ЛАБОРАТОРНА РОБОТА

Тема: Використання Git та GitHub у PHP-проєктах

🎯 Мета роботи

Навчитися:

- використовувати Git для контролю версій PHP-проєкту;
- створювати та використовувати гілки;
- працювати з GitHub;
- виконувати базовий workflow розробника.

🧰 Обладнання та програмне забезпечення

- ОС: Windows
- Git
- GitHub (обліковий запис)
- VS Code (або інший редактор)
- PHP 7+

📂 Завдання

1. Створити локальний PHP-проєкт
2. Ініціалізувати Git-репозиторій
3. Створити віддалений репозиторій на GitHub
4. Створити гілку для нової функції
5. Додати PHP-файл у гілці
6. Виконати злиття гілки з main
7. Завантажити проєкт на GitHub

📝 Хід роботи

🔷 Крок 1. Створення PHP-проєкту

Створити папку проєкту:
php-git-lab/
Створити файл index.php:

```
<?php
echo "Hello, Git and GitHub!";
```

🔷 Крок 2. Ініціалізація Git

У папці проєкту виконати:

```
git init
git add .
git commit -m "Initial commit"
```

🔷 Крок 3. Файл .gitignore

`.gitignore` — це спеціальний текстовий файл у проєктах Git, який вказує системі, які файли та папки слід **ігнорувати** (не відстежувати і не додавати

до репозиторію), що є критично важливим для виключення тимчасових файлів, логів, залежностей, налаштувань IDE та інших неважливих для проєкту елементів. Він використовує шаблони для визначення, що виключити, і допомагає тримати репозиторій чистим та ефективним.

Що він робить

- **Виключає непотрібне:** Запобігає випадковому додаванню скомпільованих файлів (`.class` , `.o`), лог-файлів (`.log`), файлів налаштувань (`.env` , `.vscode` , `*.swp`), залежностей (`node_modules`) до системи контролю версій.
- **Гнучкий синтаксис:** Дозволяє використовувати шаблони, підстановки (`*` , `?`) та ігнорувати цілі каталоги за їхніми назвами.

Як це працює

1. **Створення:** Ви створюєте файл `.gitignore` (іноді у корені проєкту) і додаєте туди назви файлів або шаблони, які потрібно ігнорувати, по одному на рядок.
2. **Шаблони:**
   1. `*.log` – ігнорує всі файли з розширенням `.log` .
   2. `build/` – ігнорує всю папку `build` .
   3. `!important.log` – *не* ігнорувати файл `important.log` , навіть якщо попередній шаблон його захопив.
3. **Автоматично:** Коли ви виконуєте команди типу `git add .` , Git читає цей файл і пропускає все, що в ньому зазначено.

Створити файл .gitignore:
/vendor
.env
*.log

🔷 Крок 4. Створення репозиторію на GitHub

1. Увійти на GitHub
2. Натиснути **New repository**
3. Вказати назву php-git-lab
4. Створити репозиторій

Підключення GitHub:
git branch -M main
git remote add origin https://github.com/USERNAME/php-git-lab.git
git push -u origin main

🔷 Крок 5. Створення гілки

Створити гілку для сторінки профілю:
git checkout -b feature-profile

🔷 Крок 6. Робота в гілці

Створити файл profile.php:
<?php
echo "Profile page";
Зберегти зміни:
git add profile.php
git commit -m "Add profile page"

🔷 Крок 7. Злиття гілки
git switch main
git merge feature-profile

🔷 Крок 8. Завантаження на GitHub
git push

✅ Результат роботи
- створено PHP-проєкт;
- ініціалізовано Git-репозиторій;
- створено гілку для нової функції;
- виконано злиття гілок;
- проєкт опубліковано на GitHub.

❓ Контрольні питання
1. Що таке Git?
2. Що таке GitHub?
3. Для чого використовують гілки?
4. Чим відрізняється git commit від git push?
5.