Міністерство освіти і науки України
Луцький національний технічний університет
Факультет комп'ютерних та інформаційних технологій
Кафедра інженерії програмного забезпечення

КВАЛІФІКАЦІЙНА РОБОТА

ЗА СТУПЕНЕМ ВИЩОЇ ОСВІТИ «БАКАЛАВР»

РОЗРОБКА ВЕБДОДАТКУ ПП «ІНТЕРТРЕЙД-ЛУЦЬК» ДЛЯ ПРОФЕСІЙНОГО ПІДБОРУ ТА ПРИГОНУ АВТО З США З ВИКОРИСТАННЯМ VUE.JS, PHP ТА MYSQL

DEVELOPMENT OF A WEB APPLICATION FOR "INTERTRADE-LUTSK" FOR PROFESSIONAL SELECTION AND IMPORT OF CARS FROM THE USA USING VUE.JS, PHP AND MYSQL

спеціальність 121 «Інженерія програмного забезпечення»
освітня програма «Інженерія програмного забезпечення»

Виконав: здобувач вищої освіти
групи ІПЗ-41
Стичук Петро Вадимович

Керівник:
Андрущак Ігор Євгенович

Кваліфікаційну роботу
допущено до захисту
«10» 06 2025 р.
Гарант освітньої програми:
к.т.н., доцент
Ліщина Наталія Миколаївна

Луцьк – 2025 року

ЛУЦЬКИЙ НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ

Факультет *комп'ютерних та інформаційних технологій*
Кафедра *інженерії програмного забезпечення*
Ступінь вищої освіти *бакалавр*
Галузь знань: *12 «Інформаційні технології»*
Спеціальність: *121 «Інженерія програмного забезпечення»*
Освітня програма: «*Інженерія програмного забезпечення»*

ЗАТВЕРДЖУЮ
Завідувач кафедри
_____
«*20*» *12* 20*24*р.

ЗАВДАННЯ

НА КВАЛІФІКАЦІЙНУ РОБОТУ ЗДОБУВАЧУ ВИЩОЇ ОСВІТИ

Стичуку Петру Вадимовичу
(прізвище, ім'я, по батькові)

1. Тема кваліфікаційної роботи: *«Розробка вебдодатку ПП "ІНТЕРТРЕЙД-ЛУЦЬК" для професійного підбору та пригону авто з США з використанням Vue.js, PHP та MySQL».*
Керівник роботи: *д.т.н., професор Андрущак І. Є.*
затверджені наказом закладу вищої освіти від *«19» грудня 2024 р.* № *474/01-02.*
2. Строк подання здобувачем вищої освіти кваліфікаційної роботи бакалавра *«10» червня 2025 р.*
3. Вихідні дані до роботи: *Vue.js + Vite (Node.js, npm), PHP 8 (Composer), MySQL (InnoDB), MAMP (Apache + PHP 8 + MySQL), Visual Studio Code, методичні вказівки до виконання кваліфікаційної роботи бакалавра.*
4. Зміст розрахунково-пояснювальної записки (перелік питань, що потрібно розробити): *аналіз ринку імпорту авто зі США та конкурентних сервісів; функціональні та нефункціональні вимоги до вебдодатку; проєктування архітектури системи й моделі даних MySQL; опис оптимального технологічного стеку; опис реалізації фронтенд, бекенд і REST-API для обміну даними; опис модульного, інтеграційного й end-to-end тестування; опис заходів безпеки (автентифікація, захист від ін'єкцій і XSS).*
5. Перелік графічного матеріалу: *12 рисунків, 1 додаток.*

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис	
		завдання видав	завдання прийняв
Аналіз предметної області	*Андрущак І. Є.*		
Специфікація вимог до розробленої системи	*Андрущак І. Є.*		
Розробка об'єкта проектування	*Андрущак І. Є.*		
Нормоконтроль	*Вознюк А. В.*		
Гарант ОП	*Ліщина Н. М.*		
Показник запозичень тексту	1,48 %		
Академічна доброчесність	*Андрущак І. Є.*		

7. Дата видачі завдання *«20» грудня 2024 р.*

№	Назва етапів кваліфікаційної роботи бакалавра	Строк виконання етапів роботи	Примітка
1	Огляд літературних джерел по темі кваліфікаційної роботи бакалавра	до 04.02.2025 р.	ВИКОН
2	Аналіз проблеми розробки та впровадження об'єкту проектування	до 01.03.2025 р.	ВИКОН
3	Обгрунтування вибору шляхів, технологій і засобів вирішення поставленого завдання	до 15.03.2025 р.	ВИКОН
4	Розробка функціонально-структурної схеми роботи об'єкта проектування та проектування бази даних	до 29.03.2025 р.	ВИКОН
5	Практична реалізація об'єкта проектування та розробка бази даних	до 26.04.2025 р.	ВИКОН
6	Тестування та налагодження об'єкта проектування	до 03.05.2025 р.	ВИКОН
7	Здача чистового варіанту кваліфікаційної роботи бакалавра на кафедру	до 10.06.2025 р.	ВИКОН

Здобувач вищої освіти _____ Петро СТИЧУК

Керівник кваліфікаційної роботи _____ Ігор АНДРУЩАК

АНОТАЦІЯ

Стичук П. В. Розробка вебдодатку ПП «ІНТЕРТРЕЙД-ЛУЦЬК» для професійного підбору та пригону авто з США з використанням Vue.js, PHP та MySQL. Рукопис. Кваліфікаційна робота бакалавра ОП «Інженерія програмного забезпечення» спеціальності «Інженерія програмного забезпечення». Луцький національний технічний університет. Луцьк, 2025.

Кваліфікаційна робота бакалавра складається зі вступу, трьох розділів, висновків, списку використаних джерел та додатків.

У першому розділі здійснено аналіз ринку імпорту вживаних автомобілів із США в Україні та конкурентних сервісів, обґрунтовано доцільність розробки та сформульовано завдання кваліфікаційної роботи. У другому розділі проведено моделювання системи, описано функціональні та нефункціональні вимоги, спроектовано архітектуру MVC та реляційну модель даних у MySQL, а також обґрунтовано вибір технологічного стеку. У третьому розділі реалізовано практичну частину: налаштовано середовище розробки, створено фронтенд на Vue.js, бекенд на PHP із REST-API, розроблено фізичну схему бази даних, а також проведено тестування, налагодження й забезпечено захист системи. У висновках підкреслено, що виконані задачі забезпечили створення безпечного, продуктивного та масштабованого вебдодатку, що відповідає вимогам ПП «ІНТЕРТРЕЙД-ЛУЦЬК».

Ключові слова: імпорт авто із США, вебдодаток, Vue.js, PHP, MySQL, REST-API, MVC.

ABSTRACT

Stychuk P. Development of a Web Application for "INTERTRADE-LUTSK" for Professional Selection and Import of Cars from the USA Using Vue.js, PHP and MySQL. Manuscript. Qualification work of the bachelor's program "Software engineering" in the specialty "Software engineering". Lutsk National Technical University. Lutsk, 2025.

The bachelor's qualification thesis consists of an introduction, three chapters, conclusions, a list of references, and appendices.

In Chapter 1, the market for importing used cars from the USA to Ukraine and existing competitor services was analyzed, the feasibility of development was justified, and the objectives of the qualification work were formulated. In Chapter 2, system modeling was conducted: functional and non-functional requirements were defined, the MVC architecture and a relational MySQL data model were designed, and the choice of the technology stack was substantiated. In Chapter 3, the practical implementation was carried out: the development environment was configured, the frontend was created using Vue.js, the backend was implemented in PHP with a RESTful API, the physical database schema was developed, and testing, debugging, and security measures were ensured. The conclusions emphasize that these tasks have resulted in a secure, efficient, and scalable web application that meets the requirements of Intertrade-Lutsk LLC.

Keywords: import of cars from USA, web application, Vue.js, PHP, MySQL, REST API, MVC.

ЗМІСТ

ВСТУП

Актуальність теми. Сучасний автомобільний ринок України зазнає стрімких змін під впливом глобалізації, коливань курсу національної валюти та зростаючих запитів споживачів на якісні й недорогі транспортні засоби. Одним із найпопулярніших напрямів, що набув значного поширення в останні роки, є імпорт уживаних автомобілів із США. Висока конкуренція, більший вибір моделей, інноваційні комплектації та переважно гарний технічний стан американських авто внаслідок жорстких норм безпеки й регулярного обслуговування роблять їх привабливими для українських покупців. Особливо актуальним цей напрям став у період економічної нестабільності, коли можливість придбати машину з США за вигіднішою ціною порівняно з аналогічними пропозиціями на внутрішньому ринку допомагає значно знизити фінансове навантаження на сімейний бюджет.

Проте процес придбання автомобіля в Сполучених Штатах пов’язаний із низкою складнощів, що перешкоджають багатьом потенційним клієнтам. Насамперед це – підбір надійного постачальника, аналіз стану автомобіля за віддаленими фото- та відеоматеріалами, розрахунок повної вартості доставки, митного оформлення й сертифікації в Україні.

У такому контексті потреба в зручному вебдодатку, який би об’єднав функціонал пошуку, порівняння та замовлення авто зі США, стала вкрай нагальною. Оптимізована платформа повинна забезпечувати інтуїтивно зрозумілий інтерфейс для українського користувача, автоматизовані алгоритми фільтрації за ключовими параметрами, актуальні дані про вартість доставки та митні платежі.

Метою кваліфікаційної роботи бакалавра є розробка вебдодатку ПП «ІНТЕРТРЕЙД-ЛУЦЬК» для професійного підбору та пригону автомобілів із США, який базується на сучасних вебтехнологіях Vue.js, PHP та MySQL.

Об’єктом кваліфікаційної роботи бакалавра є вебдодаток ПП «ІНТЕРТРЕЙД-ЛУЦЬК» – інформаційно-керуюча система для автоматизації

процесів пошуку, оцінки, замовлення та доставки вживаних автомобілів із США.

Предметом кваліфікаційної роботи бакалавра є практична реалізація та інтеграція фронтенд-модулів на базі Vue.js, бекенд-логіки на PHP і реляційної бази даних MySQL для забезпечення функціональності пошуку, фільтрації, калькуляції вартості доставки й митного оформлення автомобілів.

Для досягнення поставленої мети були сформульовані такі завдання:

– проаналізувати ринок імпорту авто зі США та конкурентні сервіси;

– визначити функціональні та нефункціональні вимоги до вебдодатку;

– спроєктувати архітектуру системи й модель даних MySQL;

– обрати оптимальний технологічний стек;

– реалізувати фронтенд, бекенд і REST-API для обміну даними;

– провести модульне, інтеграційне й end-to-end тестування;

– забезпечити безпеку (автентифікація, захист від ін’єкцій і XSS).

Отже, розробка подібного вебдодатку ПП «ІНТЕРТРЕЙД-ЛУЦЬК» не лише відповідає актуальним потребам українського ринку імпорту автомобілів зі США, але й сприятиме підвищенню прозорості, ефективності та зручності всього процесу для кінцевого споживача.

РОЗДІЛ 1
АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ І ПОСТАНОВКА ЗАВДАНЬ НА КВАЛІФІКАЦІЙНУ РОБОТУ

1.1 Аналіз сучасного стану проблеми

У першому кварталі 2024 року імпорт автомобілів в Україну виріс на 67 % порівняно з аналогічним періодом минулого року – до понад 86 000 одиниць загальною вартістю $1,1 млрд. Зростання обумовлене як відновленням попиту після спаду 2023 року, так і збільшенням частки електромобілів у загальному потоці імпорту [1].

За перші сім місяців 2024 року в Україну було ввезено понад 228 000 авто – це на 14,2 % більше, ніж за аналогічний період 2023-го (рис. 1.1).

Рисунок 1.1 – Кількість ввезених транспортних засобів [2]

При цьому 71 % імпортованих транспортних засобів були вживаними, а середній вік авто склав 9 років. Попит на електромобілі зріс на 83 %, що свідчить про зростаючу екологічну свідомість покупців та доступність відповідних моделей із США [2].

Найбільшу кількість нововвезених автомобілів зареєстровано в Києві – 13,7 % (рис. 1.2). До трійки лідерів також увійшли Львівська область (10,2 %) та Одеська область (7,4 %).

Рисунок 1.2 – Статистика реєстрації ввезених автомобілів у регіонах України [2]

Натомість у 2023 році імпорт вживаних авто скоротився майже наполовину – до близько 377 000 одиниць, найнижчого показника за три роки. Основними причинами падіння були зростання вартості доставки, невизначеність із митним законодавством та слабкий курс гривні [3].

