**Source Control**

*Система контролю версій - це інструмент, який дозволяє вам зберігати та робити зміни в файлах та коді програми, що розробляється.*

Як використовується **SC**:

* **Зберігання історії змін:** СКВ дозволяє зберігати копії файлів та коду програми, що дозволяє легко відслідковувати та порівнювати зміни в проекті.
* **Контроль версій:** СКВ забезпечує контроль версій файлів та коду програм що розробляється, що дозволяє легко відновлювати старі версії проекту.
* **Резервне копіювання:** СКВ забезпечує збереження резервних копій проектів, що дозволяє відновлювати старі версії проектів.
* **Розробка у команді:** СКВ дозволяє декільком розробникам працювати над одним проектом, що забезпечує зручну систему роботи та обміну змінами.
* **Стеження за проблемами:** СКВ дозволяє стежити за проблемами в коді та легко відновлювати версії проекту до того моменту, коли проблема ще не виникла.
* **Зручна співпраця з клієнтами:** СКВ дозволяє зручно співпрацювати з клієнтами, які можуть бачити та перевіряти зміни в проекті, що дозволяє знижувати час який потрібний на узгодження замовлень та зменшує кількість недорозумінь.

**Git**

**Git** - #1 система контролю версій в світі.

Існує декілька великих провайдерів системи Git:

* **GitHub**
* **Bitbucket**
* **GitLab**
* **AWS**
* **CodeCommit**

**Git Repository** - це колекція файлів, що відноситься до одного проекту. За своєю структурою - це папка з файлами і іншими папками. В рамках репозиторію - гіт зберігає історію, гілки, доступи і т.д.

**Git Branches** - це вказівник на певний коміт.

**.gitignore** - це текстовий файл, який містить список файлів або каталогів, які Git повинен ігнорувати під час контролю версій файлів в репозиторії.

**Pull Request** - це запит на те, щоб вашу роботу в певній гілці подивились інші, залишили комент. та в випадку взаємного рішення гілку злили з іншою (merge).

**Головні команди Git CLI**

* **git init** - створює порожнє сховище Git або повторно ініціалізує існуюче.
* **git pull** - отримує з іншого репозиторію або локальної гілки та інтегрує з ними.
* **git fetch** - завантажує об’єкти та посилання з іншого репозиторію.
* **git commit** - записує зміни до сховища.
* **git push** - оновлює віддалені посилання разом із пов’язаними об’єктами.
* **git clone** - клонує сховища в новий каталог.
* **git add** - додає вміст файлу до індексу.
* **git checkout** - перемикає гілки або відновлює робочі файли дерева.
* **git branch** - список, створення або видалення гілок.
* **git diff** - показує зміни між комітами, комітом і робочим деревом і т.д.
* **git stash** - зберігає зміни в зміненому робочому каталозі.
* **git status** - показує статус робочого дерева.
* **git merge** - об’єднує два або більше історій розвитку разом.
* **git log** - показує журнал комітів.

**Another Source Controls:**

* **TFS -** *Team Foundation Servicer by Microsoft.*
* **Subversion -** *was popular before Git.*
* **Mercurial -** *launched together with Git.*
* **ClearCase -** *IBM’s source control system.*

**C#**

**Етапи від створення прог. до виконання:**

* *Зберігаємо файл.*
* *Будуємо проект(и)/солюшен.*
* *Компілятор генерує DLL/EXE в мові IL (Intemidiate Language).*
* *Під час запуску програми JIT (Just-In-Time) компілятор транслює IL в машинний код.*

**SDK (*Software Development Kit*)** - інструмент для створення софту. Для створення С# застосунків використовується ***.NET SDK***