ВІЙСЬКОВИЙ ІНСТИТУТ ТЕЛЕКОМУНІКАЦІЇ ТА ІНФОРМАТИЗАЦІЇ ІМЕНІ ГЕРОЇВ КРУТ

Кафедра комп’ютерних інформаційних технологій

КУРСОВИЙ ПРОЄКТ

з дисципліни: «Технологія розробки програмного забезпечення військового призначення»

на тему: “Розробка RESTfull додатку на основі технології Node.js для автоматизації діяльності обліку транспортних засобів автопарку інституту”

Виконав: курсант навчальної групи 304

солдат Михайло КЛЬОЦ

Перевірив: викладач кафедри №22 п

майор Сергій РОМАНЕНКО

**КИЇВ-2025**

**АНОТАЦІЯ**

**курсової роботи на тему:**

**«Розробка RESTfull додатку на основі технології Node.js для автоматизації діяльності обліку транспортних засобів автопарку інституту»**

Курсова робота містить: 21 сторінку, 5 рисунків,   
10 джерел, 5 додатків.

Робота присвячена проектуванню і розробці RESTful додатку для автоматизації процесу обліку транспортних засобів автопарку інституту з використанням технології Node.js. У розробці застосовано концепції об’єктно-орієнтованого програмування та сучасні підходи до створення вебсервісів.

Програмний додаток реалізує функціонал для збереження, перегляду, редагування та фільтрації інформації про транспортні засоби. Особливу увагу приділено реалізації зручного API для взаємодії з базою даних та забезпечення надійності й масштабованості системи.

***ABSTRACT***

***of the course work on theme:***

***"Using the C # language to develop software applications"***

The term paper contains: 21 pages, 5 figures, 10 source, 5 appendices..

The work is dedicated to the design and development of a RESTful application for automating the process of accounting for the institute's vehicle fleet using Node.js technology. The development employs object-oriented programming concepts and modern approaches to web service creation.

The application implements functionality for storing, viewing, editing, and filtering information about vehicles. Special attention is given to implementing a user-friendly API for interaction with the database and ensuring the system's reliability and scalability.

ЗМІСТ

[РОЗДІЛ 1. АНАЛІЗ СПОСОБІВ І МЕТОДІВ АВТОМАТИЗАЦІЇ ОБЛІКУ ТРАНСПОРТНИХ ЗАСОБІВ 7](#_Toc202275924)

[1.1 Функціонал програми 7](#_Toc202275925)

[1.2 Недоліки існуючих програмних рішень і напрямки їх вдосконалення 8](#_Toc202275926)

[РОЗДІЛ 2. ПРОЕКТУВАННЯ ТА ОСОБЛИВОСТІ РОЗРОБКИ 10](#_Toc202275927)

[2.1 Вимоги до програми 10](#_Toc202275928)

[2.2 Схеми алгоритмів основних методів 10](#_Toc202275929)

[2.3 Опис технологій, що використовуються для вирішення завдання 12](#_Toc202275930)

[2.3.1 Node.js: 12](#_Toc202275931)

[2.3.2 Express.js 12](#_Toc202275932)

[2.3.3 MongoDB 12](#_Toc202275933)

[2.3.4 REST API 13](#_Toc202275934)

[РОЗДІЛ 3. ОСОБЛИВОСТІ ВИКОРИСТАННЯ ПРОГРАМНОГО ДОДАТКУ 14](#_Toc202275935)

[3.1 Реєстрація та авторизація 14](#_Toc202275936)

[3.2 Головне меню: Облік транспорту 15](#_Toc202275937)

[3.3 Історія обслуговування транспортних засобів 16](#_Toc202275938)

[3.4 Додавання нового транспортного засобу 16](#_Toc202275939)

[ВИСНОВОК 19](#_Toc202275940)

[СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ 21](#_Toc202275941)

ВСТУП

**Актуальність роботи.** У сучасних умовах ефективне управління транспортними засобами є важливою складовою функціонування будь-якого підприємства, зокрема й інституту. Облік транспортних засобів, відстеження їхнього технічного стану, планування технічного обслуговування та аналіз експлуатаційних показників потребують значних витрат часу та зусиль при ручному веденні документації.

Актуальним є питання автоматизації цього процесу шляхом створення зручного та функціонального програмного забезпечення, яке дозволить зменшити кількість помилок, пришвидшити обробку даних і забезпечити централізований доступ до всієї інформації про транспортні засоби.

Функціонування запропонованого програмного додатку передбачає автоматизацію обліку транспортних засобів автопарку інституту шляхом розробки RESTful додатку з використанням технології Node.js. Такий підхід дозволить створити зручний інтерфейс взаємодії з базою даних та забезпечити ефективне управління інформацією.

