

Міністерство освіти і науки України  
Вінницький національний технічний університет  
Факультет менеджменту та інформаційної безпеки

Кафедра МБІС

## **ЛАБОРАТОРНА РОБОТА №5**

з дисципліни «Основи програмування захищених інформаційних систем»  
на тему: «Знайомство з Node.js та Express.js»

Студента 1 курсу групи УБ-246  
напряму підготовки 125 спеціальності  
«Управління Інформаційною Безпекою»

Червоний Назар Валентинович

Перевірив:

ас. каф. МБІС К.В. Безпальй

Вінниця 2025

## Загальні вимоги:

1. Використати **Express.js** для створення сервера.
2. Всі дані зберігаються в **масиві в пам'яті**.
3. Реалізувати CRUD-операції:
  - **C** (Create) – Додавання нової сутності
  - **R** (Read) – Отримання списку та окремих елементів
  - **U** (Update) – Оновлення даних сутності
  - **D** (Delete) – Видалення сутності
4. Обов'язково обробляти **помилки** (наприклад, спроба отримати неіснуючий запис).
5. **Крім розробки API**, необхідно створити **HTML-інтерфейс** для роботи з API.
6. Використати **Materialize CSS** або інший UI-фреймворк (Bootstrap, Bulma тощо) для покращення візуального відображення.
7. **HTML-сторінки** мають підтримувати такі дії:
  - Відображення **списку** об'єктів із API (**GET**).
  - Можливість **додавати** новий об'єкт (**POST**).
  - Можливість **редагувати** об'єкт (**PUT**).
  - Видалення об'єкта зі списку (**DELETE**).
8. Використання **JavaScript (Fetch API)** для запитів до сервера.
9. Код HTML має бути **відокремлений** від логіки API (сервер працює окремо).
10. Може бути **дві HTML-сторінки**:
  - Одна для **відображення списку**.
  - Інша для **додавання/редагування**.

## Варіант 12:

### 1. index.html

- Таблиця зі списком ігор: назва, мінімальна і максимальна кількість гравців, час гри, дії
- Кнопка "Add New Game" веде на form.html
- Дані підвантажуються через JavaScript у `<tbody id="game-list">`

### 2. form.html

- Форма додавання або редагування гри
- Поля: Name, MinPlayers, MaxPlayers, PlayTime
- Кнопки "Save" і "Cancel"
- Використовується приховане поле для збереження індексу гри під час редагування

### 3. script.js

- Використовується Fetch API для запитів до сервера
- `renderGameList()` – виводить список ігор, додає кнопки "Edit" та "Delete"
- `setupForm()` – налаштовує логіку форми редагування/додавання
- Дані зберігаються в оперативній пам'яті сервера, не в localStorage

### 4. index.js (сервер)

- Налаштування Express.js, підключення JSON-парсера та статичних файлів
- Масив `games` містить об'єкти: `{ id, name, minPlayers, maxPlayers, playTime }`

Маршрути:

- GET /games — список ігор
- GET /games/:id — гра за ID
- POST /games — додавання гри
- PUT /games/:id — редагування гри
- DELETE /games/:id — видалення гри

**Висновок:** У ході виконання цієї лабораторної роботи було створено веб-додаток для управління каталогом автомобілів. Основна мета полягала у розробці клієнт-серверної архітектури з використанням **HTML, CSS (Materialize), JavaScript, Node.js та Express.js**.

**Досягнуті результати:**

**1. Розроблено фронтенд:**

Сторінка `index.html` містить таблицю для відображення списку автомобілів.

`form.html` надає можливість додавати або редагувати записи.

Використано **Materialize CSS** для покращеного дизайну.

**2. Реалізовано логіку роботи з даними (`script.js`):**

Використання **localStorage** для збереження даних.

Функції для додавання, редагування та видалення автомобілів.

**3. Створено серверну частину на Node.js (`server.js`, `cars.js`):**

Веб-сервер працює на **Express.js**.

Реалізовані API-ендпоїнти для отримання, додавання та видалення автомобілів.