

Варіант 1

1. Чим клієнт відрізняється від сервера у клієнт-серверній архітектурі Вебу?
2. Що таке SQL-ін'єкція і як підготовлені запити (prepared statements) допомагають від неї захиститися?
3. У чому полягає різниця між автентифікацією та авторизацією?
4. Що таке монолітна архітектура застосунку і які її основні ознаки?

Варіант 2

1. Яке призначення основних HTTP-методів GET та POST і чим вони принципово відрізняються?
2. Чому не можна довіряти даним, які надходять від клієнта (з форми чи URL), і що таке їх валідація?
3. За що відповідає кожен з компонентів архітектурного патерну MVC (Model, View, Controller)?
4. Чим синхронна міжсервісна комунікація (наприклад, через REST) відрізняється від асинхронної (через черги повідомлень)?

Варіант 3

1. Що означає stateless-природа (відсутність стану) протоколу HTTP?
2. Яке призначення формату обміну даними JSON і чому він став стандартом для сучасних вебсервісів?
3. Для чого потрібні транзакції при роботі з базами даних (наприклад, при банківському переказі коштів)?
4. Що таке мікросервісна архітектура і які її ключові принципи?

Варіант 4

1. Назвіть основні класи статус-кодів HTTP (2xx, 3xx, 4xx, 5xx) та поясніть їхнє загальне значення.
2. Як за допомогою PHP прочитати дані у форматі JSON, які клієнт надіслав у «сирому» тілі POST-запиту?
3. Як працюють сесії в PHP і для чого обов'язково використовується функція `session_start()`?
4. Що таке патерн Front Controller (єдина точка входу) і яку проблему маршрутизації він вирішує?

Варіант 5

1. Що таке вихідний буфер (output buffering) у PHP і для чого використовується функція `ob_start()`?
2. Чим розширення PDO для роботи з базами даних відрізняється від «рідного» `mysqli`?
3. Чому паролі користувачів у базі даних необхідно зберігати у вигляді хешу, а не у відкритому тексті?
4. Що таке каскадний збій у міжсервісній взаємодії та як патерн `Circuit Breaker` (запобіжник) допомагає йому запобігти?

Варіант 6

1. Що таке `Server-Side Rendering (SSR)` і `Client-Side Rendering (CSR)` та в чому різниця між ними?
2. Опишіть призначення функції `htmlspecialchars()` і поясніть, від яких атак вона захищає (наприклад, `XSS`).
3. Яку функцію виконує маршрутизатор (`Router`) у вебзастосунку?
4. За яких умов перехід від моноліту до мікросервісів є технічно та організаційно виправданим?

Варіант 7

1. Як за допомогою розширення `CURL` у PHP виконати звернення до зовнішнього API?
2. Поясніть призначення основних CRUD-операцій (`Create`, `Read`, `Update`, `Delete`) при роботі з даними.
3. Як токени формату `JWT (JSON Web Token)` вирішують проблему автентифікації для API порівняно з класичними сесіями?
4. Чому при виклику зовнішнього сервісу через мережу (`REST API`) обов'язково потрібно встановлювати таймаут?

Варіант 8

1. Які компоненти входять до складу дистрибутиву `XAMPP` і за що вони відповідають?
2. Яку інформацію про завантажений файл містить суперглобальний масив `$_FILES`?
3. Чому в добре спроектованому MVC-застосунку компонент `View` (Представлення) не повинен містити SQL-запитів до бази даних?
4. Що означає принцип «децентралізованих даних» у мікросервісній архітектурі?

Варіант 9

1. Які функції PHP використовуються для швидкого читання та запису всього вмісту текстового файлу?
2. В чому полягає різниця між методами `query()` та `prepare()` в PDO при виконанні запитів до бази даних?
3. Яку роль виконує протокол OAuth2 і для яких типових сценаріїв він застосовується (наприклад, «Увійти через Google»)?
4. Опишіть принцип роботи черги повідомлень за патерном Publish/Subscribe.

Варіант 10

1. Чому PHP-файл потрібно відкривати через браузер за адресою `http://localhost`, а не просто подвійним клацанням миші у системі?
2. Для чого при завантаженні файлу на сервер використовується спеціальна функція `move_uploaded_file()`, а не просте переміщення?
3. Яку проблему вирішує використання механізму автозавантаження класів (autoloading) у PHP-проектах?
4. Які основні складнощі та недоліки можуть виникнути при переході на мікросервісну архітектуру?