За даними асоціації «Укравтопром», у 2024 році українці придбали близько 41 700 уживаних легковиків із США – це на 46 % більше, ніж у 2023

році. Автомобілі з американського ринку становили 19 % від загальної кількості вживаних машин, вперше зареєстрованих у нашій країні, при цьому середній вік «американців» становив 5,8 року [4].

Популярність імпорту з США пояснюється в першу чергу значною економією: близько 45 % придбаних на американських аукціонах авто коштували менше \$7 000 за лот, а окремі моделі, наприклад Tesla Model 3 або BMW X5, виявляються на 20-30 % дешевшими за аналогічні пропозиції в Україні [5].

Крім того, американські автомобілі приваблюють ширшим вибором комплектацій та вищими стандартами безпеки: популярність електромобілів (Tesla, Chevrolet Bolt) і позашляховиків з місткими двигунами підтверджує готовність українців експериментувати з новими технологіями та моделями, недоступними на внутрішньому ринку (рис. 1.3).

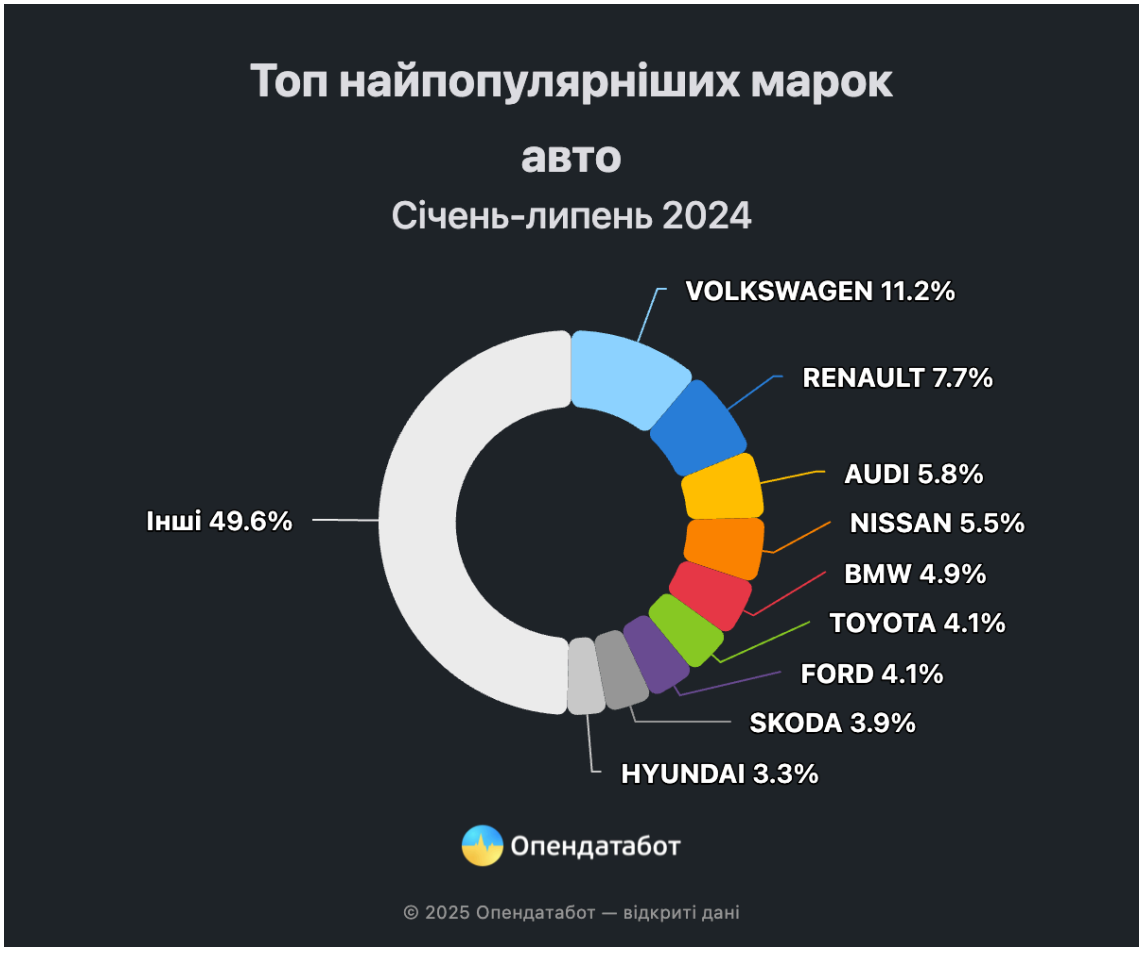

Рисунок 1.3 – Топ найпопулярніших марок авто [2]

З огляду на вищезазначене, попри явні економічні та технологічні переваги, процес купівлі та ввезення авто із США залишається фрагментованим і трудомістким. Це обумовлює необхідність створення єдиного вебдодатку, який автоматизує підбір, розрахунок повної вартості, оформлення документів та моніторинг логістики на всіх етапах імпорту.

Вебдодатки сьогодні є найзручнішим інструментом для організації складних бізнес-процесів з кількох ключових причин:

1) доступність із будь-якого пристрою та в будь-який час. Користувачеві не потрібно завантажувати та встановлювати спеціалізоване програмне забезпечення – достатньо будь-якого сучасного браузера на ПК, планшеті чи смартфоні. Це забезпечує оперативний доступ до інформації про лоти, статус доставки чи зміни митного законодавства незалежно від місцезнаходження та операційної системи;

2) централізація даних і єдина точка істини. Усі дані про автомобілі, калькуляції вартості, документи та повідомлення зберігаються в одній базі. Адміністратор чи менеджер можуть вносити зміни в один раз – і вони відразу ж стають доступними всім користувачам. Це усуває ризик розбіжностей між електронними таблицями, нотатками чи різними інтерфейсами сторонніх сервісів;

3) реальний час і інтеграція з зовнішніми API. Через вебдодаток можна автоматично запитувати актуальні курси валют, ставки акцизу, тарифи на доставку від логістичних компаній чи чинні митні збори. Інтеграція з API аукціонів (eBay Motors, Copart, IAAI) дозволяє миттєво імпортувати лоти без ручного копіювання, а внутрішні веб-сервіси сповіщають клієнта про зміну статусу оформлення прямо в інтерфейсі або на email/Telegram;

4) масштабованість і гнучкість розгортання. Завдяки архітектурі клієнт-сервер та поділу на мікросервіси чи модулі, систему легко розширювати новими функціями: чат-ботом для консультацій, аналітичними панелями, персональними кабінетами з історією замовлень тощо. При збільшенні

навантаження (більше користувачів, нові регіони доставки) можна масштабувати серверну частину без перенаписування фронтенду;

5) адаптивний інтерфейс і покращений UX. Використання компонентних фреймворків (Vue.js) дозволяє створювати інтерактивні елементи: динамічні фільтри за кількома параметрами, миттєве сортування, картки з фотографіями та історією техобслуговування. У поєднанні з адаптивною верстkою це забезпечує комфортну роботу й на мобільних екранах, де багато користувачів здійснюють пошук «на ходу»;

6) безпечна робота з конфіденційними даними. Вебдодаток із правильно налаштованим бекендом (PHP) та реляційною базою даних (MySQL) дозволяє централізовано контролювати доступ, шифрувати з'єднання (HTTPS), впроваджувати рольову модель доступу та захищати від типових веб-атак (SQL-ін'єкції, XSS, CSRF).

З урахуванням наведеного, створення вебдодатку для професійного підбору та пригону авто з США є обґрунтованим не лише з огляду на зручність користувача, але й з точки зору ефективності бізнес-процесів, безпеки та масштабованості рішення. Це суттєво підвищить швидкість операцій, мінімізує людський фактор і дозволить ПП «ІНТЕРТРЕЙД-ЛУЦЬК» ефективніше конкурувати на ринку імпорту автомобілів в Луцьку.

На локальному ринку імпорту вживаних автомобілів із США можна виокремити щонайменше три активні компанії, що позиціонують себе як «під ключ» сервіси з підбору, доставки й розмитнення: A_Dream, CargoLine та Lube Avto. Нижче наведено їх короткий опис, основні переваги та виявлені недоліки.

Компанія A_Dream спеціалізується на постачанні вживаних авто з США до міста Луцьк під замовлення. На своєму сайті вони обіцяють економію до 50 % порівняно з українським ринком, швидку доставку та 2-річну гарантію на поставлений транспортний засіб [6].

Аналіз сайту A_Dream показує, що компанія позиціонує себе як повноцінний «під ключ» сервіс з підбору, доставки та ремонту автомобілів із США з гарантією на ключові агрегати терміном до двох років. На головній

сторінці розміщено онлайн-калькулятор доставки, який дозволяє попередньо оцінити всі витрати з точністю до 99 %, а також детально описано шість етапів процесу: підбір лоту, розрахунок вартості, підготовка до відправки, доставка, сертифікація та передача авто клієнту (рис. 1.4).

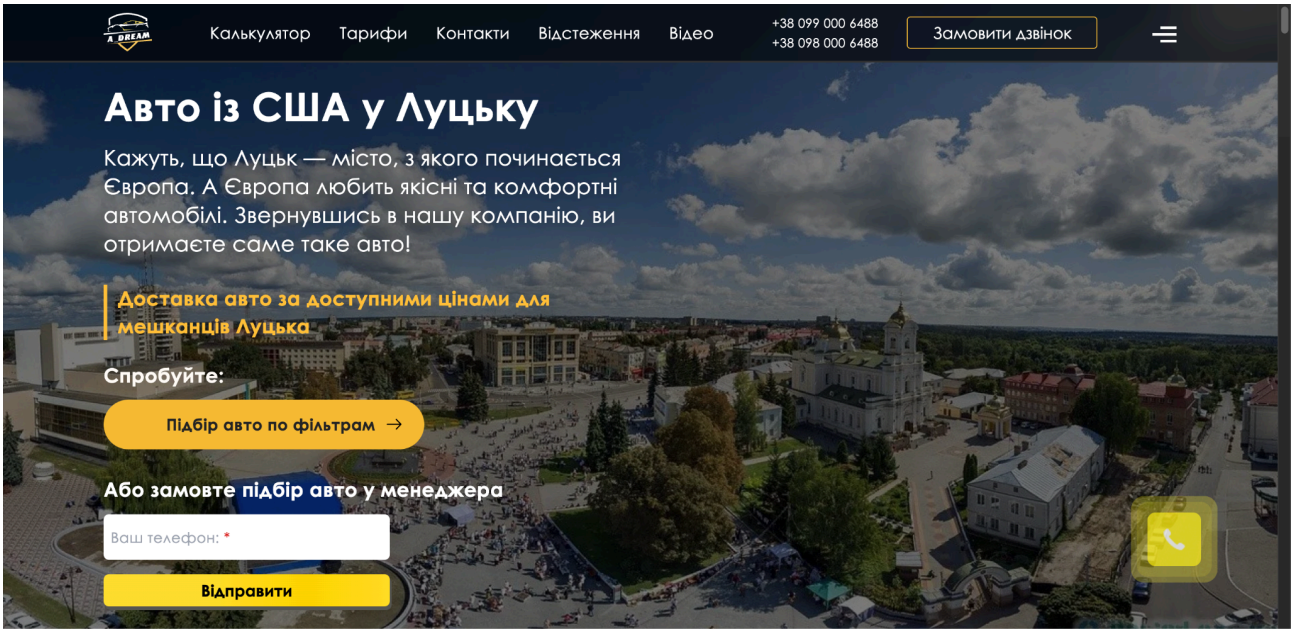

Рисунок 1.4 – Домашня сторінка компанії A_Dream [6]

Серед переваг A_Dream варто відзначити прозору послідовність усіх процедур та зручний калькулятор, що дозволяє відразу отримати приблизну повну вартість імпорту. Наявність локальних представництв і власних фахівців у США підвищує довіру клієнтів і гарантує якісну перевірку авто перед покупкою. Також компанія реалізувала SEO-оптимізацію під різні регіони, зокрема створивши окремі сторінки «Авто із США у Луцьку» та інших міст, що сприяє кращій видимості в пошукових системах.

Водночас сайт має низку недоліків. По-перше, відсутня можливість реєстрації особистого кабінету, де клієнт міг би зберігати історію запитів і відстежувати статус поточних замовлень без необхідності звертатися до менеджера. По-друге, каталог наявних авто містить лише базові фільтри (двигун, пробіг, ціна) і не дозволяє гнучкого сортування за роком випуску, типом кузова чи іншим параметрам. По-третє, хоча на сайті заявлена гарантія

два роки, відсутні детальні умови її надання – невідомо, які вузли вона покриває та які сервісні центри обслуговують клієнтів. Крім того, мовне наповнення сторінок є нерівномірним (російська, українська та англійська змішано без чіткої структури), що може ускладнювати сприйняття інформації.

