz współpracę programistyczną. Narzędzie to, oparte jest na systemie kontroli wersji **Git**, które umożliwia programistom wspólną pracę nad projektem, śledzenie historii zmian, rozwiązywanie konfliktów i wiele innych.

Jest szeroko używany w społeczności programistycznej jako platforma do hostowania otwartoźródłowych projektów (**open source**), współpracy zespołowej i zarządzania kodem źródłowym. Wprowadza wiele narzędzi ułatwiających pracę programistyczną i wspiera otwarty charakter rozwoju wielu projektów. W celu korzystania z **Github** należy założyć na nim najpierw konto.

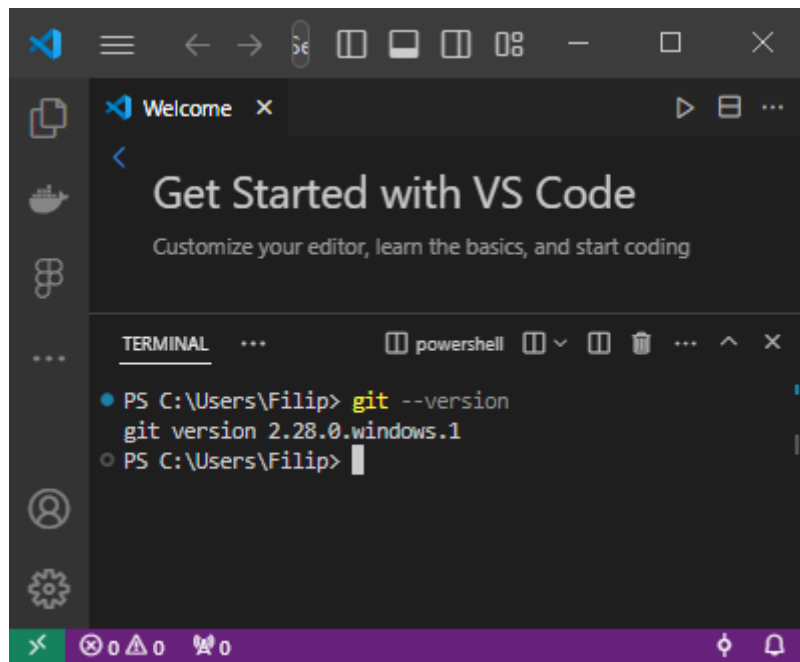


Instalacja

W pierwszej kolejności, należy przejść na oficjalną stronę **Git** (<https://git-scm.com/>) i pobrać go, za pomocą odpowiedniego przycisku, który pobierze wersję instalatora dla systemu Windows. Po pobraniu, należy uruchomić instalator. Zaproponuje on podczas procesu instalacji domyślne ustawienia, ale można dostosować je według własnych preferencji. Podczas instalacji, istnieje możliwość wyboru edytora tekstu. Dodatkowo, w trakcie instalacji można zaznaczyć opcję – *Git from the command line and also from 3rd-party software*, w celu dodania **Git** do zmiennej środowiskowej **PATH**. Krok ten jest zalecany, ponieważ w przyszłości, ułatwia korzystanie z **Git** z poziomu wiersza poleceń. Po zakończeniu instalacji, należy w terminalu użyć polecenia:

git --version

Zależnie od wyniku otrzymanego po użyciu tego polecenia, uzyskamy informację czy **Git** został zainstalowany poprawnie i jest wykrywany w systemie. W przypadku, jeżeli nie uzyskamy tej informacji, należy w pierwszej kolejności zrestartować terminale/edytor kodu/IDE. Jeżeli wykonanie tej czynności nie pomoże, należy sprawdzić czy wszystkie kroki zostały wykonane poprawnie.