**Мета роботи:** розробка програмного модулю, який забезпечить автоматизований облік транспортних засобів, дозволить ефективно управляти даними про стан, переміщення, обслуговування автопарку інституту, а також знизить навантаження на персонал, відповідальний за ведення облікової документації.

Виходячи з мети роботи, виникають наступні завдання:

* Проаналізувати існуючі підходи та технології до автоматизації обліку транспортних засобів, включаючи вже реалізовані системи та їх функціональні можливості.
* Розробити структуру бази даних для зберігання інформації про транспортні засоби, технічне обслуговування, ремонти, статуси та інші параметри.
* Реалізувати RESTful API на базі Node.js для взаємодії з даними автопарку.
* Створити інтерфейс користувача або продемонструвати приклад клієнтської взаємодії з API для перегляду, фільтрації, додавання, редагування та видалення даних про транспортні засоби.
* Підготувати інструкцію користувача для роботи з розробленим програмним модулем.

**Об’єкт дослідження:** процес обліку та управління інформацією про транспортні засоби автопарку інституту.

**Предмет дослідження:** методи та засоби автоматизації обліку транспортних засобів з використанням сучасних вебтехнологій, зокрема RESTful архітектури та технології Node.js.

# АНАЛІЗ СПОСОБІВ І МЕТОДІВ АВТОМАТИЗАЦІЇ ОБЛІКУ ТРАНСПОРТНИХ ЗАСОБІВ

## Функціонал програми

Програмний модуль на тему «Автоматизація обліку транспортних засобів автопарку інституту» може мати різноманітний функціонал залежно від специфіки задач, які необхідно вирішувати. Основні функції, які можуть бути реалізовані, включають:

**Реєстрація та авторизація користувачів:**

* + Можливість створення облікового запису з унікальним логіном і паролем.
  + Розмежування доступу до функціоналу додатку (адміністратор, персонал/перегляд інформації).

**Облік транспортних засобів:**

* + Додавання нових записів про транспортні засоби (тип, марка, модель, рік випуску, державний номер).
  + Зберігання інформації про технічний стан, статус.

**Редагування та перегляд даних:**

* + Можливість редагування існуючих записів.
  + Фільтрація та сортування транспортних засобів за різними параметрами (технічний стан, тип, марка, модель).

**Ведення журналу обслуговування:**

* + Фіксація інформації про технічні огляди, ремонти, заміну комплектуючих.
  + Автоматичне сповіщення про заплановане ТО.

## Недоліки існуючих програмних рішень і напрямки їх вдосконалення

Існують певні недоліки у деяких існуючих програмних рішень, призначених для обліку транспортних засобів автопарку. Основні з них:

* Обмежений функціонал: Багато рішень надають лише базові можливості – зберігання інформації про транспорт, без підтримки детального ведення історії обслуговування, аналітики, автоматичних сповіщень тощо.
* Складний або застарілий інтерфейс: Деякі програми мають неінтуїтивний або морально застарілий інтерфейс, що ускладнює роботу персоналу з системою, особливо за великої кількості записів.
* Відсутність інтеграції з іншими системами: Багато платформ не підтримують інтеграцію з внутрішніми інформаційними системами установ, наприклад з бухгалтерським або логістичним програмним забезпеченням.
* Низький рівень захисту даних: Відсутність надійної авторизації, шифрування чи ролей доступу може призвести до витоку або несанкціонованої зміни інформації.
* Недостатня мобільність: Частина систем не має вебінтерфейсу або адаптації під мобільні пристрої, що обмежує доступ до даних поза межами офісу.

Напрями вдосконалення таких рішень включають:

* Розширення функціональних можливостей: Додавання можливостей для ведення історії обслуговування, нагадувань про ТО, генерації звітів, статистики використання автопарку тощо.
* Оптимізація інтерфейсу користувача: Розробка зручного, адаптивного та сучасного інтерфейсу відповідно до принципів UX/UI-дизайну.
* Інтеграція з іншими системами: Забезпечення взаємодії з існуючими сервісами установи (API, експортування даних, синхронізація).
* Забезпечення інформаційної безпеки: Реалізація багаторівневої авторизації, шифрування даних, ведення журналу дій користувачів.
* Мобільність і доступність: Реалізація вебінтерфейсу або мобільної версії для забезпечення повного доступу до системи з будь-якого пристрою.

Висновок до розділу 1

Проведений аналіз показує, що більшість наявних систем обліку транспортних засобів мають базовий функціонал, якого часто недостатньо для ефективного управління автопарком інституту. Основні недоліки включають обмежений функціонал, складність інтерфейсу, відсутність інтеграції з іншими системами та низький рівень захисту даних.

Для усунення вказаних недоліків доцільною є розробка сучасного RESTful додатку з використанням технології Node.js, який дозволить реалізувати масштабований, безпечний та зручний інструмент автоматизації обліку транспортних засобів із урахуванням усіх сучасних вимог до таких систем.