Для вебдодатку ПП «ІНТЕРТРЕЙД-ЛУЦЬК» варто врахувати цю експертизу та реалізувати такі покращення: забезпечити особистий кабінет із історією запитів і повідомленнями про етапи оформлення, розширити функціонал каталогу інтерактивними фільтрами й сортуванням за ключовими параметрами, чітко прописати умови гарантійного обслуговування в окремому розділі, а також уніфікувати мовне наповнення сайту українською для підвищення довіри аудиторії й SEO-ефективності. Такий підхід дозволить уникнути недоліків конкурентів і запропонувати користувачам максимально прозорий та зручний сервіс.

CargoLine – логістична компанія з представництвом у Луцьку, що окрім доставки вантажів пропонує послугу імпорту авто з США «під ключ». На сайті вони позиціонують себе як гаранта «юридичної чистоти» та «реального пробігу» (рис. 1.5) [7].

Рисунок 1.5 – Домашня сторінка компанії CargoLine [7]

Аналіз сайту CargoLine показує, що компанія позиціонується як повноцінний комплексний сервіс, що охоплює підбір автомобіля на американських аукціонах, організацію доставки до України та митно-брокерське оформлення «під ключ». Інтерактивний каталог дозволяє відсортувати лоти за брендом, роком випуску та ціною, а вбудований калькулятор митних платежів миттєво розраховує розмір зборів, що робить ціноутворення максимально прозорим. На кожній сторінці чітко наведені контактні дані менеджерів і перелік доступних способів оплати – від карток Visa та Mastercard до Приват24 і Liqpay – що підвищує довіру користувачів і дозволяє одразу зв’язатися з фахівцем для уточнення деталей.

Водночас у CargoLine немає особистого кабінету, де клієнт міг би зберігати обрані лоти й переглядати історію запитів: усі дії виконуються через єдину форму зворотного зв’язку. Фільтрація на сайті обмежується вибором бренду та базовим сортуванням, тоді як такі важливі параметри, як пробіг, тип кузова або діапазон цін, доводиться уточнювати окремо. Окремої уваги заслуговують гарантійні зобов’язання компанії – хоча на сайті згадується «100 % виконання зобов’язань», жодних деталей про терміни або перелік покриття немає, і клієнт змушений звертатися до менеджера за роз’ясненнями. Крім того, надмірна кількість повторів у меню й дублювання назв розділів (наприклад, окремі пункти «Доставка», «Розмитнення» та «Калькулятор») створюють інформаційний шум і можуть збивати з пантелику, особливо на мобільних пристроях, де меню не завжди автоматично трансформується в компактний «гамбургер».

Для майбутнього вебдодатку ПП «ІНТЕРТРЕЙД-ЛУЦЬК» доцільно реалізувати власний особистий кабінет із збереженням історії запитів та сповіщеннями про статус виконання замовлення, розширити можливості фільтрації одночасно за кількома параметрами (рік, пробіг, ціна, тип кузова), деталізувати гарантійні умови прямо в каталозі та оптимізувати навігацію, об’єднавши повторювані розділи. Це дозволить зробити сервіс більш

інтуїтивним, знизити кількість ручних запитів до менеджерів і підвищити загальну прозорість процесу імпорту авто з США.

Lube Avto пропонує «повний спектр» послуг із імпорту авто з США: від пошуку та технічної оцінки до документального оформлення й доставки в Луцьк [8].

Сторінка «Авто в наявності з США у Луцьку» на сайті Lube Avto демонструє один із найширших у регіоні асортиментів – на момент перегляду доступно 864 лоти з докладними характеристиками (модель, рік випуску, пробіг, тип двигуна, коробка передач, привід) та можливістю сортування за новизною. Інтерфейс фільтрів охоплює понад десять параметрів, включаючи тип транспорту, регіон, бренд і модель, рік виготовлення, діапазон цін, колір і пошкодження, що дозволяє користувачам тонко налаштовувати пошук без необхідності переходити на інші сторінки (рис. 1.6). Наявність кнопок «Увійти» й «Калькулятор» свідчить про передбачені особистий кабінет і окремий інструмент для оцінки митних платежів і доставки, а зовнішній сервіс відстеження вантажу (shipping.lubeavto.com.ua) забезпечує прозорість логістики в режимі реального часу.

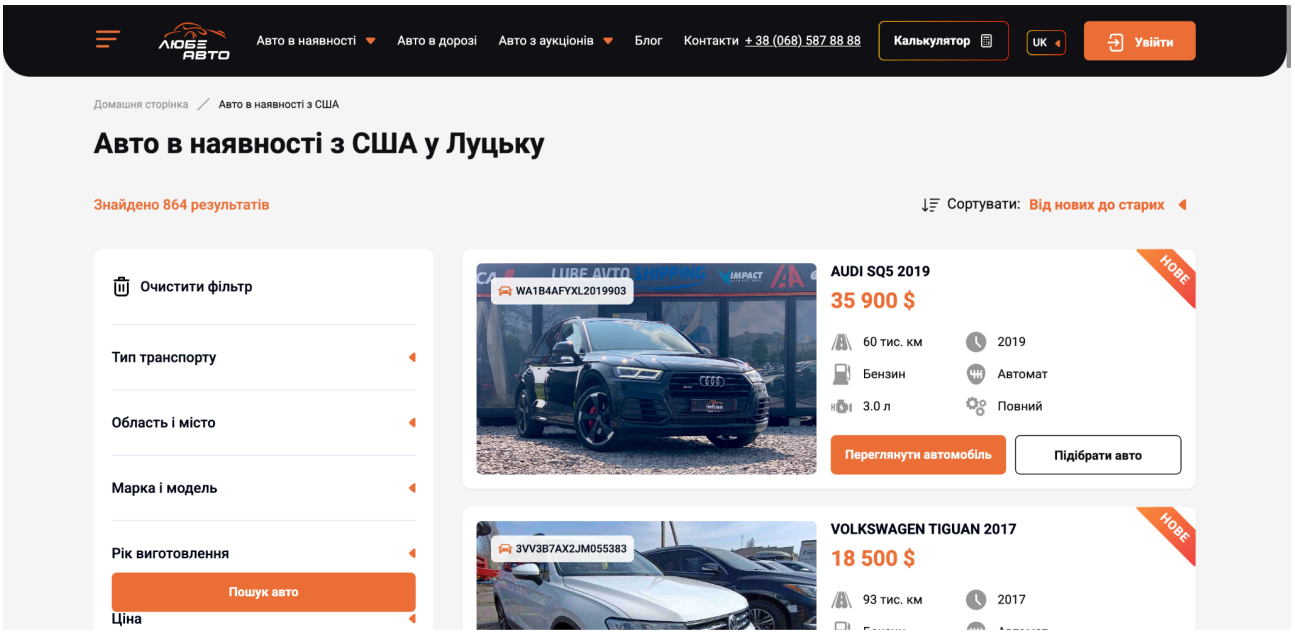

Рисунок 1.6 – Домашня сторінка компанії Lube Avto [8]

Водночас інтерфейс перевантажений елементами: надмірна кількість фільтрів та брендових списків створює інформаційний шум і може уповільнювати пошук на мобільних пристроях, де панель фільтрів займає значну частину екрану. Незважаючи на видимі посилання на гарантійні умови, детальний опис покриття та термінів гарантії винесено в окремий розділ сайту, що змушує користувача здійснювати додаткові кліки. Окрім того, вартість доставки й митних платежів не інтегрована безпосередньо в картки лотів – для розрахунку потрібно переходити до калькулятора, що руйнує єдину систему прийняття рішення.

Для майбутнього вебдодатку ПП «ІНТЕРТРЕЙД-ЛУЦЬК» доцільно зменшити інформаційний шум, об’єднавши фільтри в компактні розділи з можливістю згортання, інтегрувати попередній розрахунок всіх витрат безпосередньо в картки лотів і вивести основні умови гарантії поруч із ключовими характеристиками авто. Реалізація персонального кабінету з історією запитів і сповіщеннями про статус виконання замовлення, а також єдина панель відстеження логістики зроблять сервіс ще зручнішим і прозорішим для користувача.

Можна констатувати, що стрімке зростання імпорту вживаних автомобілів із США в Україні пояснюється значною економією, ширшим вибором моделей та вищими стандартами безпеки, проте залишається обтяженим багатоступеневим оформленням і невизначеністю витрат. Наявні локальні сервіси – A_Dream, CargoLine та Lube Avto – пропонують «під ключ» рішення з калькуляторами та фільтрами, але всі вони мають суттєві прогалини: відсутність особистого кабінету й гнучкої фільтрації, фрагментарність гарантійних умов та інформаційний шум у навігації. Отже, створення єдиного вебдодатку для ПП «ІНТЕРТРЕЙД-ЛУЦЬК», що поєднає прозорі алгоритми розрахунку вартості, інтуїтивний інтерфейс з адаптивними фільтрами, персональний кабінет користувача та інтегроване відстеження логістики, є вкрай доцільним і дозволить закрити наявні ніші на ринку.

1.2 Постановка завдання на кваліфікаційну роботу бакалавра

Метою кваліфікаційної роботи бакалавра є розробка вебдодатку ПП «ІНТЕРТРЕЙД-ЛУЦЬК» для професійного підбору та пригону автомобілів із США, який базується на сучасних вебтехнологіях Vue.js, PHP та MySQL.

Для досягнення поставленої мети були сформульовані такі завдання:

– проаналізувати ринок імпорту авто зі США та конкурентні сервіси;

– визначити функціональні та нефункціональні вимоги до вебдодатку;

– спроєктувати архітектуру системи й модель даних MySQL;

– обрати оптимальний технологічний стек;

– реалізувати фронтенд, бекенд і REST-API для обміну даними;

– провести модульне, інтеграційне й end-to-end тестування;

– забезпечити безпеку (автентифікація, захист від ін’єкцій і XSS).

РОЗДІЛ 2
СПЕЦИФІКАЦІЯ ВИМОГ ДО РОЗРОБЛЮВАНОГО ВЕБДОДАТКУ ПП «ІНТЕРТРЕЙД-ЛУЦЬК»

2.1 Аналіз, визначення вимог до розроблюваного програмного забезпечення та проектування програмного забезпечення

Визначення функціональних та нефункціональних вимог є одним із найважливіших етапів розробки програмного забезпечення. Саме на цьому кроці формулюється чіткий перелік можливостей, які система має реалізувати, а також критерії якості її роботи. Правильно сформульовані вимоги дозволяють уникнути непорозумінь між замовником і командою розробки, забезпечити однозначність технічного завдання, спростити подальше тестування та підтримку проєкту.

Функціональні вимоги:

– клієнтська частина має надавати можливість фільтрації автомобілів за ключовими атрибутами: марка, ціна та рік випуску;

– інтерфейс замовлення авто повинен містити форму, де користувач вводить своє ім’я, номер телефону, поштову адресу, марку, модель автомобіля та додатковий коментар;

– має бути розділ «Новини», у якому автоматично відображаються актуальні повідомлення про поставки, внутрішні події компанії та зміни в роботі сервісу;

– сторінка «Контакти» повинна містити усі необхідні реквізити: адреси офісів, телефони та електронну пошту;

– адміністративна панель має дозволяти створення та редагування карток автомобілів із такими полями: назва, модель, рік випуску, пробіг, тип двигуна, коробка передач, привід, колір, ціна, опис та кілька фотографій;

– адмін може додавати, редагувати та видаляти новини, вказуючи заголовок, повний текст та необов’язкові зображення;

– система повинна зберігати всі заявки клієнтів (як на покупку готових авто, так і на індивідуальний підбір), забезпечувати перегляд їхньої історії у адмін-панелі та зміну статусу обробки.

Нефункціональні вимоги:

– час завантаження сторінки пошуку не повинен перевищувати 2 секунд при навантаженні до 100 одночасних користувачів;

– архітектура має дозволяти горизонтальне розширення (додавання нових серверів) без зміни клієнтської частини;

– захист від SQL-ін’єкцій, XSS та CSRF-атак; шифрування з’єднання через HTTPS; розмежування прав доступу між адміністраторами та гостями;

– адаптивний дизайн для коректного відображення на пристроях із різними розмірами екранів (мобільні, планшети, десктопи);

– автоматичне резервне копіювання бази даних з інтервалом не більше ніж 24 години; моніторинг стану сервера та оповіщення про збої;

– інтуїтивна навігація, чіткі підказки в формах, єдина стилістика інтерфейсу; підтримка однієї мовної локалі – української;

– коректна робота в сучасних версіях основних браузерів (Chrome, Firefox, Edge, Safari);

– структурований модульний код на фронтенді (Vue.js) та бекенді (PHP) із коментарями англійською мовою; чітка документація API;

– семантична розмітка, метатеги, карта сайту і robots.txt для покращення індексації в пошукових системах.

