**Теоретичне завдання**

**1. Для чого використовується версійності API?**

Версіювання API використовується для збереження сумісності між різними версіями програмного інтерфейсу. Це означає, що коли публікується нова версія API, можна внести зміни до вже існуючих методів, додати нові методи, або навіть видалити деякі з методів.

Використання версій API дозволяє забезпечити зв'язок між клієнтами і сервером при зміні API, при цьому не впливаючи на функціонал, який вже використовується. Таким чином, версіювання API дозволяє уникнути розриву функціональності між клієнтами та сервером та забезпечити поступове оновлення API.

**2. Що таке контракт між серверною і клієнтською частиною?**

Контракт між серверною і клієнтською частиною − це узгодження інтерфейсу та параметрів взаємодії між клієнтом і сервером. В контракті визначається, які методи та параметри повинні бути використані для спілкування між клієнтом та сервером, а також які формати даних повинні бути передані взаємодії.

Контракт може бути представлений різними форматами, зокрема він може бути визначений у вигляді специфікації API, файлу документації, або навіть у вигляді коду. Його головна мета полягає в забезпеченні стабільної взаємодії між клієнтом та сервером незалежно від того, які зміни будуть внесені до реалізації. Контракт дозволяє розділити розробку клієнтської і серверної частини, щоб розробники могли працювати над різними аспектами додатка, не втручаючись один в одного.

**3. Які є шляхи додавання версійності до проекту ASP.NET WebAPI?**

Додавання версійності до проекту ASP.NET WebAPI можна здійснити за допомогою наступних шляхів:

URI-based Versioning: цей підхід використовується для додавання версії до URI запиту. Наприклад, у випадку, коли маємо контролер ProductsController, версія може бути визначена через використання наступного шаблону маршруту: /api/v1/products. Щоб додати нову версію, можна використовувати шаблон /api/v2/products. Цей підхід є простим і досить популярним.

Query-based Versioning: у цьому підході версія визначається через параметр запиту v або інший параметр. Наприклад, у випадку з ProductsController запит може виглядати так: /api/products?v=1.

Header-based Versioning: у цьому підході версія визначається через заголовок HTTP запиту. Наприклад, можна використовувати заголовок X-Version: 1.

Media-Type-based Versioning: у цьому підході версія визначається через тип медіа, який передається в запиті. Наприклад, можна використовувати тип медіа application/vnd.company.product.v1+json.

Крім цих підходів, можливо використовувати різні бібліотеки та фреймворки, такі як Microsoft.AspNetCore.Mvc.Versioning, які дозволяють забезпечити версію контролерів та дій через атрибути або конфігураційні файли.

**4. У яких випадках необхідно додавати версійність?**

1. Зміни в API, які можуть зруйнувати існуючі клієнтські додатки, які вже використовують API. У цьому випадку введення нової версії API дозволяє зберегти сумісність з існуючими додатками.
2. Потреба у внесенні змін в API, але збереження сумісності з існуючими клієнтськими додатками не є обов'язковим. Введення нової версії API в цьому випадку може бути корисним, щоб забезпечити сумісність з новими клієнтськими додатками та дозволити їм використовувати оновлену версію API.
3. Потреба у внесенні змін в API без додавання нової версії. У деяких випадках можливо здійснити зміни в API без введення нової версії, наприклад, якщо зміни не змінюють існуюче поведінку API. Однак, якщо зміни можуть вплинути на існуючі клієнтські додатки, введення нової версії може бути безпечнішою опцією.