# ПРОЕКТУВАННЯ ТА ОСОБЛИВОСТІ РОЗРОБКИ

## Вимоги до програми

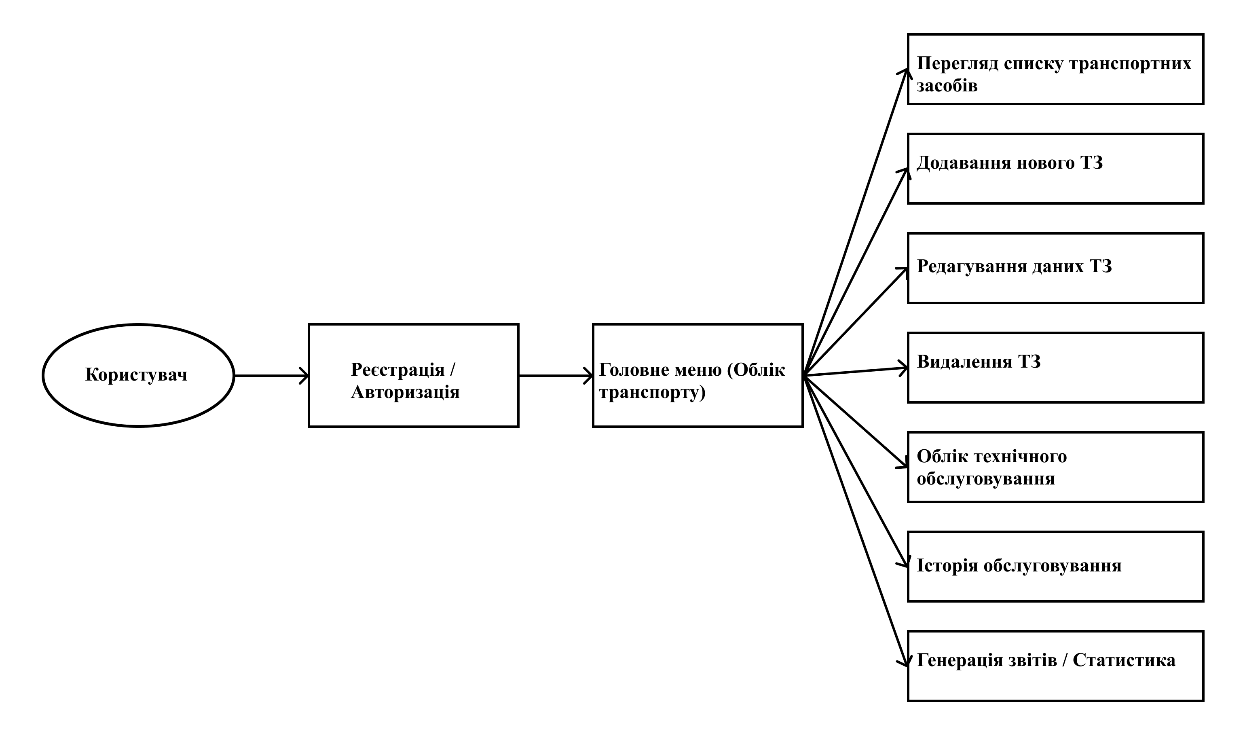


Рисунок 2.1. Діаграма варіантів використання розроблювального програмного модулю

## Схеми алгоритмів основних методів

У рамках проєкту було реалізовано кілька ключових REST-методів, кожен з яких виконує певну логіку на серверній стороні. Основні алгоритми представлено у вигляді блок-схем.

Метод 1: POST /vehicles — Додавання нового транспортного засобу

1. Перевірка даних: Перевірка валідності отриманих даних (марка, модель, номер, VIN, дата реєстрації тощо). Якщо дані неповні або некоректні — повертається повідомлення про помилку.
2. Створення запису: Якщо всі поля заповнені коректно — створюється новий об’єкт транспортного засобу.
3. Збереження в базу: За допомогою MongoDB об'єкт зберігається в колекції vehicles.
4. Результат: У разі успішного збереження — повертається відповідь з ID нового запису.

Метод 2: GET /vehicles — Отримання списку транспортних засобів

1. Підключення до бази: Встановлюється з’єднання з MongoDB.
2. Отримання даних: Дані вибираються із колекції vehicles, з можливістю фільтрації (наприклад, за статусом, типом тощо).
3. Формування відповіді: Список повертається у вигляді JSON-масиву.

Метод 3: PUT /vehicles/:id — Редагування інформації про ТЗ

1. Отримання ID: ID транспортного засобу береться з параметра запиту.
2. Перевірка прав доступу (за потреби): Авторизований користувач перевіряється на наявність прав.
3. Оновлення полів: Дані, передані в тілі запиту, використовуються для оновлення документа в базі.
4. Збереження змін: Після успішного оновлення — повертається повідомлення про успіх.