Цей перелік вимог закладає основу для подальшого проєктування архітектури, моделювання UML-діаграм і формалізованої специфікації, що забезпечать прозорість процесу розробки та високу якість кінцевого продукту.

Для системного проєктування веб-додатку ПП «ІНТЕРТРЕЙД-ЛУЦЬК» було обрано нотацію UML (Unified Modeling Language), оскільки вона є загальноприйнятим стандартом для формалізованого опису як структурних, так і поведінкових аспектів ПЗ. UML дозволяє одразу перейти від високорівневого бачення (сценаріїв використання) до детальної специфікації сутностей і їхніх

взаємозв’язків, що гарантує єдине розуміння вимог між замовником, аналітиком і розробником [9].

Архітектурно-дизайнерським рішенням обрано багаторівневу клієнт-серверну модель у стилі MVC (Model-View-Controller):

– Model – описує дані (автомобілі, заявки, новини) і зберігається в реляційній БД MySQL;

– View – це фронтенд-частина на Vue.js, яка відповідає за відображення списків авто, форм замовлення, новин тощо;

– Controller – бекенд-логіка на PHP, що приймає HTTP-запити, опрацьовує їх і взаємодіє з БД через REST-API [10].

Такий поділ сприяє зрозумілому моделюванню, оскільки UML-діаграми чітко демонструють ролі та функції учасників системи, забезпечує масштабованість завдяки можливості додавати нові сервіси без порушення існуючої структури, а також спрощує підтримку програмного забезпечення за рахунок чіткого розмежування обов’язків між логічними рівнями.

Діаграма прецедентів демонструє, як основні ролі в системі взаємодіють із функціоналом веб-додатку. Вона розбита на дві частини – клієнтську і адміністративну – щоб одразу відокремити дії звичайного відвідувача від можливостей співробітника компанії (рис. 2.1).

Клієнт заходить на сайт без необхідності реєстрації, переглядає каталог автомобілів і може скористатися фільтрами або залишити заявку. Адміністратор працює у захищеній панелі керування й відповідає за наповнення та підтримку каталогу, публікацію новин і обробку запитів.

Ця діаграма чітко показує розмежування відповідальностей: клієнтська частина відповідає за пошук і замовлення авто, тоді як адміністраторська панель забезпечує керування контентом і обробку запитів. Такий поділ сприяє прозорому розумінню функціональних меж кожного користувача та полегшує подальший розвиток і підтримку системи.

Рисунок 2.1 – Діаграма прецедентів

Прецеденти актора «Клієнт»:

– перегляд каталогу авто (UC1). Клієнт відкриває сторінку з переліком усіх доступних лотів;

– фільтрація за маркою, ціною, роком (UC2). У межах каталогу застосовує фільтри, щоби швидко знайти потрібний варіант;

– надіслати заявку на підбір/покупку (UC3). Заповнює форму з ім’ям, телефоном, поштою, характеристиками авто й коментарем;

– перегляд новин (UC4). Знаходить актуальні повідомлення про поставки та внутрішні події компанії.

Прецеденти актора «Адміністратор»:

– додати/редагувати авто (UC5). Створює нові картки лотів або оновлює дані про наявні авто в каталозі;

– додати/редагувати новину (UC6). Публікує або коригує пости з заголовками, текстом і зображеннями;

– переглянути заявки (UC7). Отримує доступ до списку всіх клієнтських заявок і змінює їхній статус.

Діаграма класів ілюструє основні сутності предметної області та їхні властивості й зв’язки. Вона допомагає зрозуміти, які дані зберігає система й як вони між собою пов’язані, що є фундаментом для проєктування бази даних і реалізації бізнес-логіки (рис. 2.2).

Ця діаграма класів формалізує структуру даних та їхню взаємодію: вона закладає основу для побудови реляційної моделі в MySQL, забезпечуючи цілісність даних (через ключі) і спрощуючи подальшу імплементацію CRUD-операцій у контролерах PHP та відображення у компонентах Vue.js. Завдяки чіткому опису атрибутів і зв’язків розробник одразу бачить, які таблиці і як слід створити, а також як обробляти «Запити» і «Новини» в системі.

Зв’язки між класами:

– користувач (адміністратор) створює одну або кілька «Заявок», що відображає асоціацію 1 → 0..* між «Користувач» та «Заявка»;

– «Адміністратор» наслідує загальні властивості «Користувача», маючи розширені права;

– заявка може стосуватися одного чи кількох «Автомобілів (і навпаки), тобто між «Заявка» і «Автомобіль» існує зв’язок «багато-до-багатьох»;

– новина може згадувати один або кілька Автомобілів, що також реалізовано як зв’язок багато-до-багатьох між «Новина« та «Автомобіль».

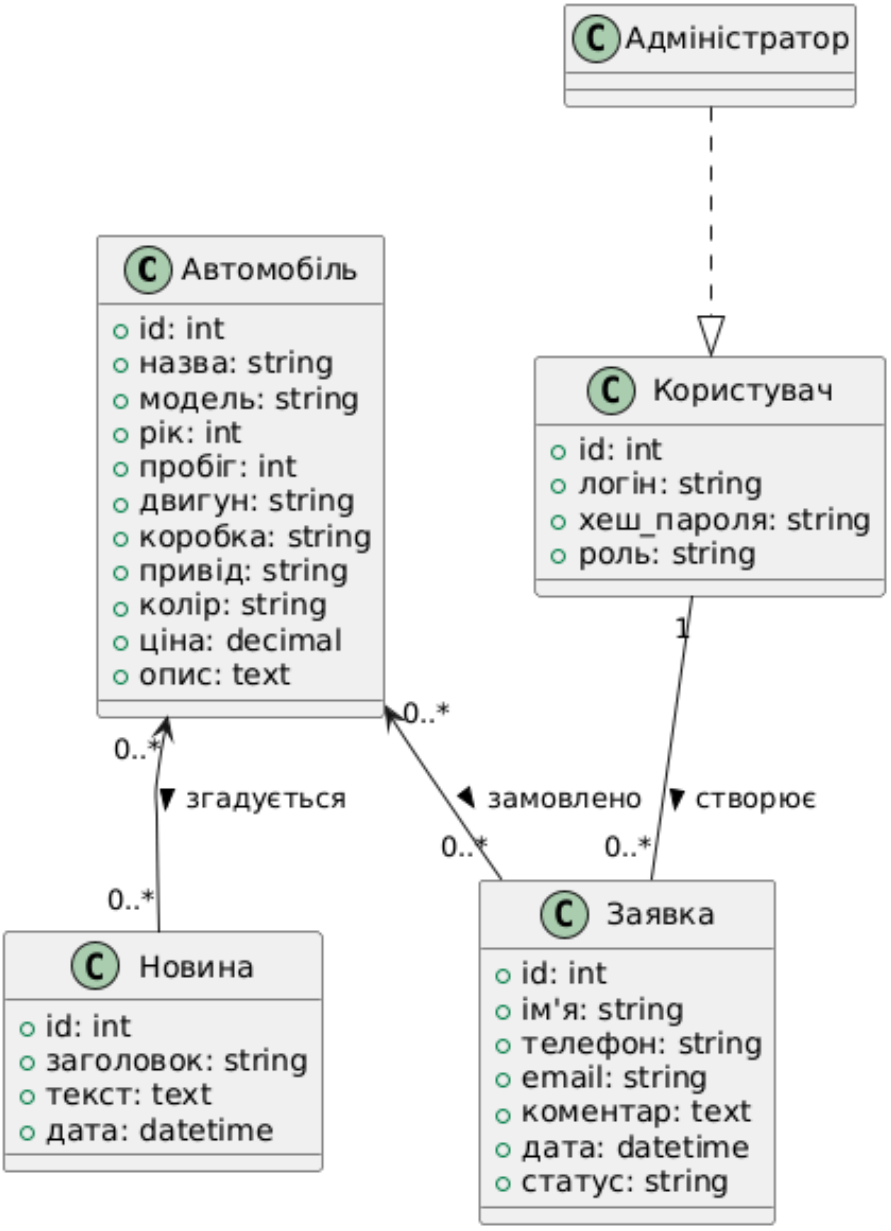

Рисунок 2.2 – Клас-діаграма

На клас-діаграмі представлено структуру програмної системи у вигляді класів із відповідними атрибутами та зв’язками.

1. «Автомобіль».

– id – унікальний ідентифікатор лоту;

– назва, модель, рік, пробіг, двигун, коробка, привід, колір, ціна – основні характеристики для пошуку та відображення в каталозі;

– опис – детальна текстова інформація та примітки.

2. «Заявка».

– id – первинний ключ заявки;

– ім’я, телефон, email – контактні дані клієнта;

– коментар – побажання чи уточнення до замовлення;

– дата – час надсилання;

– статус – поточний стан обробки (нове, в роботі, виконано тощо).

3. «Новина».

– id – унікальний ідентифікатор повідомлення;

– заголовок – короткий опис теми;

– текст – повний вміст;

– дата – дата публікації.

4. «Користувач».

– id – первинний ключ запису;

– логін, хеш_пароля – для аутентифікації в адмін-панелі;

– роль – рівень доступу (наприклад, «адміністратор»).

У результаті проведеної роботи обґрунтовано вибір UML-нотації та архітектурного стилю MVC для забезпечення чіткого моделювання системи. Сформульовано функціональні й нефункціональні вимоги, що охоплюють пошук і фільтрацію автомобілів, оформлення заявок, керування контентом, а також вимоги до продуктивності, безпеки та адаптивності. Наведено дві ключові UML-діаграми: діаграму прецедентів, яка ілюструє ролі клієнта й адміністратора, та клас-діаграму основних сутностей (Автомобіль, Заявка, Новина, Користувач) із відповідними атрибутами та взаємозв’язками.

2.2 Вибір засобів, методів і алгоритмів вирішення поставленого завдання

Правильний вибір технологічного стеку – це не просто підбір модних назв, а фундаментальний крок, який визначає, з якою швидкістю ви виведете продукт на ринок, наскільки легко його підтримувати та масштабувати, а головне – наскільки безпечним і надійним він буде для кінцевих користувачів і

бізнесу. Неправильне рішення на цьому етапі може призвести до затримок у розробці, проблем із продуктивністю під навантаженням або складнощів у впровадженні нових функцій.

Для успішної розробки необхідно ретельно обрати ключові технологічні компоненти, які визначатимуть ефективність, масштабованість та підтримку проєкту:

– фронтенд-фреймворк. Оцінка легкості навчання, швидкості рендерингу та багатства екосистеми (компоненти, плагіни, стан-менеджмент);

– серверний стек і мова програмування. Порівняння продуктивності, доступності хостингу, зрілості спільноти та кількості готових бібліотек;

– система керування базами даних (СУБД). Потреба в транзакційності, складності запитів і обсязі даних визначає вибір між реляційними та NoSQL-рішеннями;

– інструменти збірки та автоматизації. Засоби для швидкої розробки, збірки та розгортання фронтенду та бекенду (bundler, task-runner, CI/CD);

– середовище розробки та DevOps-утиліти. Контейнери для уніфікованого локального середовища, система контролю версій, інтеграція тестування й автоматичного деплою.

Правильний вибір фронтенд-фреймворку задає тон усьому проєкту: від швидкості розробки й простоти підтримки до плавності взаємодії кінцевих користувачів з інтерфейсом. Був обраний Vue.js, оскільки він ідеально поєднує простоту освоєння, реактивну модель даних і невеликий «ваговий» профіль бібліотеки.

Ключові аргументи на користь Vue.js:

– синтаксис на основі HTML-шаблонів і зрозумілий API роблять перші кроки швидкими навіть для тих, хто працює з JavaScript уперше;

– Vue автоматично відстежує залежності даних і оновлює саме ті частини DOM, які змінилися, що підвищує продуктивність інтерфейсу;

– офіційні пакети «vue-router» для маршрутизації й «Pinia» для стан-менеджменту забезпечують однорідний досвід і просту інтеграцію;

– бібліотека разом із базовими плагінами займає близько 50 КБ у стисненому вигляді, що скорочує час першого завантаження сторінки;

– величезна кількість відкритих компонентів і плагінів допомагає швидко реалізовувати нестандартні UI-елементи.