Konfiguracja

Git wymaga prawidłowego skonfigurowania, żeby połączyć nasze konto **Git** z urządzeniem na którym wytwarzamy oprogramowanie. W pierwszym kroku, należy ustawić nazwę użytkownika oraz adres e-mail. Możemy tego dokonać za pomocą terminala systemowego (tzw. **cmd**) lub terminal dostępny w **VSC** a następnie, wpisując w terminalu następujące polecenie i podając swoje dane użytkownika:

git config --global user.name „Twoje Imię Nazwisko”

git config --global user.email „twój@email.com”

Jeżeli, nie wybrałeś edytora podczas instalacji, możesz to zrobić później za pomocą polecenia:

git config --global core.editor „code --wait”

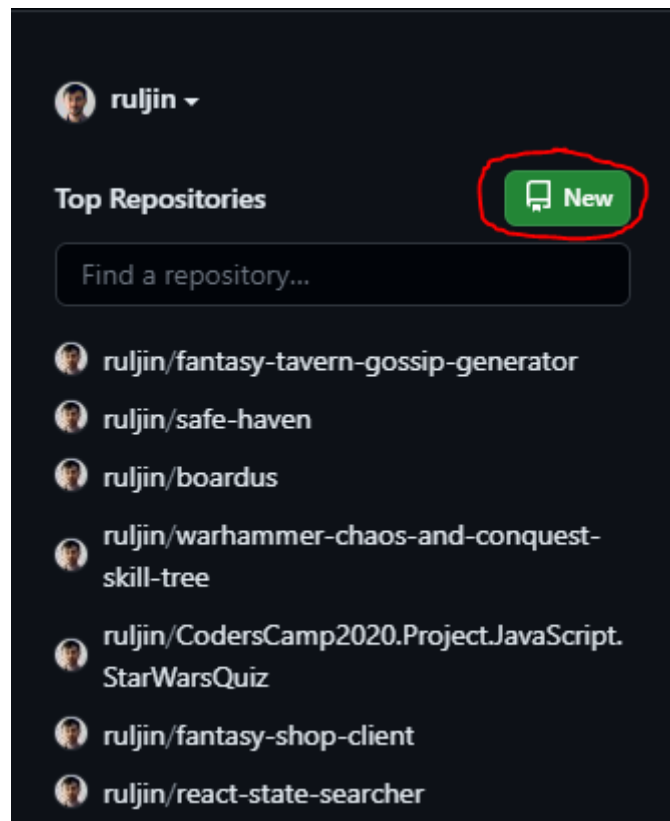
Polecenie to, ustawia **VSC** jako domyślny edytor. Konfiguracja ręczna, opisana w tym punkcie, nie powinna być wymagana, jeżeli wykonaliśmy prawidłowo poprzedni krok (patrz: **instalacja**) i podczas etapie, na którym zostaniemy poproszeni o podanie danych do naszego **Git**, je podaliśmy.

Integracja z Visual Studio Code

Git można skonfigurować z **VSC** za pomocą odpowiednich rozszerzeń. Zabieg ten, pozwala na dodanie dodatkowych narzędzi wspierających współpracę z **Git** w naszym **VSC** i nie jest krokiem obowiązkowym. Najpierw, należy otworzyć rozszerzenia w **VSC** (zakładka: **Extensions**) i wyszukanie w nich hasła **Git**. Po odnalezieniu rozszerzeń powiązanych z ni, należy je zainstalować. Następnie, gdy otworzymy projekt w **VSC** i projekt jest w repozytorium **Git**, **VSC** automatycznie rozpozna go i zacznie śledzić zmiany. Po tych krokach, **Git** powinien być zarówno skonfigurowany w systemie Windows jak i zintegrowany z **VSC**, umożliwiając korzystanie z poleceń **Git** zarówno z poziomu terminalu, jak i interfejsu graficznego dostępnego w **VSC**.

Tworzenie nowego repozytorium

Istnieje kilka sposobów tworzenia nowego repozytorium. Jedną z możliwości polega na zalogowaniu się na swoje konto na **Github**. W następnym kroku, należy kliknąć przycisk **new**, znajdujący się w lewym, górnym rogu głównej strony po zalogowaniu się.



Następnie, zostaniemy przekierowani na stronę z formularzem, gdzie możemy uzupełnić następujące informacje:

- repository template – czyli szablon repozytorium, jeżeli takowe posiadamy,
- owner – czyli jednostka organizacyjna, która będzie odpowiadać oraz zarządzać repozytorium (ma znaczenie gdy należymy do większej liczby jednostek organizacyjnych, niż nasze własne konto),
- public/private – po wybraniu jednostki organizacyjnej, pojawi się nam lista wyboru mówiąca o dostępności do tworzonego repozytorium,
- repository name – nazwa repozytorium,
- description – opcjonalne pole na krótki opis repozytorium,
- initialize this repository with – możliwość dodania do nowo tworzonego repozytorium, podstawowego pliku README.md, szablonu pliku .gitignore oraz licencji na podstawie, której regulującej co można z treścią tego repozytorium zrobić.