Метод 4: DELETE /vehicles/:id — Видалення ТЗ

1. Отримання ID: ID транспортного засобу витягується з URL.
2. Пошук у базі: Шукається об’єкт з таким ID.
3. Видалення: Якщо знайдено — запис видаляється з колекції.
4. Результат: Повертається підтвердження про видалення або повідомлення про помилку.

Метод 5: GET /reports/history — Отримання історії обслуговування

1. Пошук записів: Дані про технічне обслуговування вибираються з відповідної колекції.
2. Сортування: Записи сортуються за датою виконання.
3. Формування звіту: Дані готуються у форматі, придатному для перегляду або експорту.

## Опис технологій, що використовуються для вирішення завдання

### Node.js:

Node.js — це середовище виконання JavaScript на стороні сервера, що дозволяє створювати масштабовані й ефективні серверні додатки.

Переваги Node.js:

* Події та неблокуюче введення/виведення дозволяють ефективно працювати з багатьма запитами.
* Величезна кількість готових модулів через npm.
* Добре підходить для створення RESTful API.

### Express.js

Express — це фреймворк для Node.js, що спрощує створення серверів та маршрутизацію запитів.

Основні переваги:

* Гнучка система маршрутів.
* Мідлвари для обробки запитів, помилок, логування.
* Простота інтеграції з базами даних.

### MongoDB

MongoDB — це документо-орієнтована база даних NoSQL, що зберігає дані у форматі BSON (аналог JSON).

Чому MongoDB:

* Гнучка структура зберігання.
* Добре працює з JavaScript.
* Висока масштабованість.

### REST API

REST (Representational State Transfer) — архітектурний стиль для побудови вебсервісів.

Основні характеристики REST API:

* Стандартизовані HTTP-методи (GET, POST, PUT, DELETE).
* Безстанова взаємодія клієнт-сервер.
* Зручний для фронтенд-розробки.

Висновки до розділу 2

У цьому розділі описано ключові REST-методи, які реалізовано у рамках додатку для обліку транспортних засобів. Кожен метод було детально проаналізовано, представлено у вигляді блок-схеми та пояснено його логіку. Це дозволяє чітко розуміти, як працює взаємодія між клієнтом і сервером у межах розробленого додатку.

Для реалізації рішення використано сучасний технологічний стек: Node.js + Express для серверної частини, MongoDB як основна база даних, REST як архітектурний стиль. Це дозволяє побудувати гнучку, масштабовану й зручну у використанні систему для ведення обліку автопарку.

# ОСОБЛИВОСТІ ВИКОРИСТАННЯ ПРОГРАМНОГО ДОДАТКУ

## Реєстрація та авторизація

На рисунку 3.1 представлено два вебінтерфейси програми — реєстрації та авторизації користувача.

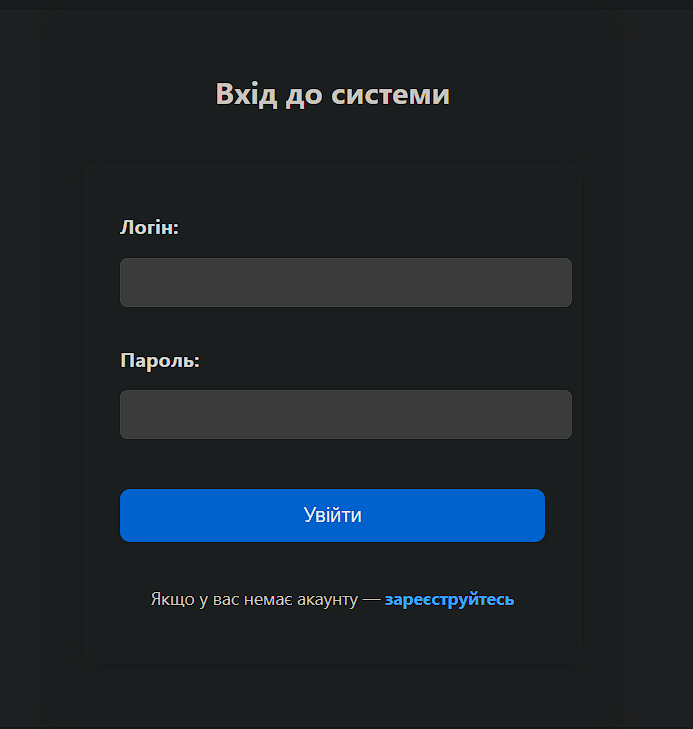
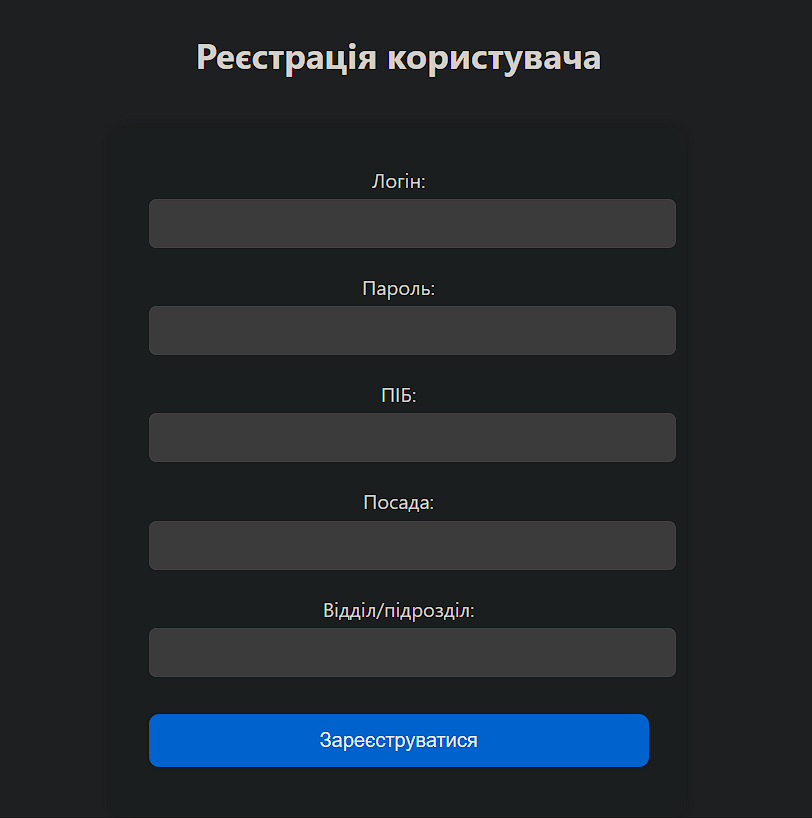
 