Статистичні дані підтверджують широку підтримку Vue.js у професійному середовищі. За результатами опитування JetBrains Developer Ecosystem 2023, Vue регулярно використовують 32 % розробників, що ставить його на друге місце за популярністю після React (57 %) і випереджає Angular (20 %) [11]. Щотижневі завантаження пакету «@vue/core» на npm перевищують 5 млн, що свідчить про стабільне застосування Vue в проєктах різного масштабу. І, нарешті, за даними опитування State of JS 2024, Vue зберіг другу позицію серед фреймворків за часткою опитаних і показав зростання в утриманні користувачів на три процентних пункти (рис. 2.3).

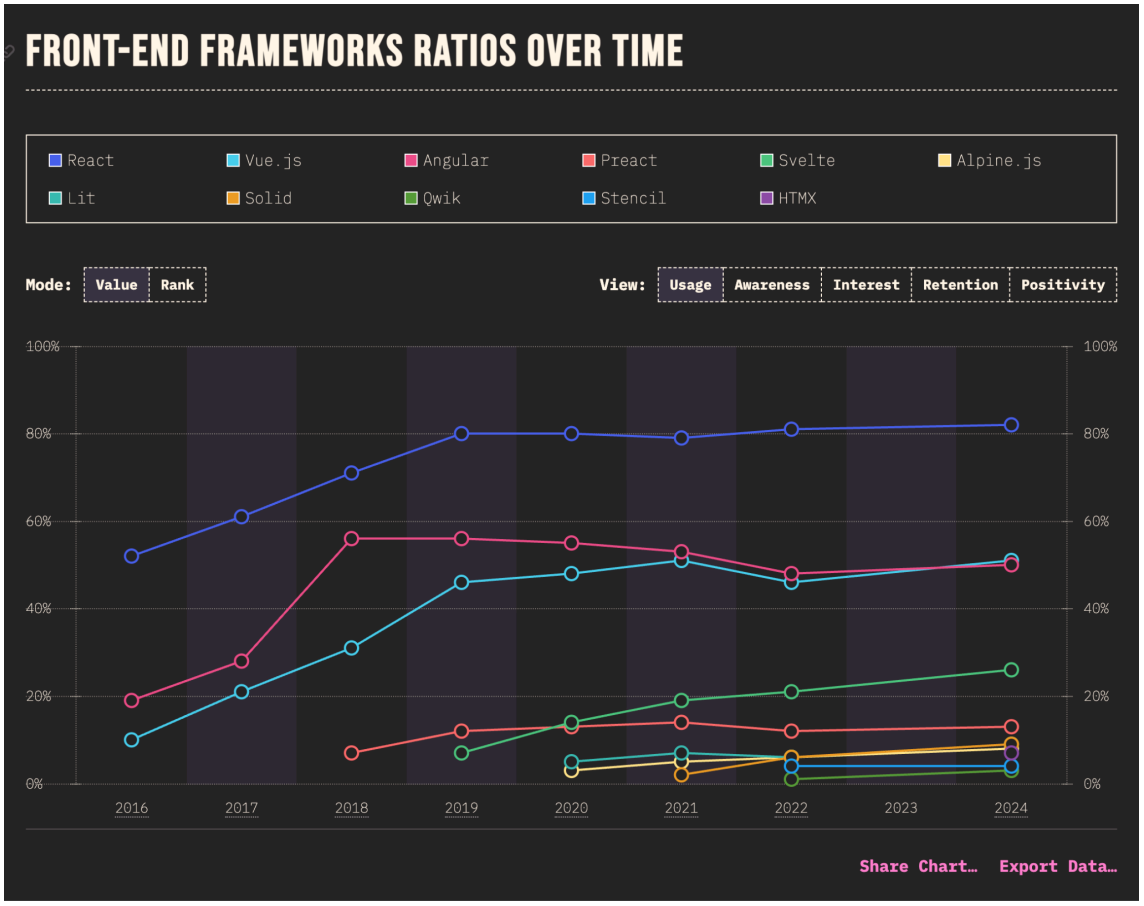

Рисунок 2.3 – Динаміка популярності фронтенд-фреймворків [11]

Вибір Vue.js забезпечує нам швидкий старт, гнучкість архітектури та високу продуктивність інтерфейсу, що є критично важливим для користувацького досвіду у вебдодатку ПП «ІНТЕРТРЕЙД-ЛУЦЬК».

Правильний вибір серверного стека та мови програмування прямо впливає на швидкість розробки, продуктивність під навантаженням і вартість підтримки системи. Для вебдодатку ПП «ІНТЕРТРЕЙД-ЛУЦЬК» обрано поєднання PHP 8 + MySQL + Nginx/Apache, оскільки це класична LAMP/LEMP-конфігурація.

На початок 2025 року PHP використовується у 74,5 % всіх сайтів з відомим серверним ПЗ [12], а 86 % розробників уже перейшли на версії 8.x заради кращої продуктивності й сучасних можливостей [13]. Практично всі провайдери надають підтримку PHP «з коробки», а MySQL є стандартною реляційною базою в більшості середовищ. PHP 8 приніс JIT-компіляцію, union-типи, атрибути, покращену обробку помилок і оптимізований синтаксис, що дозволяє писати чистий, типізований код із високою продуктивністю та безпекою.

Цей стек органічно поєднується з Vue.js: фронтенд працює як незалежний клієнт і звертається до нашого бекенду через легкий REST-API, передаючи й отримуючи дані у форматі JSON. Розділення обов’язків («frontend-backend»):

– Vue.js відповідає за побудову інтерактивного інтерфейсу та реактивну роботу з даними;

– PHP на стороні сервера приймає HTTP-запити, виконує бізнес-логіку (обробка фільтрів, форм заявок, CRUD для авто/новин), взаємодіє з MySQL і повертає відповіді клієнту.

Такий підхід дозволяє масштабувати фронтенд і бекенд окремо, спрощує тестування й підтримку, а також забезпечує гнучкість: у майбутньому за потреби можна додати мікросервіси (наприклад, окремий сервіс аналітики чи чат-бот) без зміни наявного коду Vue.

Вибір MySQL обґрунтовано його перевіреною надійністю та широким поширенням у корпоративному секторі та серед невеликих проєктів. Станом на

червень 2025 року за даними DB-Engines Ranking MySQL посідає перше місце серед відкритих реляційних СУБД і друге загалом, поступаючись лише Oracle [14]. Рушій InnoDB гарантує повну підтримку ACID-транзакцій, відмовостійкість та ефективне блокування рядків, що критично важливо для коректної обробки одночасних запитів клієнтів вебдодатку. Підтримка JSON-полів і повнотекстового пошуку дозволяє здійснювати гібридні запити без відмови від реляційної моделі.

Наявність інструментів адміністрування, таких як phpMyAdmin і MySQL Workbench, а також вбудовані механізми реплікації й кластери роблять супровід і масштабування бази простими й передбачуваними. У контексті LAMP/LEMP-аріхтектури з PHP на бекенді і Vue.js на фронтенді MySQL виступає ідеальним вибором, що поєднує швидкість, безпеку та простоту інтеграції.

Vite повністю переосмислює фронтенд-воркфлоу, відмовляючись від традиційної попередньої збірки під час розробки та використовуючи нативні ES-модулі для миттєвого завантаження тільки змінених файлів, що дозволяє локальному серверу стартувати за лічені мілісекундита оновлювати інтерфейс у режимі «гарячої» перезагрузки майже без затримок. Завдяки такому підходу цикл «змінив – перевірив» скорочується в рази, особливо в проєктах із великою кодовою базою. Стабільність і популярність Vite підтверджує статистика: понад 424 628 живих сайтів вже працюють у продакшні з Vite [15].

У цьому проєкті Vite стане базою для швидкої розробки інтерфейсу на Vue.js, забезпечуючи просту конфігурацію, розширювану плагінами архітектуру та оптимізовані продакшн-бандли через Rollup, що значно підвищить ефективність роботи й зручність підтримки коду.

Для локальної розробки використовувався MAMP – зручний набір із Apache, MySQL та PHP, який дозволяє підняти робоче середовище буквально в один клік без складної конфігурації [16]. Завдяки MAMP усі компоненти вже налаштовані та готові до роботи на Windows і macOS, що особливо зручно, коли потрібно швидко перевірити зміни в контролерах PHP або протестувати міграції

бази даних без зайвих зусиль. Крім того, інтерфейс MAMP надає доступ до налаштувань версій PHP і MySQL, дозволяючи легко переключатися між різними релізами в разі потреби локального тестування сумісності.

В якості основного редактора обрано Visual Studio Code [17]. Його швидкий старт, мінімалістичний інтерфейс і величезний каталог розширень створюють ідеальні умови для роботи з Vue.js і PHP одночасно. Завдяки плагінам для синтаксичного підсвічування, інтелектуального автодоповнення та інтегрованого дебагера можна одразу переходити від написання компонентів у Vue до налагодження бекенд-методів у PHP без перемикання вікон. Вбудована підтримка Git і можливість налаштувати завдання (tasks) під запуск скриптів Composer або npm роблять VS Code справжнім «швейцарським ножем» в арсеналі розробника та суттєво прискорюють цикл «код-перевірка-деплой».

У підсумку було визначено ключові критерії для вибору інструментів – швидкість і простота розробки, підтримуваність і масштабованість, безпека, вартість експлуатації. Вибраний стек не лише відповідає технічним вимогам проєкту, але й забезпечує гнучкість подальшого розвитку: чітке розділення клієнтської та серверної частин через REST-API полегшує масштабування, а готові DevOps-утиліти дозволяють уніфікувати середовище розробки та впровадити CI/CD-процеси.

РОЗДІЛ 3
РОЗРОБКА ВЕБ-ДОДАТКУ ПП «ІНТЕРТРЕЙД-ЛУЦЬК»

3.1 Практична реалізація об’єкта проектування

Спочатку було встановлено локальний сервер за допомогою MAMP: у кілька кліків піднято сервер Apache, PHP 8.x і MySQL. Це дало змогу миттєво перевіряти як фронтенд-запити, так і запити до бази без необхідності складного налаштування (рис. 3.1).

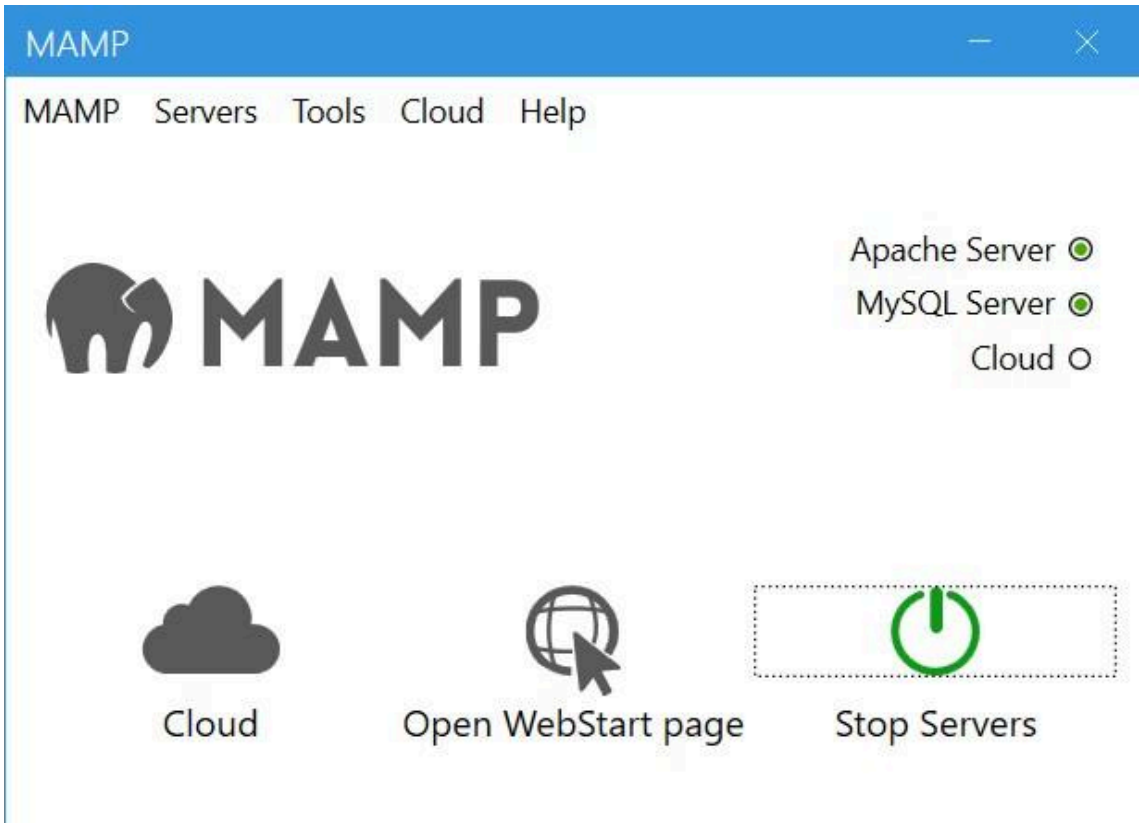

Рисунок 3.1 – Запущений сервер у програмі MAMP

Після цього в каталозі проєкту ініціалізується Vue-додаток через Vite командою «npm create vite@latest», обрано шаблон «vue» і встановлено всі залежності через «npm install». Одночасно створено структуру бекенду:

– api/config/ для налаштувань підключення до бази;

– api/controllers/ для обробки HTTP-запитів;

– api/models/ для доступу до даних;

– api/uploads/ для фото автомобілів;

– public/ для статичних файлів (index.html, favicon).