Następnie, po uzupełnieniu formularza, należy kliknąć create repository.

Create a new repository

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

Required fields are marked with an asterisk (*).

Repository template

No template ▾

Start your repository with a template repository's contents.

Owner *


Repository name *

Choose an owner ▾

/

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

Description (optional)

 Please choose an owner to see the available visibility options.

Initialize this repository with:

☐

Add a README file

This is where you can write a long description for your project. [Learn more about READMEs.](#)

Add .gitignore

.gitignore template: None ▾

Choose which files not to track from a list of templates. [Learn more about ignoring files.](#)

Choose a license

License: None ▾

A license tells others what they can and can't do with your code. [Learn more about licenses.](#)

Create repository

Utworzone w ten sposób repozytorium jest puste. Na koniec, zostaniemy przeniesieni na ekran, gdzie znajduje się krótki opis w jaki sposób, możemy połączyć nasz lokalny katalog z tym repozytorium z poziomu terminala oraz wypchnąć na niego zmiany, połączyć istniejące repozytorium i wypchnąć zmiany lub zaimportować kod z innego repozytorium.

The screenshot shows the GitHub interface for a new repository named 'lorem-ipsum'. At the top, there are buttons for 'Pin', 'Unwatch' (1), 'Fork', and 'Star' (0). Below this, there are two main sections: 'Set up GitHub Copilot' and 'Add collaborators to this repository'. The 'Quick setup' section offers options to 'Set up in Desktop', 'HTTPS', or 'SSH' with the URL 'https://github.com/ruljin/lorem-ipsum.git'. It also provides instructions on how to get started by creating a new file or uploading an existing file. The '...or create a new repository on the command line' section shows a series of terminal commands:

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

 The '...or push an existing repository from the command line' section shows:

```
git remote add origin https://github.com/ruljin/lorem-ipsum.git
git branch -M main
git push -u origin main
```

 The '...or import code from another repository' section mentions Subversion, Mercurial, or TFS projects.

Odtwarzanie repozytorium

W celu odtworzenia już istniejącego repozytorium, do którego mamy dostęp należy po wejściu na nie, kliknąć *code*.

The screenshot shows the GitHub interface for a repository named 'character-creator-client'. At the top, there are buttons for 'Pin', 'Unwatch' (2), and a 'Code' button. The 'Code' button is highlighted with a red circle. Below the repository name, there is a table of files and folders. The 'Code' button is located in the top right corner of the repository view. The 'Code' button is highlighted with a red circle. The 'Code' button is located in the top right corner of the repository view. The 'Code' button is highlighted with a red circle.

Spowoduje to rozwinięcie się listy, gdzie będziemy mieli dostęp do unikalnego linku, potrzebnego do użycia polecenia `git clone`. Następnie, należy otworzyć swój edytor w miejscu, gdzie chcemy repozytorium zdalne odtworzyć. Ostatni krok, polega na wpisaniu w terminalu **VSC** polecenia:

`git clone https://github.com/nazwa_uzytkownika/nazwa_repozytorium`

Polecenie to, odtworzy nasze repozytorium zdalne w miejscu, gdzie otwarty jest edytor i/lub gdzie wskazuje aktualnie terminal (jeżeli zmienialiśmy lokalizację za jego pomocą).

Podstawowe polecenia Git

Istnieje wiele poleceń w **Git**. Niżej, wymienione zostało kilka z nich:

- **`git init`** – inicjalizacja nowego repozytorium,
- **`git status`** – sprawdzenie stanu zmian,
- **`git add nazwa_pliku`** – dodanie plików do obszaru *stage*,
- **`git add *`** - dodanie wszystkich plików do obszaru *stage*,
- **`git commit -m „opis zmian”`** – zapisanie zmian w lokalnym repozytorium,
- **`git branch nazwa_galezi`** – utworzenie nowej gałęzi,
- **`git checkout nazwa_galezi`** – przełączenie się na inną gałąź,
- **`git checkout -b nazwa_galezi`** – utworzenie nowej gałęzi oraz przełączenie się na nią,
- **`git push origin nazwa_galezi`** – wysłanie zmian do repozytorium zdalnego.