**Контрольні запитання**

**1. Що являє собою система контролю версій?**

Система контролю версій (Version Control System, VCS) — це програма, яка фіксує зміни у файлах з часом, дозволяє відновлювати попередні версії коду, відслідковувати зміни, працювати в команді над одним проектом без втрати даних.  
**Приклад:** Git — найпопулярніша VCS.

**2. Основні команди для роботи з Git та похідних:**

* git init — ініціалізація нового репозиторію.
* git clone <url> — клонування репозиторію.
* git status — перегляд стану файлів.
* git add <file> — додати файл до індексації.
* git commit -m "message" — зафіксувати зміни з коментарем.
* git push — відправити коміти на сервер (GitHub).
* git pull — завантажити останні зміни з репозиторію.
* git branch — перегляд гілок.
* git checkout <branch> — переключитися на іншу гілку.
* git merge <branch> — об'єднати гілку з поточною.

**3. Конфлікти злиття версій. Практики вирішення конфліктів:**

**Конфлікт злиття** виникає, коли дві гілки змінюють один і той самий рядок коду або один і той самий файл по-різному.  
**Як вирішувати:**

* Git вкаже місце конфлікту в коді через спецсимволи (<<<<<<<, =======, >>>>>>>).
* Розглянути зміни і вручну залишити правильний варіант.
* Після виправлення:

git add <файл>

git commit

* Практики: частіше оновлювати локальну гілку (git pull), працювати в окремих гілках, обговорювати важливі зміни в команді.

**4. Призначення API. Робота з API-бібліотеками:**

**API (Application Programming Interface)** — це набір правил та протоколів для взаємодії різних програмних компонентів.  
**Призначення:** дозволяє одній програмі "спілкуватися" з іншою без знання внутрішніх деталей реалізації.  
**Робота з API-бібліотеками:**

* Бібліотеки (requests, axios, fetch тощо) полегшують надсилання запитів до API.
* За допомогою них можна отримувати дані, надсилати їх, обробляти відповіді серверів.

**5. Методи HTTP-запитів (безпечні, ідемпотентні, піддаються кешуванню):**

| **Метод** | **Безпечний** | **Ідемпотентний** | **Кешується** | **Призначення** |
| --- | --- | --- | --- | --- |
| GET | ✅ | ✅ | ✅ | Отримати дані |
| POST | ❌ | ❌ | ❌ | Створити нові дані |
| PUT | ❌ | ✅ | ❌ | Повністю оновити ресурс |
| PATCH | ❌ | ❌ | ❌ | Частково оновити ресурс |
| DELETE | ❌ | ✅ | ❌ | Видалити ресурс |
| HEAD | ✅ | ✅ | ✅ | Отримати тільки заголовки відповіді |
| OPTIONS | ✅ | ✅ | ❌ | Дізнатися, які методи підтримує сервер |

**Примітка:**

* *Безпечний* = не змінює стан сервера.
* *Ідемпотентний* = багаторазовий виклик дає той самий результат.

**6. JSON та XML. Структури об'єктів. Їх особливості:**

| **Формат** | **JSON** | **XML** |
| --- | --- | --- |
| Синтаксис | Легкий, читається людиною | Більш громіздкий, теговий |
| Структура | Об'єкти ({}) і масиви ([]) | Елементи з тегами (<tag>...</tag>) |
| Форматування | Парситься легко у всіх мовах програмування | Потребує спеціальних парсерів |
| Використання | Сучасні веб-додатки, мобільні застосунки | Старі системи, банківські сервіси |

**JSON** більше підходить для веб-розробки через свою простоту.  
**XML** надійніший для складних структур з атрибутами.

**7.Засоби тестування API (Postman, cURL, Burp Suite):**

* **Postman** — найзручніший інструмент для тестування API без коду. Дозволяє відправляти запити, переглядати відповіді, автоматизувати тести.
* **cURL** — консольна програма для відправки HTTP-запитів. Дуже корисна для автоматизації через термінал.
* **Burp Suite** — професійний інструмент для тестування безпеки веб-додатків (перехоплення запитів, атаки на API).