Далі у VS Code були підключені розширення для Vue, ESLint і PHP Intellisense, щоб одразу мати перевірку синтаксису й автодоповнення.

Нижче показана структура файлів і папок (рис. 3.2). Вона допомагає зберегти чистоту коду й зрозумілість при масштабуванні.

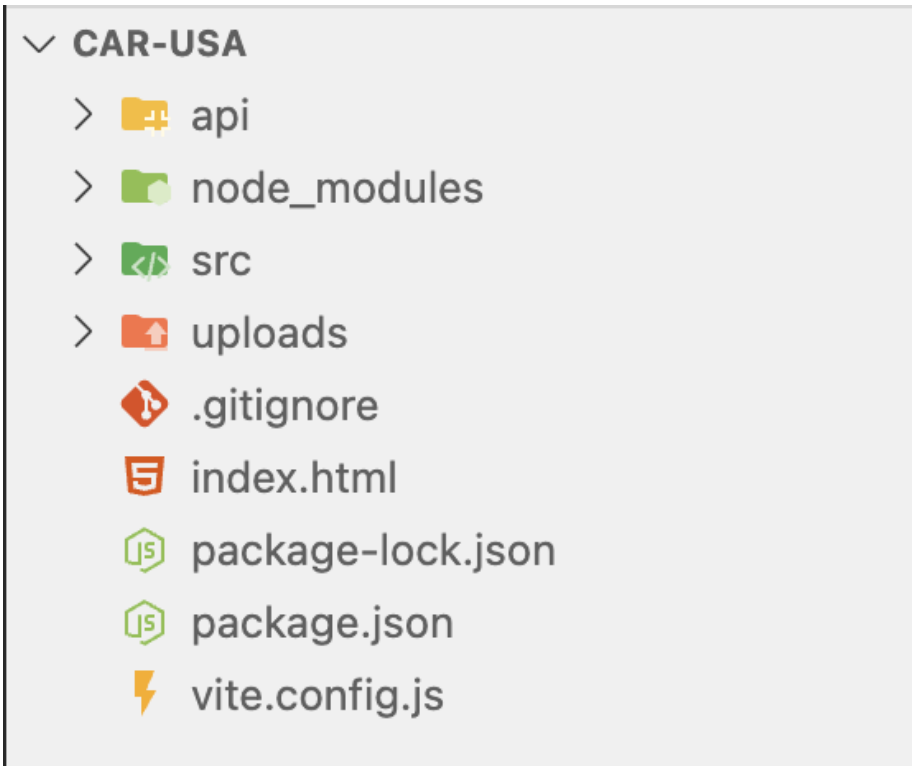

Рисунок 3.2 – Структура файлів і папок

Для підключення до бази в api/config/database.php використали фрагмент коду наведеного в лістингу 3.1.

Лістинг 3.1 – Підключення до бази в api/config/database.php

```
<?php
function getConnection() {
   $host = 'localhost';
   $dbname = 'car_usa';
   $username = 'root';
   $password = 'root';
      $port = 3306; // Змініть порт на стандартний 3306 для віддалених серверів (не 8889)
```

```
    try {
        $pdo = new PDO(
            "mysql:host=$host;dbname=$dbname;charset=utf8mb4",
            $username,
            $password,
            [
                PDO::ATTR_ERRMODE => PDO::ERRMODE_EXCEPTION,
                PDO::ATTR_DEFAULT_FETCH_MODE => PDO::FETCH_ASSOC,
                PDO::ATTR_EMULATE_PREPARES => false
            ]
        );
        return $pdo;
    } catch (PDOException $e) {
        // Зберігаємо детальну інформацію про помилку в лог
                    error_log("Database connection error: " .
$e->getMessage());

        // Повертаємо більш детальну помилку
                throw new PDOException("Connection failed: " .
$e->getMessage());
    }
}

// Функція для тестування підключення з виведенням результату
function testConnection() {
    try {
        $conn = getConnection();
        echo "Успішне підключення до бази даних!";
        return true;
    } catch (PDOException $e) {
        echo "Помилка підключення: " . $e->getMessage();
        return false;
    }
}

// Розкоментуйте наступний рядок для тестування
// testConnection();
```

кінець лістингу 3.1

І нарешті, у vite.config.js мінімально підключено плагін Vue та налаштували проксі (ліст. 3.2).

Лістинг 3.2 – Конфігурація у файлі vite.config.js

```
import { defineConfig } from 'vite'
import vue from '@vitejs/plugin-vue'
import path from 'path'
```

```
// https://vitejs.dev/config/
export default defineConfig({
 plugins: [vue()],
 resolve: {
   alias: {
     '@': path.resolve(__dirname, './src')
   }
 },
 server: {
   port: 5173,
   proxy: {
     '/api': {
       target: 'http://localhost/car-usa',
       changeOrigin: true,
       rewrite: (path) => path.replace(/^\/api/, '')
     }
   }
 }
})
```

кінець лістингу 3.2

Середовище готове: фронтенд і бекенд чітко розділені, всі залежності встановлені, а структура папок одразу підказує, куди додавати нову функціональність чи фрагмент коду. Це дозволить швидко рухатися далі до розробки конкретних модулів.

Далі виконано ініціалізацію клієнтської частини через Vite і підключення всіх необхідних npm-залежностей. У терміналі були виконані команди з лістингу 3.3.

Лістинг 3.3 – Ініціалізація клієнтської частини через Vite

```
npm create vite@latest intertrade-lutsk --template vue
cd intertrade-lutsk
npm install vue@3 vue-router@4 pinia axios
npm install -D eslint vite
```

кінець лістингу 3.3

Це автоматично створило package.json з базовими скриптами (dev, build, serve) (ліст. 3.4), а також конфігурацію vite.config.js із налаштованим плагіном Vue і проксі на бекенд.

Лістинг 3.4 – Конфігурація у файлі package.json

```
{
 "name": "car-usa",
 "version": "1.0.0",
 "private": true,
 "scripts": {
   "dev": "vite",
   "build": "vite build",
   "preview": "vite preview"
 },
 "dependencies": {
   "axios": "^1.6.7",
   "vue": "^3.4.15",
   "vue-router": "^4.5.1"
 },
 "devDependencies": {
   "@vitejs/plugin-vue": "^5.0.3",
   "vite": "^5.1.0"
 }
}
```

кінець лістингу 3.4

Проведено ініціалізацію бекенд-частини через Composer: у корені проекту виконано composer init, після чого під’єднано пакет для роботи з оточенням (vlucas/phpdotenv) та JWT-автентифікацію (firebase/php-jwt).

У результаті з’явився файл composer.json із секцією автозавантаження PSR-4, яка спрямовує неймспейси на папку api/.

Проведено налаштування ESLint для Vue та PHP CS Fixer для бекенду, що забезпечило єдині стандарти коду. Після встановлення залежностей виконано запуск команд npm run dev для фронтенд-сервера та php -S localhost:8888 -t api/ для бекенду, що дозволило одночасно розробляти обидві частини системи в єдиному середовищі.

Виконано налаштування точки входу фронтенда та маршрутизації за допомогою Vue Router, що забезпечило клієнтській частині чітку навігацію між сторінками додатку.

Створено файл src/router/index.js, де ініціалізовано Vue Router у режимі history та визначено маршрути для основних View-компонентів (додаток В).

Після цього виконано інтеграцію роутера в основний файл src/main.js, щоб забезпечити доступ до маршрутів у всьому додатку (ліст. 3.5).

Лістинг 3.5 – Інтеграцію роутера в основний файл src/main.js

```
import { createApp } from 'vue'
import App from './App.vue'
import router from './router'
import axios from 'axios'
import './assets/main.css'

// Configure axios defaults
axios.defaults.baseURL = 'http://localhost/car-usa'

const app = createApp(App)

// Make axios available globally
app.config.globalProperties.$axios = axios

// Use router
app.use(router)

// Mount app
app.mount('#app')
```

кінець лістингу 3.5

Клієнтська частина готова до динамічного переходу між сторінками без перезавантаження, а структура маршрутизації чітко відокремлює публічний функціонал (головна, пошук авто, новини, форма замовлення, контакти) від секції адмін-панелі (логін, дашборд). Це сприяє чистій організації коду та полегшує подальше розширення навігації (додавання захищених маршрутів, lazy-loading компонентів та інше).

Виконана реалізація основних Views, які забезпечують весь базовий функціонал як для кінцевого користувача, так і для адміністратора:

– Home.vue – лендингова сторінка з короткою інформацією про компанію, ключовими перевагами та закликом до дії (CTA);

– AvailableCars.vue – каталог доступних автомобілів із панеллю фільтрів за маркою, ціною та роком; картки лотів відображають базові характеристики та фото;

– OrderForm.vue – сторінка з формою замовлення: поля для імені, телефону, email, вибору моделі авто та коментаря;

– News.vue – перелік актуальних публікацій із заголовками, короткими анонсами й датами; кожна новина веде на детальний перегляд;

– Contacts.vue – статична сторінка з реквізитами офісу, телефонами, email та картою проїзду.

У секції адмін-панелі реалізовано:

– Admin/Login.vue – сторінка авторизації з полями login/password і обробкою помилок;

– Admin/Dashboard.vue – оглядова панель із швидким доступом до розділів керування авто, новинами та заявками;

– Admin/Cars.vue – список лотів із кнопками «Додати», «Редагувати» і «Видалити», а також формою для введення всіх атрибутів авто;

– Admin/News.vue – інтерфейс для створення та редагування публікацій з полем заголовка, текстом і можливістю завантаження зображень;

– Admin/Orders.vue – таблиця клієнтських заявок із відображенням контактів, коментарів і статусів; можливість змінити статус заявки.

Реалізовані сторінки покривають весь спектр бізнес-процесів: від залучення й інформування клієнта до повного циклу управління контентом і обробки замовлень в адмін-панелі.

Бекенд-контролери реалізовані у вигляді окремих PHP-скриптів у корені каталогу api та в підпапці api/admin. Це рішення пришвидшує старт проєкту й робить маршрутизацію максимально зрозумілою:

– запити до api/cars.php обробляють параметри $_GET для фільтрації та повертають відфільтрований масив авто у форматі JSON;

– в api/orders.php метод POST читає JSON-тіло через php://input, перевіряє обов’язкові поля та зберігає заявку в БД, повертаючи HTTP/1.1 201 Created;

– скрипт api/news.php реалізує вивантаження всіх новин, відсортованих за датою;

– в адмін-папці файли cars.php, news.php і orders.php обробляють CRUD-операції, включно з завантаженням зображень через $_FILES та оновленням статусу заявок;

– файл api/config/auth.php містить функції перевірки токена або сесії, які викликаються на початку кожного адмін-скрипта, щоб захистити кінці точок від неавторизованого доступу.

Таке спрощене розташування файлів робить бекенд прозорим: кожен скрипт відповідає за один набір операцій (список авто, заявки чи новини), а центральна конфігурація в config/database.php і config/auth.php зводить усі загальні налаштування в одне місце. Це гарантує легке розуміння коду навіть для нових учасників команди та забезпечує швидку навігацію по проєкту.

Виконано підключення до MySQL через PDO: у файлі api/config/database.php налаштовані параметри з’єднання (хост, ім’я бази, користувач, пароль, кодування), а єдиний метод getConnection() повертає готовий об’єкт PDO з увімкненим режимом помилок і асоціативним режимом вибірки.

Схема таблиць побудована відповідно до UML-моделі: є таблиці cars (для лотів авто із полями id, make, model, year, mileage, engine, gearbox, drive, color, price, description), orders (для заявок із полями id, name, phone, email, comment, date, status), news (для публікацій із id, title, text, date) і додаткова таблиця users для адміністраторів (id, login, password_hash, role). Індекси встановлені на полях, за якими відбувається пошук (make, price, year), а зовнішні ключі – між orders і cars, щоб забезпечити цілісність даних. Детальний опис DDL та налаштування зв’язків між таблицями наведено в наступному розділі про базу даних.