Рисунок 3.1.Вікна: авторизації та реєстрації

Після запуску додатку користувач потрапляє на сторінку реєстрації, де може створити обліковий запис шляхом введення унікального логіну та паролю (не менше 8 символів). Якщо введений логін вже існує, система повідомляє про це.

У випадку, якщо користувач уже зареєстрований, він може натиснути на посилання «Я вже маю акаунт», яке перенаправляє його до вікна авторизації. У цьому вікні потрібно ввести логін і пароль для входу в систему.  
Також передбачена можливість повернення до реєстрації — кнопка «Не маю акаунта».

## Головне меню: Облік транспорту

На рисунку 3.2 представлено головне вікно додатку — «Облік транспортних засобів».

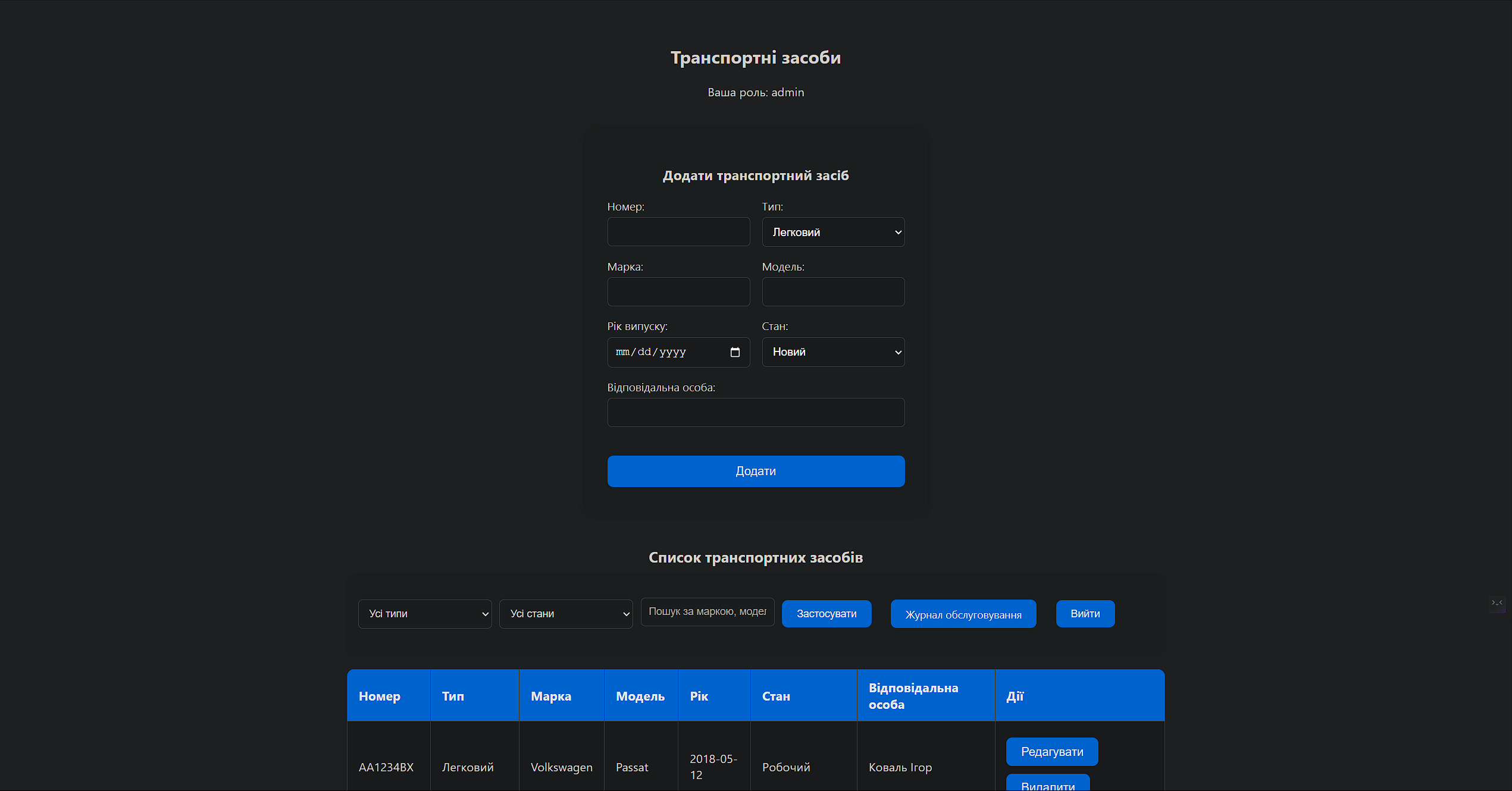


Рисунок 3.2. Головне меню обліку транспорту

Після авторизації користувач потрапляє до основного інтерфейсу, який дозволяє:

* Переглядати наявні транспортні засоби;
* Додавати новий запис про транспорт;
* Переглядати історію обслуговування;
* Редагувати інформацію про вже внесені ТЗ;
* Здійснювати пошук та фільтрацію за типом, номером або датою обслуговування.

Також передбачено:

* Кнопка “Вийти з акаунту” — повертає користувача на екран авторизації.

## Історія обслуговування транспортних засобів

На рисунку 3.3 зображено інтерфейс перегляду історії обслуговування транспортних засобів.