Для завантаження фотографій автомобілів у адмін-панелі передбачено обробку масиву $_FILES прямо в скрипті api/admin/cars.php (ліст. 3.6). Після валідації токена адміністратора та перевірки обов’язкових полів форми, скрипт проходить по кожному тимчасовому файлу у $_FILES['images'], визначає поширення оригінального імені та створює унікальне ім’я на основі функції

uniqid(). Далі відбувається перевірка MIME-типу (щоб не дозволити завантаження небезпечних файлів) і розміру (наприклад, не більше 5 МБ). Після цього кожен файл переміщується з тимчасової папки до api/uploads/cars/, а за допомогою GD-функцій автоматично генерується мініатюра з фіксованою шириною 300 px, яка зберігається поруч у папці thumbnails/. У разі помилки скрипт повертає JSON із кодом 400 і повідомленням про невдалість завантаження, що дозволяє фронтенду показати відповідне оповіщення адміністратору.

Лістинг 3.6 – Обробка масиву $_FILES

```
// Фрагмент коду: обробка завантаження зображень у api/admin/cars.php
if (!empty($_FILES['images']['tmp_name'])) {
    foreach ($_FILES['images']['tmp_name'] as $i => $tmpPath) {
        $orig = $_FILES['images']['name'][$i];
        $ext  = pathinfo($orig, PATHINFO_EXTENSION);
        $newName = uniqid('car_', true) . '.' . $ext;
        $dest = __DIR__ . '/../uploads/cars/' . $newName;
        // Перевірка MIME-типу і розміру
        $finfo = finfo_open(FILEINFO_MIME_TYPE);
        $type = finfo_file($finfo, $tmpPath);
        finfo_close($finfo);
             if (!in_array($type, ['image/jpeg','image/png']) || $_FILES['images']['size'][$i] > 5*1024*1024) {
            http_response_code(400);
            echo json_encode(['error' => 'Invalid image file']);
            exit;
        }
        // Переміщення основного файлу
        if (!move_uploaded_file($tmpPath, $dest)) {
            http_response_code(500);
            echo json_encode(['error' => 'Upload failed']);
            exit;
        }
        // Генерація мініатюри через GD
        list($w, $h) = getimagesize($dest);
        $thumbW = 300; $thumbH = intval($h * ($thumbW / $w));
        $srcImg = $type === 'image/png'
            ? imagecreatefrompng($dest)
            : imagecreatefromjpeg($dest);
        $thumb = imagecreatetruecolor($thumbW, $thumbH);
            imagecopyresampled($thumb, $srcImg, 0,0,0,0, $thumbW, $thumbH, $w, $h);
          $thumbPath = __DIR__ . '/../uploads/cars/thumbnails/' . $newName;
```

```
            imagejpeg($thumb, $thumbPath);
            imagedestroy($srcImg);
            imagedestroy($thumb);
            // Занести ім'я файлу в БД через модель Car
            $images[] = $newName;
        }
        // Наприклад: $carModel->saveImages($carId, $images);
}
```

кінець лістингу 3.6

Цей підхід гарантує безпеку (лише зображення в дозволених форматах), передбачуваність і консистентність (унікальні назви файлів) та готову для відображення на сайті версію фото у вигляді мініатюр, що прискорює рендеринг каталогу.

Виконано базову валідацію на клієнтській стороні: усі форми перевіряють коректність email і телефонного номера, наявність обов'язкових полів і довжину коментаря ще до відправки запиту. На сервері застосовано підготовлені вирази PDO і функції filter_var() для очищення вхідних даних, що повністю виключає SQL-ін'єкції, а також встановлено заголовки безпеки (CORS, X-Content-Type-Options, X-Frame-Options) і CSRF-токени для захисту адмін-форм. Детальніше механізми валідації та протидії вебзагрозам описані в окремому розділі.

Інтеграція клієнтської частини з бекендом реалізована через простий та прозорий REST-інтерфейс. У Vue-компонентах ми використовуємо axios для взаємодії з PHP-скриптами в папці api/. Наприклад, у компоненті AvailableCars.vue виклик onMounted виконує запит із параметрами фільтрів і оновлює реактивну змінну cars, що автоматично перерисовує список. Аналогічно, у формі замовлення (OrderForm.vue) метод submitOrder надсилає JSON-тіло, а бекенд повертає підтвердження з HTTP 201 Created.

Весь обмін даними захищений через HTTPS, а клієнт обробляє коди відповідей і в разі помилок виводить користувачеві інформативні повідомлення. Такий підхід забезпечує чіткий поділ обов'язків: Vue відповідає за

відображення й взаємодію, PHP – за бізнес-логіку та роботу з базою, що спрощує масштабування та тестування кожного з шарів окремо.

У результаті практичної реалізації ми отримали повноцінний вебдодаток, у якому чітко розділено клієнтську та серверну логіку, встановлено однорідне середовище розробки й структуровану файлову систему. Фронтенд на Vue.js забезпечує інтуїтивну навігацію та реактивний інтерфейс із динамічною фільтрацією та відправкою форм, а бекенд на PHP із використанням PDO-з'єднання й REST-контролерів гарантує безпечну обробку запитів і збереження даних у MySQL. Завантаження фото автомобілів і генерація мініатюр реалізовані через надійну обробку $_FILES і GD-бібліотеку, а інтеграція з клієнтом відбувається за допомогою axios із прозорим REST-API. Попередня валідація на обох рівнях та базові заходи безпеки створюють міцний фундамент для подальшого вдосконалення. Такий підхід забезпечує простоту підтримки, масштабованість системи й готовність до розширення новими функціональними модулями.

3.2 Тестування та налагодження інформаційно-комп'ютерної системи

Для гарантування надійності й коректності роботи вебдодатку застосовано комплексний підхід до тестування та налагодження, що включає автоматизовані та ручні методи.

По-перше, здійснено модульне тестування бекенд-логіки за допомогою PHPUnit. Тести покривають основні методи моделей (Car::getFiltered(), Order::create(), News::all()) і контролерів, перевіряючи правильність SQL-запитів та обробки помилок. Завдяки цьому вдалося виявити й усунути помилку з некоректною передачею параметрів фільтрації за роком: замість числового значення в SQL потрапляв рядок, що призводило до порожнього результату.

По-друге, на фронтенді використовували Vue Test Utils [18] та Jest [19] для тестування окремих компонентів: CarFilter.vue і OrderForm.vue. Було перевірено, що за різних комбінацій вхідних props відпрацьовують усі гілки

логіки (відображення помилок валідації, емісія подій filter-changed), і відреаговано на виявлену невідповідність очікувань у поведінці кнопки «Надіслати», виправивши порядковість перевірок у методі submitOrder().

Інтеграційні тести проводилися через Postman [20]: створено колекцію запитів до всіх кінцевих точок API (/api/cars.php, /api/orders.php, /api/news.php, /api/admin/...). Це дозволило перевірити сценарії end-to-end – від відправки форми на фронтенді до появи запису в таблиці orders. В ході тестування було виявлено, що при великій кількості фотографій одночасне надсилання масиву $_FILES іноді призводило до помилки через перевищення post_max_size; проблема була вирішена збільшенням відповідних налаштувань у php.ini та додаванням обробки помилки на боці клієнта.

Для перевірки коректного відображення інтерфейсу й юзабіліті проведено ручне кросбраузерне тестування в Chrome, Firefox і Edge, а також перевірку адаптивності на мобільних емуляторах. Було усунуто недоліки в стилях: модернізовано компоненти фільтрів, щоб уникнути «обриву» панелі фільтрації на вузьких екранах.

Усі виявлені недоліки були своєчасно усунуті: виправлено серверні налаштування для обробки великих завантажень, додано обробку помилок у формах, оптимізовано SQL-запити індексами, вдосконалено стилі CSS для коректної роботи на мобільних пристроях. Такий підхід до тестування та налагодження став запорукою стабільності й передбачуваності роботи вебдодатку ПП «ІНТЕРТРЕЙД-ЛУЦЬК».

3.3 Розробка бази даних

Реалізовано фізичну модель бази даних (рис. 3.3) відповідно до раніше побудованих UML- і ER-діаграм. Як СУБД обрана MySQL з рушієм InnoDB, що забезпечує повноцінну ACID-транзакційність та підтримку зовнішніх ключів.

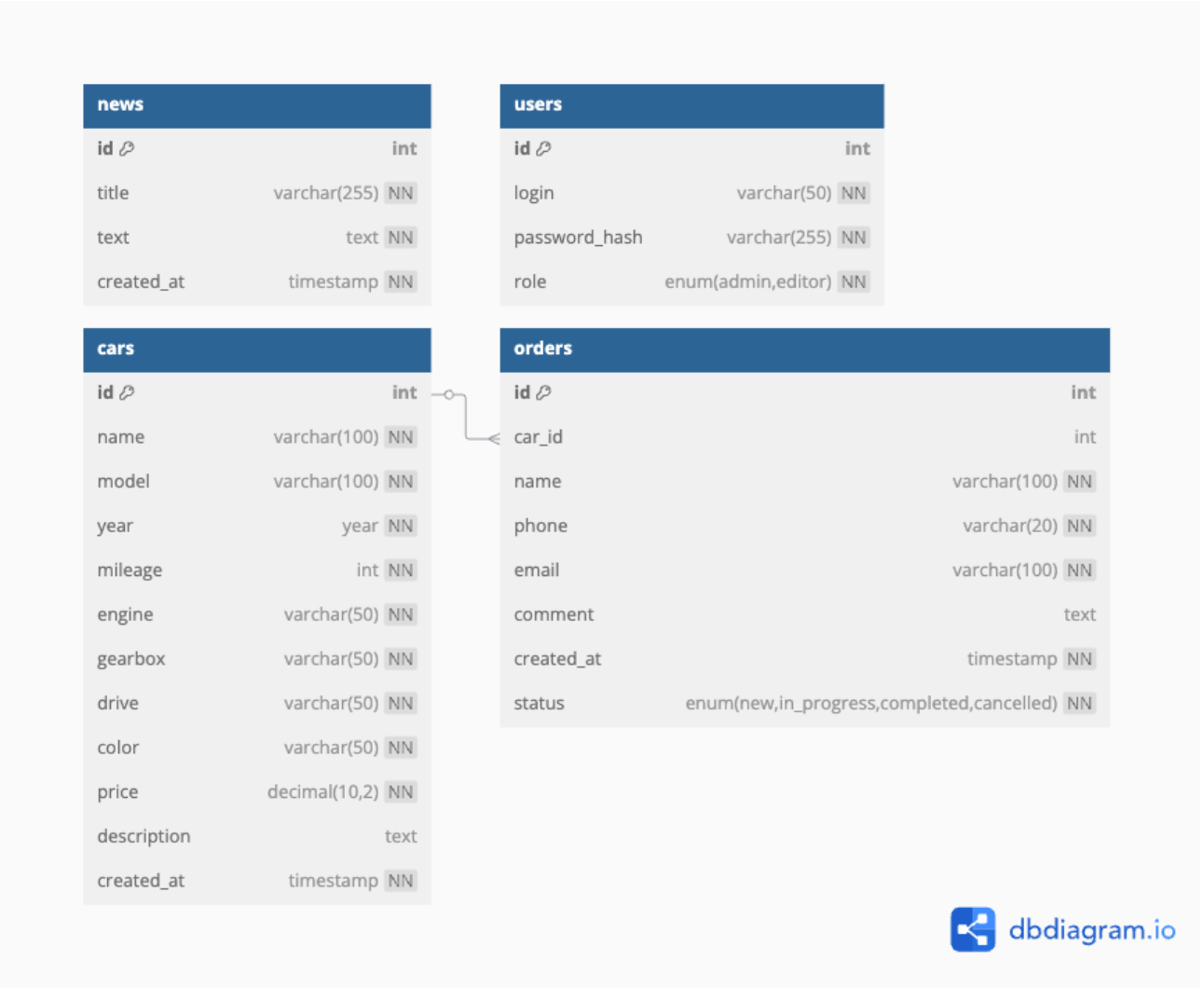

Рисунок 3.3 – Схема БД

Основні сутності представлено чотирма таблицями:

– users з полями id, login, password_hash та role, де id – авто-інкрементний первинний ключ, а login – унікальний індекс для швидкої аутентифікації адміністраторів;

– cars з атрибутами name, model, year, mileage, engine, gearbox, drive, color, price, description і міткою created_at, що автоматично фіксує дату додавання лоту;

– orders, у якій поле car_id посилається на cars.id через зовнішній ключ із каскадним видаленням, а також містяться контактні дані замовника, коментар та статус обробки;

– news для зберігання публікацій із заголовком, текстом та датою створення.