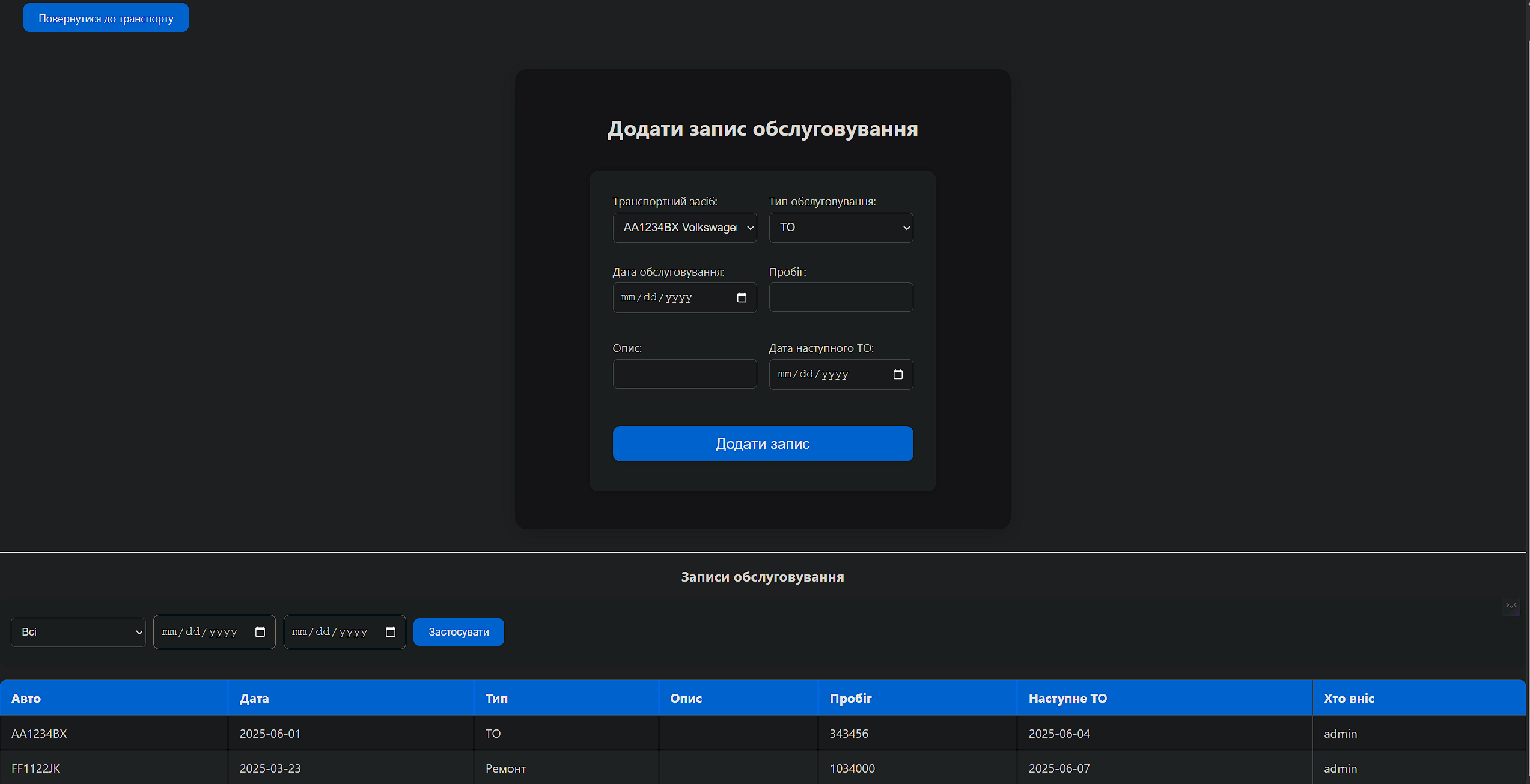


Рисунок 3.3. Сторінка історії обслуговування

Користувач може:

* Переглянути перелік усіх попередніх записів щодо ремонтів або технічного обслуговування;
* Використати функцію сортування за датою;
* Видалити всі записи з бази (лише адміністратор або уповноважений користувач);
* Повернутися до головного меню за допомогою кнопки “Повернутися до транспорту”.

## Додавання нового транспортного засобу

На рисунку 3.4 представлено вікно додавання нового транспортного засобу до бази даних.

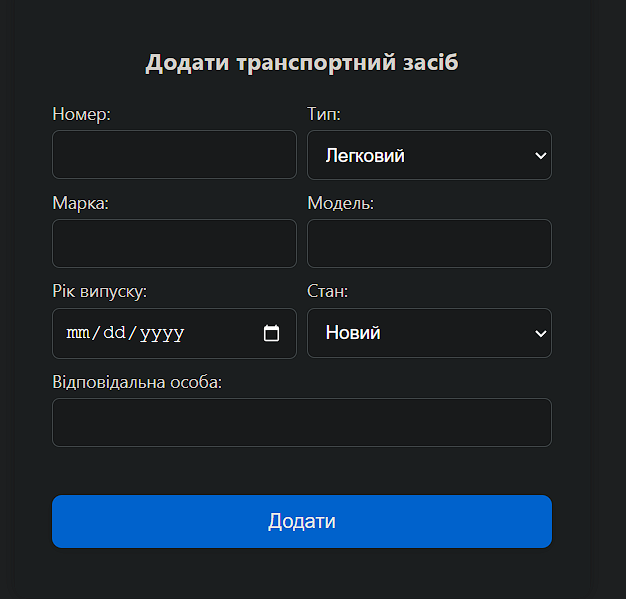


Рисунок 3.4. Вікно додавання ТЗ

Поля, які обов’язково мають бути заповнені:

* Тип транспортного засобу (автомобіль, автобус, спецтехніка тощо);
* Державний номер;
* Марка ТЗ;
* Модель ТЗ;
* Рік випуску;
* Стан ТЗ;
* Прізвище відповідального особи.

Після натискання кнопки «Додати»:

* Дані перевіряються на повноту та коректність;
* Якщо всі поля заповнено правильно — запис додається в базу даних MongoDB;
* Якщо виникає помилка (наприклад, не вказано номер або дата некоректна) — з’являється відповідне повідомлення про помилку.

Висновок до 3 розділу

Інтерфейс додатку спроектований з урахуванням зручності користування. Система покриває базові функціональні потреби: реєстрація, авторизація, додавання, редагування, пошук і облік технічного стану транспортних засобів.  
Кожен екран супроводжується підказками та системою обробки помилок, що полегшує роботу з додатком для користувачів будь-якого рівня підготовки.

# ВИСНОВОК

У ході виконання курсової роботи було проаналізовано існуючі програмні рішення у сфері автоматизації процесів замовлення послуг та обліку ресурсів. Виявлено, що більшість таких рішень або не адаптовані до потреб конкретних організацій, або не забезпечують достатнього рівня зручності, функціональності та масштабованості. Це зумовлює необхідність створення спеціалізованих рішень, які дозволять оптимізувати облік транспортних засобів та пов’язаних із цим операцій.

Метою розробки програмного модуля стало створення зручного, ефективного та масштабованого вебзастосунку для ведення обліку транспортних засобів інституту, що дозволяє здійснювати авторизацію та реєстрацію користувачів, додавання та перегляд даних про транспортні засоби, ведення історії обслуговування, а також фільтрацію, сортування та збереження інформації.

У процесі роботи було реалізовано:

* створення та обробку облікових записів користувачів (реєстрація, авторизація, перевірка унікальності логіна);
* внесення, перегляд та редагування даних про транспортні засоби;
* облік історії технічного обслуговування із можливістю сортування;
* збереження даних у базі даних MongoDB та у текстовому документі;
* побудову інтуїтивного інтерфейсу користувача з використанням HTML, CSS та JavaScript;
* розробку REST API для обміну даними між клієнтською частиною та сервером (Node.js + Express).

Використані технології, зокрема Node.js, Express.js, MongoDB, забезпечили надійність та продуктивність, а також відкрили можливості для подальшого розширення функціональності.

На основі створених методів та блок-схем функціонування основних модулів додатку було реалізовано повноцінний вебінтерфейс, який може бути використаний як готове рішення для автоматизації внутрішніх процесів в організаціях.

Напрями подальшого вдосконалення:

* Масштабування системи — розширення додатку на кілька підрозділів або організацій, підтримка обліку транспортних засобів у різних регіонах;
* Інтеграція з іншими сервісами — наприклад, GPS-моніторинг, зовнішні API для сервісного обслуговування;
* Формування звітів та статистики — генерація звітів за період, за типами транспорту, частотою обслуговування тощо;
* Покращення безпеки — використання JWT-токенів, шифрування паролів, додаткова аутентифікація;
* Мобільна адаптація — створення мобільної версії або PWA для зручного доступу з телефонів та планшетів;
* Інтерфейс адміністратора — управління користувачами, аналіз даних, контроль за базою ТЗ.

Таким чином, розроблений додаток відповідає поставленим завданням та забезпечує основу для створення повноцінної системи обліку транспортних засобів з можливістю масштабування, аналітики та подальшого розвитку.

# СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Сема О. А. Проєктування інформаційних систем : навчальний посібник. — Київ : Ліра-К, 2021. — 312 с.
2. Сафонова Н. О. Системи управління базами даних : навч. посібник. — Львів : Новий Світ–2000, 2019. — 256 с.
3. Бондар І. В. Web-програмування: HTML, CSS, JavaScript : навч. посібник. — Харків : ХНУ імені В. Н. Каразіна, 2020. — 198 с.
4. Гнатюк С. В., Яковенко В. А. Засоби розробки веб-застосунків : навчальний посібник. — Київ : КНЕУ, 2021. — 160 с.
5. Node.js Documentation [Електронний ресурс].
6. Express.js — офіційна документація [Електронний ресурс].
7. MongoDB Documentation [Електронний ресурс].
8. MDN Web Docs (HTML, CSS, JS) [Електронний ресурс].
9. Fielding R. T. Architectural Styles and the Design of Network-based Software Architectures [Електронний ресурс] // University of California, Irvine, 2000.
10. W3Schools — Web Technologies Tutorial [Електронний ресурс].

ДОДАТОК 1

З'єднання з базою даних mongodb

javascript

copyedit

const mongoose = require('mongoose');

const connectdb = async () => {

try {

await mongoose.connect(process.env.mongo\_uri, {

usenewurlparser: true,

useunifiedtopology: true

});

console.log('mongodb connected...');

} catch (err) {

console.error(err.message);

process.exit(1);

}

};

module.exports = connectdb;

ДОДАТОК 2

схема (модель) транспортного засобу

javascript

copyedit

const mongoose = require('mongoose');

const vehicleschema = new mongoose.schema({

brand: string,

model: string,

year: number,

vin: string,

licenseplate: string,

owner: string

}, {

timestamps: true

});

module.exports = mongoose.model('vehicle', vehicleschema);

ДОДАТОК 3

маршрут: створення нового транспортного засобу

javascript

copyedit

router.post('/', async (req, res) => {

try {

const vehicle = new vehicle(req.body);

await vehicle.save();

res.status(201).json(vehicle);

} catch (error) {

res.status(400).json({ message: error.message });

}

});

ДОДАТОК 4

маршрут: отримання списку всіх транспортних засобів

javascript

copyedit

router.get('/', async (req, res) => {

try {

const vehicles = await vehicle.find();

res.json(vehicles);

} catch (error) {

res.status(500).json({ message: error.message });

}

});

ДОДАТОК 5

запуск express-сервера

javascript

copyedit

const express = require('express');

const cors = require('cors');

const connectdb = require('./db');

require('dotenv').config();

const app = express();

const port = process.env.port || 5000;

app.use(cors());

app.use(express.json());

connectdb();

app.use('/api/vehicles', require('./routes/vehicles'));

app.listen(port, () => {

console.log(`server is running on port: ${port}`);

});