Всі таблиці мають індекси на полях, які використовуються у фільтрації (наприклад, cars.year і cars.price), що підвищує швидкість виконання запитів. Будь-яка спроба видалити автомобіль призведе до автоматичного видалення пов’язаних із ним заявок, завдяки дефініції ON DELETE CASCADE. Ця структура гарантує цілісність даних, простоту написання SQL-запитів і легке розширення моделі (наприклад, додавання таблиці images для пов’язування декількох фото з одним авто).

3.4 Захист інформаційно-комп’ютерної системи

У процесі захисту вебдодатку ПП «ІНТЕРТРЕЙД-ЛУЦЬК» ми керувалися принципами Defense in Depth, застосувавши багаторівневі заходи для запобігання поширеним вебзагрозам. По-перше, весь трафік між клієнтом і сервером шифрується за допомогою HTTPS (TLS 1.2+), що виключає перехоплення чутливих даних під час передачі. Налаштування вебсервера (Nginx/Apache) доповнено заголовками безпеки: Strict-Transport-Security для примусового використання TLS, X-Content-Type-Options і X-Frame-Options для захисту від MIME-сніфінгу та clickjacking, а також Content-Security-Policy для обмеження джерел скриптів і стилів.

Аутентифікація в адмін-панелі реалізована через PHP-сесії з обов’язковим логіном і паролем. Паролі хешуються алгоритмом bcrypt із достатнім фактором ітерацій (cost ≥12), що робить атаку методом підбору практично недоступною. Для додаткового захисту від brute-force атаки введено ліміти невдалих спроб входу з тимчасовою блокуванням IP-адреси. Кожен захищений маршрут перевіряє наявність і валідність сесійної куки, а дані адмін-запитів (POST, PUT, DELETE) підкріплені унікальними CSRF-токенами, які генеруються на сервері та перевіряються при кожному запиті, запобігаючи несанкціонованому виконанню операцій.

Щоб виключити ризик SQL-ін’єкцій, усі запити до MySQL здійснюються через PDO із підготовленими виразами (prepared statements), а вхідні дані проходять серверну валідацію з `filter_var()` та кастингом до безпечних типів. На рівні фронтенду додатково працюють бібліотеки валідації форм (VeeValidate), що перешкоджають надходженню некоректних значень. Відображувальний шар у Vue.js обробляє користувацький ввод, екрануючи небезпечні символи, а HTTP-заголовки та бібліотека DOMPurify забезпечують захист від XSS-атак.

Обробка завантаженого контенту (зображень) також вбудовує етапи перевірки MIME-типу через finfo, обмеження розмірів файлів і зберігання їх поза межами документ-руту. Імена файлів генеруються за допомогою `uniqid()`, що виключає конфлікти та запобігає доступу за вгаданим шляхом. Аудит подій (успішні та невдалі спроби входу, створення/видалення записів) фіксується в логах, які обмежені в доступі та періодично архівуються з шифруванням резервних копій.

Усі ці заходи доповнені регулярним оновленням компонентів стека (PHP, MySQL, Nginx, бібліотек NPM/Composer) та щомісячним переглядом залежностей через інструменти типу Dependabot. Таке системне поєднання технологічних і процедурних рішень створює надійний захисний бар’єр і забезпечує стійкість системи до широкого спектра сучасних кіберзагроз.

ВИСНОВКИ

Отже, усі поставлені завдання виконано: проведено ґрунтовний аналіз ринку й конкурентів, чітко сформульовано функціональні та нефункціональні вимоги, спроектовано архітектуру й модель даних у MySQL, обрано оптимальний технологічний стек (Vue.js, PHP, MySQL, Vite), реалізовано клієнтську й серверну частини з REST-API, проведено модульне, інтеграційне та end-to-end тестування, а також впроваджено багаторівневі заходи безпеки. Розроблений вебдодаток готовий до подальшого розгортання та відповідає всім технічним і бізнес-вимогам ПП «ІНТЕРТРЕЙД-ЛУЦЬК».

Підсумовуючи виконання першого завдання, було здійснено ґрунтовний аналіз ринку імпорту вживаних автомобілів із США в Україні, зібрано ключові статистичні показники 2024-2025 років та розглянуто досвід трьох локальних конкурентів. Завдяки цьому вдалося чітко визначити сильні та слабкі сторони їхніх сервісів – від наявності онлайн-калькуляторів і гарантій до браку особистого кабінету й неузгодженості гарантійних умов – і сформувати обґрунтування для створення більш прозорого й зручного рішення.

У межах другого завдання було сформульовано детальний набір функціональних вимог (динамічна фільтрація за маркою, ціною й роком, подача заявок, CRUD-операції для авто та новин, історія заявок) і нефункціональних критеріїв (швидкість завантаження, масштабованість, безпека, адаптивний дизайн та SEO). Це дозволило звести до єдиного переліку чіткі технічні й бізнес-орієнтовані очікування від системи.

Третє завдання включало проєктування архітектури в стилі MVC та побудову моделі даних у MySQL. Створені UML-діаграми прецедентів і класів, ER-схема й DBML-опис таблиць заклали основу для фізичної моделі, в якій InnoDB-зовнішні ключі й індекси гарантують цілісність та продуктивність запитів.

Четверте завдання полягало в обґрунтуванні вибору технологічного стеку: Vue.js забезпечує швидку розробку з мінімальною кривою навчання та

реактивним оновленням, PHP 8 і Composer-бібліотеки гарантують сумісність і продуктивність, MySQL/InnoDB – безпечне зберігання даних, а Vite прискорює цикл «змінив-перевірив». Локальне середовище MAMP і VS Code закривають потреби розробника в уніфікованому середовищі.

У межах п’ятого завдання була виконана практична реалізація клієнтської частини (Vue-компоненти для каталогу авто, новин, форми замовлення, адмін-інтерфейсу) і серверної логіки (PHP-скрипти для CRUD-операцій), а REST-API з Axios забезпечує прозорий обмін JSON-даними між фронтендом і бекендом.

Шосте завдання охоплювало тестування: PHPUnit-модульні тести для моделей і контролерів, Jest із Vue Test Utils для компонентів, Postman-інтеграційні перевірки всіх кінцевих точок, а також ручне кросбраузерне й адаптивне тестування UI. Усі виявлені дефекти (помилки фільтрації, обмеження розміру файлів, некоректності стилів) було усунено.

Нарешті, сьоме завдання зосереджувалося на безпеці: увімкнено HTTPS із HSTS і CSP, аутентифікацію на основі PHP-сесій із bcrypt-хешуванням, CSRF-токени, PDO-prepared statements для уникнення SQL-ін’єкцій, екранізацію введених даних для запобігання XSS та безпечну обробку файлів із MIME-перевіркою й обмеженням розміру. Такі заходи створили надійний бар’єр проти сучасних кіберзагроз.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Car imports into ukraine up 67 % in Q1 2024 – customs service. Interfax. URL: https://interfax.com/newsroom/top-stories/102721/ (date of access: 03.02.2025).

2. 71 % імпортованих авто в Україну у 2024 році були вживані – Опендатабот. Опендатабот – відкриті дані про компанії, ФОП, суди та нерухомість України. URL: https://opendatabot.ua/analytics/import-cars-2024-7 (дата звернення: 12.02.2025).

3. Імпорт авто в Україну скоротився майже на 50 % у 2023 році. Економічна правда. URL: https://epravda.com.ua/news/ 2024/01/15/708737/ (дата звернення: 19.02.2025).

4. В Україні зріс попит на вживані авто з США майже вполовину. Найпопулярніші моделі. Бізнес. Цензор. URL: https://censor.net/biz/news/ 3531588/v-ukrayini-zris-popyt-na-vjyvani-avto-z-ssha-mayije-vpolovynu-nayipopularnishi-modeli (дата звернення: 23.02.2025).

5. Аналітика імпорту автомобілів зі США в Україну 2024 року. Інститут досліджень авторинку. Аналітика та прогнозування. URL: https://eauto.org.ua/ news/655-analitika-importu-avtomobiliv-zi-ssha-v-ukrajinu-2024-roku (дата звернення: 03.03.2025).

6. Авто із США у Луцьку. A_Dream. URL: https://adream.com.ua/ avto-iz-ssha/lutsk/ (дата звернення: 09.03.2025).

7. Авто зі США в Луцьку купити разом із компанією – КАРГО ЛАЙН. URL: https://cargoline.com.ua/ua/lutsk/ (дата звернення: 18.03.2025).

8. Авто з США у Луцьку купити в Україні на замовлення під ключ в Lube Avto. Авто з США купити в Україні разом з Lube Avto. URL: https://lubeavto.com.ua/store/carsfromusaincity/lutsk (дата звернення: 26.03.2025).

9. Contributors to Wikimedia projects. Unified Modeling Language – Wikipedia. Wikipedia, the free encyclopedia. URL: https://en.wikipedia.org/wiki/ Unified_Modeling_Language (date of access: 11.04.2025).

10. MVC – Glossary. MDN Web Docs. URL: https://developer.mozilla.org/en-US/docs/Glossary/MVC (date of access: 15.04.2025).

11. Front-end Frameworks. State of JavaScript 2024. URL: https://2024.stateofjs.com/en-US/libraries/front-end-frameworks (date of access: 19.04.2025).

12. PHP Usage Statistics and Popularity in 2025. ZenRows. URL: https://www.zenrows.com/blog/php-usage-statistics (date of access: 21.04.2025).

13. The State of PHP 2024 – Expert review. The JetBrains Blog. URL: https://blog.jetbrains.com/phpstorm/2025/02/state-of-php-2024/ (date of access: 24.04.2025).

14. DB-Engines Ranking Open Source vs. Commercial DBMS. DB-Engines – Knowledge Base of Relational and NoSQL Database Management Systems. URL: https://db-engines.com/en/ranking_osvsc (date of access: 27.04.2025).

15. Vite Usage Statistics. URL: https://trends.builtwith.com/framework/Vite (date of access: 01.05.2025).

16. MAMP, MAMP PRO & NAMO Documentation. URL: https://documentation.mamp.info/ (date of access: 09.05.2025).

17. Microsoft. Documentation for Visual Studio Code. Visual Studio Code – Code Editing. Redefined. URL: https://code.visualstudio.com/docs (date of access: 12.05.2025).

18. Vue Test Utils. Vue Test Utils for Vue.js 3. URL: https://test-utils.vuejs.org/ (date of access: 15.05.2025).

19. Jest. Delightful JavaScript Testing. URL: https://jestjs.io/uk/ (date of access: 16.05.2025).

20. Postman. The World's Leading API Platform. URL: https://www.postman.com/ (date of access: 18.05.2025).

ДОДАТКИ

Додаток А

Код файлу src/router/index.js

```
import { createRouter, createWebHistory } from 'vue-router'
import Home from '../views/Home.vue'
import AvailableCars from '../views/AvailableCars.vue'
import OrderForm from '../views/OrderForm.vue'
import News from '../views/News.vue'
import Contacts from '../views/Contacts.vue'
import AdminLayout from '../views/admin/AdminLayout.vue'
import AdminLogin from '../views/admin/Login.vue'
import AdminCars from '../views/admin/Cars.vue'
import AdminNews from '../views/admin/News.vue'
import AdminOrders from '../views/admin/Orders.vue'

const routes = [
 {
   path: '/',
   name: 'Home',
   component: Home
 },
 {
   path: '/avto-v-najavnosti',
   name: 'AvailableCars',
   component: AvailableCars
 },
 {
   path: '/zamovyty-avto',
   name: 'OrderForm',
   component: OrderForm
 },
 {
   path: '/novyny',
   name: 'News',
   component: News
 },
 {
   path: '/kontakty',
   name: 'Contacts',
   component: Contacts
 },
 {
   path: '/admin/login',
   name: 'AdminLogin',
   component: AdminLogin
 },
 {
   path: '/admin',
   component: AdminLayout,
   meta: { requiresAuth: true },
   children: [
     {
       path: 'cars',
```

```
        name: 'AdminCars',
        component: AdminCars
      },
      {
        path: 'news',
        name: 'AdminNews',
        component: AdminNews
      },
      {
        path: 'orders',
        name: 'AdminOrders',
        component: AdminOrders
      }
    ]
  }
]

const router = createRouter({
 history: createWebHistory(),
 routes
})

// Перевірка автентифікації
router.beforeEach(async (to, from, next) => {
 if (to.matched.some(record => record.meta.requiresAuth)) {
   try {
                                const      response      =      await
fetch('http://localhost/car-usa/api/admin/auth.php?action=check',
{
        credentials: 'include'
      })
      const data = await response.json()

      if (!data.authenticated) {
        next('/admin/login')
      } else {
        next()
      }
   } catch (error) {
      console.error('Auth check error:', error)
      next('/admin/login')
   }
 } else {
   next()
 }
})

export default router
```